UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

"DISPOSITIVO LÚDICO PARA RECONOCIMIENTO DE LÉXICO DACTILOLÓGICO PARA NIÑOS DE 6 A 8 AÑOS"

Trabajo de titulación previo a la obtención del título en Telecomunicaciones

Línea de investigación: Producción industrial y tecnología sostenible

AUTOR:

Alfonsina Sheraseth Chamorro Lima

DIRECTOR:

MSc. Jaime Roberto Michilena Calderon

Ibarra – Ecuador 2026

UNIVERSIDAD TÉCNICA DEL NORTE
DIRECCIÓN DE BIBLIOTECA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
APELLIDOS Y NOMBRES:	CHAMORRO LIMA ALFONSINA SHERASETH

DATOS DE LA OBRA	
TÍTULO:	"DISPOSITIVO LÚDICO PARA RECONOCIMIENTO DE LÉXICO DACTILOLÓGICO PARA NIÑOS DE 6 A 8 AÑOS"
AUTOR (ES):	CHAMORRO LIMA ALFONSINA SHERASETH
FECHA: DD/MM/AAAA	16/07/2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** □ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN TELECOMUNICACIONES
DIRECTOR /ASESOR:	MSc. JAIME ROBERTO MICHILENA CALDERÓN MSc. FABIÁN GEOVANNY CUZME RODRÍGUEZ

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 20 días del mes de julio de 2026

EL AUTOR:

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
ALFONSINA SHERASETH
CHAMORRO LIMA

.....

Nombre: Chamorro Lima Alfonsina Sheraseth

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra,

MSc. JAIME ROBERTO MICHILENA CALDERÓN

DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
JAIME ROBERTO MICHILENA CALDERON

(f) ……

Ing. Jaime Roberto Michilena Calderón, MsC.

C.C.: 1002198438

DEDICATORIA

"Hay amores que nos sostienen con sus manos, otros con su recuerdo,
y algunos con su silenciosa compañía; pero todos dejan en nosotros
la fuerza necesaria para continuar."

Dedico este trabajo a mi abuelita Laura, cuyo amor y enseñanzas siguen iluminando mi camino. Aunque ya no esté físicamente, vive en mis recuerdos y en todo lo que sembró en mí. Quiero creer que, desde el cielo, celebra conmigo este logro.

A mi mamá Golda, mi fuerza y mi refugio, por cada sacrificio, por creer en mí aun cuando yo dudaba y por enseñarme a nunca rendirme. Este logro también es suyo.

A Kevin, mi amor y compañero, por caminar a mi lado, abrazar mis sueños como propios y transformar, con su paciencia y ternura, el cansancio en esperanza.

A Fideo, mi compañero de desvelos y silencios, por regalarme calma con su presencia; y a Caramelo y Leo, cuyo amor puro dejó huellas eternas en mi corazón. Aunque ya no escuche sus pasos, su recuerdo continúa acompañándome.

Chamorro Lima Alfonsina Sheraseth

AGRADECIMIENTO

A Dios, por haber sido mi guía, mi fortaleza y mi refugio durante todo este camino. Gracias por sostenerme en los momentos de incertidumbre, por darme sabiduría para continuar y por permitirme alcanzar una meta que, en muchas ocasiones, parecía lejana.

A mis hermanos, Mateo, Anshel y Abigail, por ser parte esencial de mi vida, por su cariño, su compañía y por recordarme siempre el valor de la familia. Cada uno, a su manera, ha sido motivo de alegría y una fuerza importante para seguir adelante.

A Carmen, Víctor, Pame, Emi, Cinthya y a la pequeña Madi, por recibirme con cariño, hacerme sentir parte de su familia y acompañarme con su afecto durante este proceso. Gracias por cada palabra de ánimo, por los momentos compartidos y por brindarme un espacio lleno de calidez.

A mis amigos, los que estuvieron desde el inicio, los que llegaron durante el camino y los que, aun desde la distancia, nunca dejaron de hacerme sentir acompañada. En especial, a Erick, Jhon, Liss, Thalía, Xavi y Kim por su amistad sincera, por escucharme, animarme y compartir conmigo tantas alegrías y dificultades. A cada amigo que dejó una palabra de aliento, una risa o un gesto de cariño en este proceso, gracias por hacer que el camino hacia esta meta fuera mucho más bonito.

Expreso mi sincero agradecimiento a mi director, MSc. Jaime Michilena, por su orientación, compromiso y disposición durante el desarrollo de este trabajo. Sus conocimientos y observaciones fueron fundamentales para alcanzar este resultado.

De igual manera, agradezco a mi asesor, MSc. Fabián Cuzme, por su acompañamiento, paciencia y valiosos aportes, los cuales contribuyeron significativamente a la realización de este proyecto.

Finalmente, agradezco a todas las personas que, de una u otra manera, formaron parte de este proceso. Cada palabra de aliento, cada consejo y cada muestra de confianza dejaron una huella en este logro, que hoy también comparto con ustedes.

Chamorro Lima Alfonsina Sheraseth

RESUMEN

El presente trabajo aborda el diseño de un dispositivo lúdico para el reconocimiento automático de léxico dactilológico, orientado a niños de 6 a 8 años. La propuesta surge ante la falta de herramientas tecnológicas, visuales e interactivas que apoyen la práctica de señas, favorezcan la retroalimentación durante el aprendizaje y faciliten el acompañamiento docente en contextos de educación inclusiva. Para el desarrollo del sistema se aplicó una metodología en cascada, considerando las etapas de análisis, diseño, implementación y validación.

En la fase inicial se identificaron las necesidades del sistema mediante una entrevista semiestructurada y la revisión de trabajos relacionados, lo que permitió definir los requerimientos funcionales, no funcionales, de arquitectura y del dataset. Posteriormente, se seleccionaron los componentes de hardware y software, y se diseñó una arquitectura basada en procesamiento local mediante una placa NVIDIA Jetson Nano, una cámara IMX219, una pantalla táctil y una carcasa adecuada para el uso del prototipo. A nivel de software, el sistema fue implementado en Python bajo un esquema cliente-servidor, utilizando Flask como servidor local y SQLite para almacenar la información generada durante las sesiones de práctica. El reconocimiento de señas se realiza mediante la captura de gestos con cámara, el procesamiento de imágenes, la detección de la mano y la extracción de puntos de referencia con MediaPipe Hands y OpenCV. Estos datos son procesados por modelos desarrollados con TensorFlow/Keras y ejecutados mediante TensorFlow Lite, considerando señas estáticas, como letras y números, y señas dinámicas asociadas a palabras. Además, el prototipo incorpora una interfaz visual gamificada para el estudiante y un dashboard docente destinado a la gestión de actividades y seguimiento del progreso. Finalmente se realizó pruebas de reconocimiento, funcionamiento del sistema integrado, almacenamiento de datos y exportación de resultados para análisis. Los resultados permitieron comprobar que el prototipo funciona como una herramienta tecnológica de apoyo al aprendizaje dactilológico y a la inclusión educativa.

Palabras clave: léxico dactilológico, visión artificial, inteligencia artificial, MediaPipeHands, TensorFlow Lite, educación inclusiva.

ABSTRACT

The present study proposes the design of a playful interactive device for the automatic recognition of sign language vocabulary, aimed at children aged 6 to 8. The proposal arises from the lack of technological, visual, and interactive tools that support sign language practice, provide feedback during the learning process, and facilitate teacher guidance in inclusive educational settings.

A waterfall methodology was applied for the development of the system, considering the stages of analysis, design, implementation, and validation. In the initial phase, the system's needs were identified through a semi-structured interview and a review of related studies, which made it possible to define the functional, non-functional, architectural, and dataset requirements.

Subsequently, the hardware and software components were selected, and an architecture based on local processing was designed using an NVIDIA Jetson Nano board, an IMX219 camera, a touchscreen, and a suitable case for the prototype. At the software level, the system was implemented in Python under a client-server architecture, using Flask as the local server and SQLite to store the information generated during practice sessions.

Sign recognition is performed through camera-based gesture capture, image processing, hand detection, and landmark extraction using MediaPipe Hands and OpenCV. These data are processed by models developed with TensorFlow/Keras and deployed using TensorFlow Lite, considering both static signs, such as letters and numbers, and dynamic signs associated with words.

In addition, the prototype incorporates a gamified visual interface for students and a teacher dashboard intended for activity management and progress monitoring. The validation process included recognition tests, integrated system functionality tests, data storage verification, and

result export for analysis. The results confirmed that the prototype functions as a technological tool to support sign language vocabulary learning and educational inclusion.

Keywords: dactylological lexicon, computer vision, artificial intelligence, MediaPipeHands, TensorFlow Lite, inclusive education.

Índice de Contenido

CAPÍTULO I. ANTECEDENTES 21
1.1. Tema 21
1.2. Problema 21
1.3. Objetivos 23
1.3.1. Objetivo general 23
1.3.2. Objetivos específicos 23
1.4. Alcance 24
1.5. Justificación 26
2 CAPÍTULO II. FUNDAMENTACIÓN TEÓRICA 28
2.1 Discapacidad Auditiva 29
2.1.1 Clasificación de la discapacidad auditiva 29
2.1.1.1 Clasificación Según la Localización del Daño 29
2.1.1.2 Clasificación Según el Momento de Aparición 30
2.1.1.3 Clasificación Según la Gravedad 31
2.1.2 Impacto y Necesidades Educativas en la Infancia 31
2.1.2.1 Desarrollo Lingüístico y Cognitivo 31
2.1.2.2 Interacción Social y Barreras Comunicativas 32
2.1.2.3 Educativo Rendimiento Académico y Estrategias Tecnológicas 32
2.2 Léxico Dactilológico y Sistemas de Lengua de Señas 33
2.2.1 Definición y origen del léxico dactilológico 33
2.2.2 ASL como referencia dactilológica y adaptaciones locales 34
2.3 Estrategias Lúdicas e Interactivas en el Aprendizaje Infantil 34
2.3.1 Diseño de Interfaces, Gamificación y Retroalimentación Visual 35
2.3.2 Personalización y Adaptabilidad del Sistema 36
2.4 Inteligencia Artificial y Visión Artificial Aplicada 37
2.4.1 Técnicas de extracción de características espaciales 39
2.4.2 Algoritmos de clasificación para gestos estáticos 40
2.4.3 Algoritmos secuenciales para gestos dinámicos 40
2.5 Hardware para sistemas de procesamiento visual 41
2.5.1 Plataformas de procesamiento: móviles, microcontroladores y SBC 42
2.5.2 Cámaras y sensores para captura de movimiento 44
2.5.3 Procesamiento local en sistemas embebidos 45
2.5.4 Software para reconocimiento visual 46
2.5.4.1 Herramientas de visión artificial 46
2.5.4.2 Frameworks de aprendizaje automático 47
2.5.4.3 Aplicaciones locales, base de datos y reportes 48

3 CAPÍTULO III. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA 50
3.1 Metodología de desarrollo 50
3.2 Fase de análisis 51
3.2.1 Situación Actual 52
3.2.2 Identificación de Actores Involucrados (Stakeholders) 53
3.2.3 Metodología y Técnicas de Recolección 53
3.2.3.1 Entrevista Semiestructurada 54
3.2.3.2 Revisión de Trabajos Relacionados 55
3.2.4 Necesidades del Sistema 56
3.2.4.1 Resultados de la entrevista semiestructurada 57
3.2.4.2 Resultados de la revisión de trabajos relacionados 58
3.2.4.3 Necesidades identificadas 60
3.2.5 Requerimientos del Sistema 61
3.2.5.1 Nomenclatura de requerimientos 62
3.2.5.2 Requerimientos de stakeholders 63
3.2.5.3 Requerimientos funcionales 65
3.2.5.4 Requerimientos no funcionales 67
3.2.5.5 Requerimientos de arquitectura 70
3.2.5.6 Requerimientos del dataset 72
3.2.5.7 Matriz de trazabilidad 75
3.3 Fase de Diseño 76
3.3.1 Selección de componentes tecnológicos 77
3.3.1.1 Criterios de selección 77
3.3.1.2 Selección de Hardware 79
3.3.1.1 Selección de Software 90
3.3.1.1.1 Selección de Dataset 104
3.3.2 Arquitectura del sistema 111
3.3.2.1 Conexiones físicas y comunicación local del prototipo 115
3.3.2.2 Funcionamiento lógico y conectividad del sistema 116
3.3.2.1 Consideraciones y dimensionamiento energético de operación 119
3.3.3 Diseño del dataset 124
3.3.3.1 Estructura general del dataset 125
3.3.3.2 Representación mediante landmarks 126
3.3.3.3 Organización procesada para señas estáticas y dinámicas 128
3.3.3.4 Etiquetado de las muestras 130
3.3.3.1 División del dataset 130
3.3.4 Diseño del modelo de reconocimiento 131

3.3.4.1 Modelo para señas estáticas 132
3.3.4.2 Modelo para señas dinámicas 133
3.3.4.3 Consideraciones del diseño del modelo 134
3.3.5 Diseño de la base de datos 134
3.3.5.1 Entidades principales 134
3.3.5.2 Modelo entidad-relación 135
3.3.5.3 Datos registrados durante una sesión 136
3.3.5.4 Relación con el dashboard y reportes 137
3.3.6 Diseño de interfaces 138
3.3.6.1 Interfaz del estudiante 139
3.3.6.2 Dashboard docente 140
3.3.6.3 Diseño de reportes 141
3.4 Fase de Implementación 142
3.4.1 Implementación del entorno de hardware 142
3.4.1.1 Conexión de componentes electrónicos 142
3.4.1.2 Integración en el envolvente mecánico 143
3.4.2 Implementación del entorno de software 144
3.4.3 Captura y preparación del dataset 146
3.4.3.1 Recolección y organización de las muestras 146
3.4.3.2 Procesamiento espacial de las señas estáticas 149
3.4.3.3 Procesamiento espacial de las señas estáticas 152
3.4.3.4 División del dataset 153
3.4.4 Entrenamiento de los modelos de reconocimiento 154
3.4.5 Conversión e integración de los modelos entrenados 159
3.4.6 Integración y ejecución de los modelos en main.py 161
3.4.6.1 Carga de los modelos y las etiquetas 161
3.4.6.2 Configuración de la cámara y MediaPipe Hands 162
3.4.6.3 Inferencia de las señas estáticas y dinámicas 164
3.4.6.4 Comparación, retroalimentación y registro 165
3.4.6.5 Ejecución simultánea de la interfaz y el dashboard 165
3.4.7 Implementación de la interfaz del estudiante 166
3.4.8 Implementación del dashboard docente 168
3.4.9 Implementación de la base de datos y reportes 171
3.4.9.1 Exportación de resultados para la validación 173
3.4.10 Manual de usuario del sistema 174
4 CAPÍTULO IV. PRUEBAS DE FUNCIONAMIENTO 175
4.1 Evaluación de los modelos de reconocimiento 175

4.1.1 Evaluación del modelo estático para el abecedario 176
4.1.2 Evaluación del modelo estático para números 177
4.1.3 Evaluación del modelo dinámico para palabras 178
4.2 Escenario de pruebas 179
4.3 Plan de pruebas 181
4.3.1 Pruebas funcionales y de integración del sistema 182
4.3.1.1 Acceso y autenticación 183
4.3.1.2 Gestión y selección del estudiante 185
4.3.1.3 Configuración e inicio de la actividad 187
4.3.1.4 Captura y detección de la mano 190
4.3.1.5 Reconocimiento de señas estáticas 191
4.3.1.6 Reconocimiento de palabras dinámicas 193
4.3.1.7 Retroalimentación y controles 194
4.3.1.8 Registro del resultado 195
4.3.1.9 Consulta de resultados e historial 197
4.3.1.10 Exportación administrativa de resultados 200
4.3.1.11 Resumen de las pruebas funcionales y de integración 202
4.3.2 Pruebas de rendimiento 203
4.4 Validación con los niños participantes 205
4.4.1 Resultados por categoría 207
4.4.2 Resultados individuales 209
4.4.2.1 Usuarios Oyentes 209
Estudiante 1 209
Estudiante 2 210
4.4.2.2 Usuarios No Oyentes 210
Estudiante 3 211
Estudiante 4 211
Estudiante 5 212
4.4.3 Comparación entre usuarios oyentes y no oyentes 212
4.5 Validación Operativa y Calidad del Sistema (ISO/IEC 25010) 214
5 Conclusiones 219
6 Recomendaciones 221
7 REFERENCIAS 223
8 ANEXOS 226
Anexo A: Formato Entrevista semiestructurada a usuario especialista 226
Anexo A.1: Imagen de entrevista 228
Anexo B: Matriz de revisión de trabajos relacionados 229

Trabajo relacionado 1 229
Trabajo relacionado 2 230
Trabajo relacionado 3 232
Trabajo relacionado 4 233
Trabajo relacionado 5 235
Anexo C: Matrices comparativas para la selección del hardware 237
C.1 Criterios generales de comparación 237
C.2 Comparación de las placas embebidas 238
C.3 Comparación de los dispositivos de captura visual 241
C.4 Comparación de los medios de interacción visual 244
C.5 Comparación de los medios de almacenamiento 246
C.6 Comparación de los medios de conectividad 248
C.7 Comparación de las alternativas de alimentación 250
C.8 Comparación de las alternativas de estructura física 251
Anexo D: Matrices comparativas para la selección del software 254
D.1 Criterios generales considerados para la selección del software 254
D.2 Comparación de los sistemas operativos 255
D.3 Comparación de los lenguajes de programación 258
D.4 Comparación de las herramientas de procesamiento de imágenes 260
D.5 Comparación de las herramientas de detección de manos y extracción de características 262
D.6 Comparación de los frameworks de aprendizaje automático 264
D.7 Herramientas para la inferencia y el despliegue de los modelos 267
D.8 Herramientas utilizadas para el servidor local, la base de datos y la interfaz web 269
Anexo E: Resultados detallados de los modelos y matrices de confusión 273
Anexo F: Consentimiento Informado a Padres de Familia 302
Anexo G: Evidencias detalladas de la validación con los niños participantes 304
G.1 Resultados detallados por participante y seña 304
G.1.1 Estudiante 1 304
G.1.2 Estudiante 2 305
G.1.3 Estudiante 3 306
G.1.4 Estudiante 4 308
G.1.5 Estudiante 5 309
G.2 Matriz consolidada por seña y grupo evaluado 310
G.2.1 Resultados del grupo oyente 310
G.2.2 Resultados del grupo no oyente 311
G.3 Comparación detallada entre los grupos 312
Anexo H: Encuesta de Funcionamiento 314

Anexo H.1: Formato de encuesta de funcionamiento 314
Anexo H.2: Encuesta de funcionamiento realizada a docente 317
Anexo I: Manual de uso del sistema..... 319

Índice de figuras

Figura 1 *Árbol de Problemas* 22
Figura 2 Arquitectura del Proyecto 25
Figura 3 Interfaz interactiva en el diseño de juegos educativos. 35
Figura 4 Gamificación 36
Figura 5 Beneficios de la personalización en los videojuegos para niños 37
Figura 6 Esquema de un sistema de visión artificial. 38
Figura 7 Flujo general del procesamiento local en un sistema de reconocimiento visual 46
Figura 8 Arquitectura general del sistema 112
Figura 9 Diagrama de bloques de la arquitectura del sistema 113
Figura 10 Conexiones físicas y comunicación local del prototipo 116
Figura 11 Flujo lógico de funcionamiento y conectividad del sistema 117
Figura 12 Representación de los 21 landmarks 127
Figura 13 Organización de datos para señas estáticas y dinámicas 129
Figura 14 Diseño general de los modelos de reconocimiento de señas 132
Figura 15 Flujo general del modelo estático 133
Figura 16 Flujo general del modelo dinámico 133
Figura 17 Diagrama entidad-relación de la base de datos 136
Figura 18 Boceto de interfaces del sistema: (a) interfaz del estudiante y (b) dashboard docente 139
Figura 19 Conexión física de los componentes electrónicos principales 143
Figura 20 Ensamblaje final del sistema en la carcasa lúdica de madera. 144
Figura 21 Entorno de software configurado para el desarrollo del sistema 145
Figura 22 Estructura general del proyecto implementado 146
Figura 23 Muestras para la “CH” y la “T” 147
Figura 24 Captura del flujo de video desde la cámara del prototipo 148
Figura 25 Estructura de directorios del dataset 149
Figura 26 Algoritmo de preprocesamiento de imagen mediante CLAHE 150
Figura 27 Extracción de landmarks y normalización de coordenadas relativas a la muñeca 151
Figura 28 Archivos de datos serializados (.pickle) listos para el entrenamiento de los modelos 151
Figura 29 Algoritmo de segmentación temporal y almacenamiento de secuencias de 30 frames 152
Figura 30 Archivos de datos procesados en .npy listo para el entrenamiento 153
Figura 31 División del dataset 154
Figura 32 Función utilizada para crear los modelos del abecedario y los números 155
Figura 33 Función utilizada para crear el modelo LSTM de palabras 156
Figura 34 Cantidades de épocas evaluadas y ejecución del entrenamiento 157
Figura 35 Cálculo de las métricas 158
Figura 36 Condición utilizada para seleccionar el mejor modelo. 158
Figura 37 Resultados de la selección del número de épocas 159
Figura 38 Archivos generados al finalizar el entrenamiento de los modelos 159
Figura 39 Conversión de los modelos estáticos al formato TensorFlow Lite 160
Figura 40 Modelos estáticos convertidos al formato TensorFlow Lite 161
Figura 41 Carga de los modelos y archivos de etiquetas en main.py 162
Figura 42 Configuración de la cámara y detección de la mano 162
Figura 43 Detección de mano y landmarks mediante MediaPipe Hands 163
Figura 44 Inferencia de los modelos de reconocimiento 164
Figura 45 Comparación de la predicción y registro del resultado 165
Figura 46 Ejecución simultánea de la interfaz y el dashboard 166
Figura 47 Pantalla principal de la interfaz del estudiante 167
Figura 48 Retroalimentación visual durante la práctica de señas 168
Figura 49 Acceso al dashboard docente mediante red local 169

Figura 50 Pantalla principal del dashboard docente 170
Figura 51 Base de datos local implementada en SQLite 172
Figura 52 Registros almacenados durante una sesión de práctica 173
Figura 53 Generación del archivo CSV para el análisis de validación 174
Figura 54 Comparación de las métricas del modelo de abecedario según el número de épocas..... 177
Figura 55 Comparación de las métricas del modelo de números según el número de épocas 178
Figura 56 Comparación de las métricas del modelo de palabras según el número de épocas 179
Figura 57 Escenario de pruebas del prototipo integrado 180
Figura 58 Bloqueo de acceso por credenciales inválidas 183
Figura 59 Interfaz del panel principal con el rol de Docente..... 184
Figura 60 Interfaz del panel principal con el rol de Administrador 185
Figura 61 Registro inicial de usuario. 186
Figura 62 Edición de los datos del estudiante mediante cuadros de diálogo 186
Figura 63 Confirmación del estudiante activo en el panel principal..... 187
Figura 64 Verificación de los datos guardados en la base de datos SQLite 187
Figura 65 Bloqueo preventivo por falta de selección de estudiante 188
Figura 66 Configuración de las señas de la actividad..... 188
Figura 67 Sincronización con la interfaz del estudiante 189
Figura 68 Despliegue de los recursos pedagógicos visuales en la fase de aprendizaje..... 189
Figura 69 Captura de video sin presencia de la mano 190
Figura 70 Captura de video con presencia de la mano 191
Figura 71 Reconocimiento de una seña del abecedario..... 192
Figura 72 Reconocimiento de una seña numérica 192
Figura 73 Reconocimiento de gestos dinámicos en tiempo real..... 193
Figura 74 Ayuda visual presentada durante la actividad..... 194
Figura 75 Continuación de la actividad después de utilizar la opción de salto..... 195
Figura 76 Base de Datos 196
Figura 77 Consulta de resultados mediante el reporte matricial..... 198
Figura 78 Detalle de los resultados obtenidos en una seña..... 199
Figura 79 Historial detallado de las actividades realizadas 200
Figura 80 Opción administrativa para descargar los resultados 201
Figura 81 Archivo CSV generado para el análisis de resultados 201
Figura 82 Monitoreo del sistema durante el reconocimiento de una seña dinámica..... 205
Figura 83 Interfaz del estudiante durante el reconocimiento de una seña 207

Índice de Tablas

Tabla 1 Comparación general de plataformas de procesamiento para visión artificial 43
Tabla 2 Tecnologías de captura utilizadas en reconocimiento de señas 44
Tabla 3 Herramientas de software utilizadas en sistemas de reconocimiento visual 47
Tabla 4 Frameworks de inteligencia artificial y función general 48
Tabla 5 Herramientas para interfaz, almacenamiento y reportes 49
Tabla 6. Descripción de los Stakeholders involucrados 53
Tabla 7. Trabajos relacionados analizados para la identificación de requerimientos 55
Tabla 8. Resultados de la entrevista semiestructurada 57
Tabla 9. Resultados de la revisión de trabajos relacionados 59
Tabla 10. Necesidades identificadas del sistema 60
Tabla 11. Abreviaturas de Requerimientos 62
Tabla 12 Prioridades de los Requerimientos 63
Tabla 13. Requerimientos de stakeholders 63
Tabla 14. Requerimientos funcionales 65
Tabla 15 Requerimientos no funcionales 68
Tabla 16 Requerimientos de arquitectura 70
Tabla 17 Requerimientos del dataset 73
Tabla 18 Matriz de trazabilidad de requerimientos 75
Tabla 19 Criterios de selección de componentes tecnológicos 78
Tabla 20 Escala de valoración para la selección de componentes 79
Tabla 21 Evaluación y selección de la placa embebida 80
Tabla 22 Selección del dispositivo de captura visual 82
Tabla 23 Selección del Medio de interacción visual 83
Tabla 24 Selección del medio de almacenamiento 84
Tabla 25 Selección del medio de conectividad 85
Tabla 26 Selección de la fuente de alimentación 86
Tabla 27 Selección de la estructura física 88
Tabla 28 Resumen de hardware seleccionado 89
Tabla 29 Selección del Sistema Operativo 91
Tabla 30 Selección del lenguaje de programación 92
Tabla 31 Selección de la herramienta de procesamiento de imágenes 94
Tabla 32 Selección de la herramienta para detección de mano 95
Tabla 33 Selección del framework de inteligencia artificial 97
Tabla 34 Selección del motor de inferencia local 98
Tabla 35 Selección del servidor local 100
Tabla 36 Selección de base de datos y exportación de resultados 101
Tabla 37 Resumen de software seleccionado 102
Tabla 38 Selección del origen del dataset 105
Tabla 39 Selección de la representación de los datos 106
Tabla 40 Selección de la organización del dataset 108
Tabla 41 Selección de la división del dataset 109
Tabla 42 Resumen del dataset seleccionado 110
Tabla 43 Estimación de consumo eléctrico de los componentes principales del sistema 120
Tabla 44 Estructura general del dataset 125
Tabla 45 División del dataset 131
Tabla 46 Entidades principales de la base de datos 135
Tabla 47 Datos registrados durante una sesión de práctica 137
Tabla 48 Elementos considerados en la interfaz del estudiante 140
Tabla 49 Elementos considerados en el dashboard docente 140

Tabla 50 Datos considerados para los reportes de seguimiento 141
Tabla 51 Parámetros utilizados para el entrenamiento de los modelos 156
Tabla 52 Datos almacenados en la tabla registro_practicas 171
Tabla 53 Comparación del F1-score ponderado según el número de épocas 175
Tabla 54 Condiciones generales del escenario de pruebas 180
Tabla 55 Plan general de pruebas del sistema 181
Tabla 56 Comparación de los resultados exportados con SQLite 202
Tabla 57 Resumen de resultados de las pruebas funcionales 202
Tabla 58 Mediciones registradas durante el funcionamiento continuo 204
Tabla 59 Resumen de la prueba de funcionamiento continuo 205
Tabla 60 Distribución de los participantes de la prueba 206
Tabla 61 Resultados de las pruebas con usuarios por categoría 208
Tabla 62 Resultados individuales del Estudiante 1 209
Tabla 63 Resultados individuales del Estudiante 2 210
Tabla 64 Resultados individuales del Estudiante 3 211
Tabla 65 Resultados individuales del Estudiante 4 211
Tabla 66 Resultados individuales del Estudiante 5 212
Tabla 67 Resultados registrados según el grupo de usuarios 212
Tabla 68 Resultados de la validación operativa bajo el modelo de calidad de software ISO/IEC 25010 215

CAPÍTULO I. ANTECEDENTES

1.1. Tema

Dispositivo lúdico para reconocimiento de léxico dactilológico para niños de 6 a 8 años

1.2. Problema

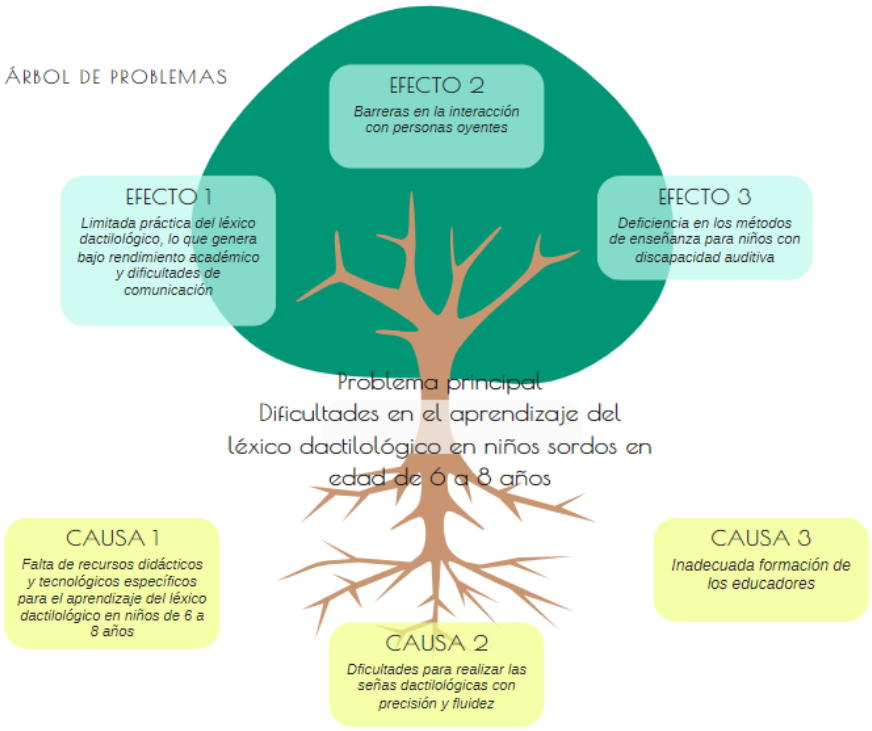
En Ecuador, los niños sordos de entre 6 y 8 años enfrentan barreras significativas para aprender el léxico dactilológico, una herramienta fundamental para su desarrollo lingüístico, social y cognitivo. Este rango etario es crucial porque durante esta etapa se consolidan habilidades esenciales para la comunicación y la interacción social. Según el Instituto Nacional de Estadística y Censos (Instituto Nacional de Estadística y Censos, 2023), el 3,01 % de la población ecuatoriana, equivalente a unas 540,650 personas, presenta algún tipo de discapacidad. De estas, cerca del 12 % tiene discapacidad auditiva, representando aproximadamente 64,878 personas. En la provincia de Imbabura, se estima que alrededor de 2,500 personas enfrentan esta condición, entre ellas un número significativo de niños en etapa escolar, lo que resalta la necesidad urgente de estrategias inclusivas específicas para esta población (Instituto Nacional de Estadística y Censos, 2023).

La falta de herramientas lúdicas y tecnológicas diseñadas para facilitar el aprendizaje del léxico dactilológico limita significativamente el desarrollo de habilidades comunicativas en niños de 6 a 8 años. Esta etapa es determinante, ya que en estos años se desarrollan competencias como la asociación de conceptos, la comprensión del lenguaje y la coordinación motora, que son fundamentales para el aprendizaje de señas dactilológicas. La ausencia de recursos interactivos y tecnológicos, como dispositivos móviles con algoritmos de inteligencia artificial, afecta directamente su capacidad para practicar y mejorar sus habilidades de comunicación. Según el Consejo Nacional para la Igualdad de Discapacidades, la carencia de tecnologías inclusivas no solo dificulta el aprendizaje, sino que perpetúa las desigualdades en

el acceso a una educación de calidad (Consejo Nacional para la Igualdad de Discapacidades, 2018). La Ley Orgánica de Educación Intercultural también subraya la obligación de garantizar la igualdad de oportunidades para todos los estudiantes, incluyendo aquellos con discapacidades auditivas, destacando la importancia de soluciones educativas adaptadas (Asamblea Nacional del Ecuador, 2024).

El árbol de problemas en la Figura 1 resume las causas principales de esta problemática, entre las que se incluyen la falta de recursos didácticos y tecnológicos, dificultades para realizar señas con precisión debido a metodologías de enseñanza limitadas, y la formación inadecuada de los docentes en el uso de herramientas tecnológicas. Estos factores generan efectos negativos como bajo rendimiento académico, falta de motivación y barreras en la interacción social con personas oyentes. Por ejemplo, los niños con discapacidad auditiva suelen tener dificultades para asociar gestos dactilológicos con palabras escritas, lo que afecta su aprendizaje y autoestima (Organización Mundial de la Salud, 2021).

Figura 1
Árbol de Problemas



Nota. Autoría propia

Este contexto evidencia la necesidad de desarrollar herramientas que incorporen un enfoque lúdico e interactivo para la enseñanza del léxico dactilológico. Un dispositivo que utilice dispositivos móviles y tecnologías de inteligencia artificial para capturar y procesar gestos podría transformar el aprendizaje al permitir a los niños practicar de manera autónoma, recibir retroalimentación inmediata y fortalecer tanto su motivación como su rendimiento académico. Además, esta propuesta busca empoderar a docentes y familias mediante recursos accesibles y adaptados, promoviendo una educación inclusiva que responda a las necesidades específicas de los niños de 6 a 8 años.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un dispositivo lúdico e interactivo para el reconocimiento automático del léxico dactilológico en niños de 6 a 8 años, utilizando inteligencia artificial y dispositivos móviles, facilitando el aprendizaje del lenguaje de señas y promoviendo la inclusión educativa

1.3.2. Objetivos específicos

- Investigar las tecnologías actuales en inteligencia artificial, reconocimiento de imágenes, diseño de interfaces lúdicas, aplicaciones móviles y metodologías de aprendizaje del lenguaje de señas para fundamentar el desarrollo del prototipo
- Determinar los requerimientos de software, hardware y algoritmos necesarios para el diseño del dispositivo lúdico.
- Diseñar un dispositivo lúdico que capture gestos dactilológicos mediante la cámara de un dispositivo móvil y los procese con algoritmos de inteligencia artificial.
- Validar el funcionamiento del prototipo mediante pruebas con usuarios que permitan evaluar la precisión y la experiencia de uso.

1.4. Alcance

El proyecto consistirá en el desarrollo de un dispositivo lúdico basado en inteligencia artificial para el reconocimiento de léxico dactilológico en niños de 6 a 8 años, utilizando el abecedario dactilológico del American Sign Language (ASL) como referencia. Este dispositivo aprovechará las cámaras integradas en dispositivos móviles para capturar gestos, que serán procesados mediante algoritmos de inteligencia artificial especializados en reconocimiento de imágenes. La arquitectura del sistema empleará una estructura cliente-servidor que permitirá el procesamiento eficiente de datos y la visualización de resultados de forma oportuna para los usuarios. Las letras dinámicas J y Z no serán consideradas debido a las limitaciones en el reconocimiento de movimientos complejos, priorizando gestos estáticos que se alinean con el alcance técnico y educativo del sistema (Goodfellow et al., 2016a).

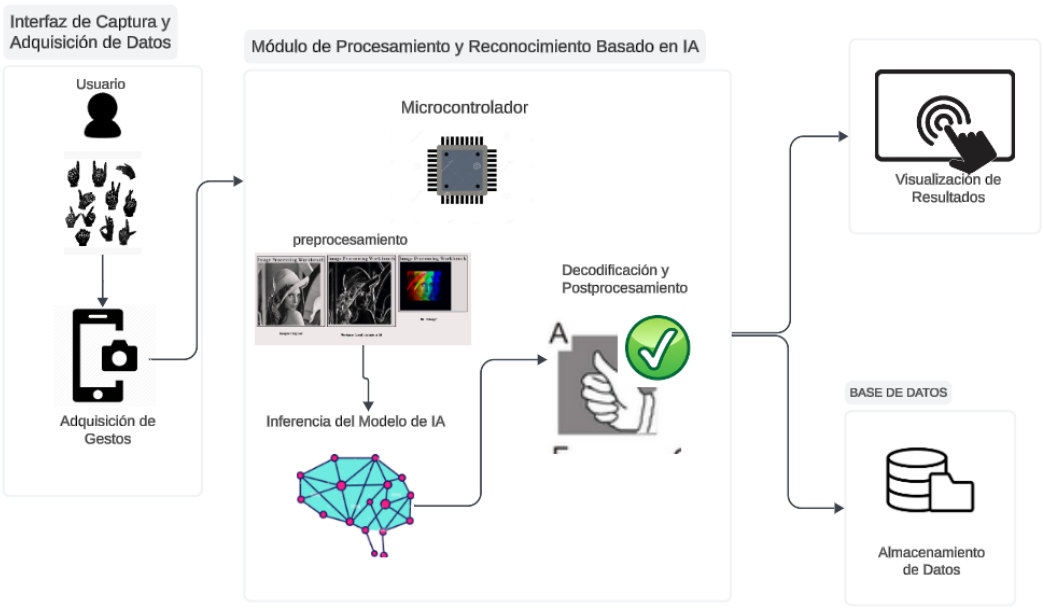
El proyecto será desarrollado bajo la metodología de cascada, la cual se compone de cinco fases secuenciales: análisis, diseño, implementación, verificación y mantenimiento. Esta metodología asegura un desarrollo estructurado, donde cada etapa se completa antes de iniciar la siguiente, garantizando la calidad y eficacia del sistema en cada paso. (Pressman & Maxim, 2019).

En la fase de análisis, se recopilarán y evaluarán los requisitos técnicos y educativos necesarios para diseñar el dispositivo. Se investigarán tecnologías de inteligencia artificial, metodologías pedagógicas aplicables al rango etario de 6 a 8 años, y las limitaciones actuales en herramientas de aprendizaje del léxico dactilológico. Este análisis servirá de base para definir los objetivos técnicos y funcionales del prototipo.

Durante la fase de diseño, se desarrollará la arquitectura tecnológica del sistema, la cual integrará de manera estructurada los módulos principales: captura de gestos, procesamiento en el servidor y visualización de resultados en el dispositivo móvil. La Figura 2 representa esta

arquitectura y muestra cómo cada componente interactúa dentro del sistema. En ella, se destaca el flujo de información desde la captura de gestos mediante la cámara del dispositivo, pasando por el servidor local donde se procesarán los datos utilizando algoritmos de inteligencia artificial, hasta llegar a la interfaz de usuario que mostrará los resultados.

Figura 2
Arquitectura del Proyecto



Nota. Autoría propia

La fase de implementación se centrará en la implementación del dispositivo, integrando los componentes tecnológicos definidos en la etapa de diseño. Esto incluirá la programación de algoritmos de reconocimiento de imágenes, la configuración del servidor local y la integración de la interfaz con las funciones de captura y procesamiento de datos. Esta etapa será fundamental para garantizar que todos los módulos trabajen de manera cohesionada, cumpliendo con los objetivos específicos del proyecto.

En la fase de verificación, se validará la precisión del sistema para reconocer gestos estáticos, la eficiencia del procesamiento y la experiencia de usuario. Se trabajará con un grupo

de niños de 6 a 8 años en un entorno controlado, lo que permitirá evaluar la funcionalidad del dispositivo en condiciones reales. Los resultados de las pruebas serán analizados para identificar posibles ajustes o mejoras necesarias antes de la implementación final.

Finalmente, en la fase de mantenimiento, se realizará la documentación del sistema para facilitar futuras modificaciones y actualizaciones. Esta incluirá las especificaciones técnicas, diagramas y resultados obtenidos durante las pruebas, con el objetivo de garantizar la continuidad y mejora del sistema en el tiempo.

1.5. Justificación

El desarrollo de un sistema para el reconocimiento automático de léxico dactilológico en niños de 6 a 8 años responde a la necesidad urgente de mejorar la inclusión educativa en Ecuador, particularmente en el ámbito de la discapacidad auditiva. Este proyecto se alinea con los Objetivos de Desarrollo Sostenible (ODS), especialmente con el ODS 4: Educación de calidad, que busca garantizar una educación inclusiva y equitativa, y el ODS 10: Reducción de las desigualdades, al promover la igualdad de oportunidades para todos (Naciones Unidas, 2015). Además, se enmarca en los lineamientos de la Agenda Zonal 1 y la Planificación Nacional para el Buen Vivir (PNTV), que impulsan el uso de tecnologías para mejorar la calidad de vida en la región (Secretaría Nacional de Planificación y Desarrollo (Secretaría Nacional de Planificación y Desarrollo, 2017)

El sistema propuesto utiliza una arquitectura cliente-servidor, permitiendo el procesamiento local de los datos. Esto garantiza una transmisión eficiente y segura, asegurando la privacidad de la información, una prioridad en el ámbito de las telecomunicaciones. La herramienta empleará modelos de inteligencia artificial entrenados específicamente para el reconocimiento de gestos estáticos del abecedario dactilológico, optimizando el aprendizaje y permitiendo la personalización en la retroalimentación (Goodfellow et al., 2016b)

El impacto social es significativo, ya que este sistema contribuirá a la inclusión de niños con discapacidad auditiva, mejorando su comunicación y desempeño académico. Según el Instituto Nacional de Estadística y Censos (INEC), en 2023, el 3,01 % de la población ecuatoriana tenía algún tipo de discapacidad, siendo la auditiva una de las más prevalentes. En la ciudad de Ibarra, se estima que alrededor de 2,500 personas viven con esta condición (Instituto Nacional de Estadística y Censos, n.d.). Implementar una solución tecnológica de estas características no solo beneficiará a los estudiantes, sino que también fortalecerá las capacidades de docentes y familias al proporcionarles un recurso práctico para apoyar el proceso de enseñanza-aprendizaje (Consejo Nacional para la Igualdad de Discapacidades, n.d.).

Desde una perspectiva ambiental, la reducción en el uso de materiales impresos contribuye a una educación más sostenible, en línea con el ODS 12: Producción y consumo responsables (Naciones Unidas, 2015). Asimismo, el enfoque de procesamiento local minimiza la dependencia de infraestructuras más costosas y energéticamente demandantes, como los servicios en la nube (Google Cloud, n.d.).

El proyecto se plantea como una solución innovadora que integra telecomunicaciones e inteligencia artificial, ofreciendo un modelo adaptable a otras instituciones y contextos educativos. Su implementación no solo resuelve una problemática específica, sino que también establece las bases para futuras investigaciones y desarrollos en el ámbito de la educación inclusiva y las tecnologías de asistencia.

CAPÍTULO II. FUNDAMENTACIÓN TEÓRICA

Este capítulo reúne las bases teóricas necesarias para comprender el desarrollo del dispositivo lúdico propuesto. El proyecto se relaciona principalmente con el aprendizaje del léxico dactilológico en niños de 6 a 8 años, por lo que resulta importante partir del contexto de la discapacidad auditiva infantil y de las dificultades que pueden presentarse en los procesos de comunicación y aprendizaje.

También se toma en cuenta la Lengua de Señas Americana, ya que representa un medio importante para favorecer la comunicación de las personas con discapacidad auditiva. En este caso, su estudio permite entender mejor la importancia de practicar señas desde edades tempranas y de contar con recursos que hagan este proceso más visual y participativo.

Otro aspecto que se considera dentro del capítulo es el uso de estrategias lúdicas e interactivas. El prototipo no se plantea únicamente como una herramienta tecnológica, sino como un apoyo que busca motivar al niño durante la práctica y facilitar el acompañamiento de la docente. Por eso, se revisan conceptos relacionados con el aprendizaje mediante actividades dinámicas, materiales visuales y recursos adaptados al contexto educativo.

En la parte tecnológica se describen los fundamentos que permiten el funcionamiento del sistema. Se incluyen temas como visión artificial, procesamiento de imágenes, detección de la mano mediante puntos de referencia, aprendizaje automático e inteligencia artificial aplicada al reconocimiento de señas. También se revisan los elementos de hardware y software que forman parte del prototipo, como la placa de procesamiento, la cámara, la interfaz visual y las herramientas utilizadas para implementar el sistema.

2.1 Discapacidad Auditiva

La discapacidad auditiva implica limitaciones en la percepción sonora que afectan directamente los procesos de comunicación, desarrollo cognitivo y aprendizaje, especialmente en edades tempranas (Organización Mundial de la Salud, 2026)

2.1.1 Clasificación de la discapacidad auditiva

La clasificación de la discapacidad auditiva permite comprender sus características y planificar estrategias que respondan a las necesidades específicas de quienes la padecen. Los criterios principales para su clasificación son la localización del daño en el sistema auditivo y el momento de aparición de la pérdida auditiva.

2.1.1.1 Clasificación Según la Localización del Daño

La clasificación de la discapacidad auditiva según la localización del daño permite identificar las estructuras específicas del sistema auditivo que están afectadas, lo cual es fundamental para determinar el tratamiento y las estrategias educativas necesarias. Según esta clasificación, la discapacidad auditiva puede dividirse en los siguientes tipos:

- Hipoacusia Conductiva:

  Este tipo de pérdida auditiva afecta el oído externo o medio, impidiendo que el sonido llegue adecuadamente al oído interno. Este tipo de hipoacusia suele ser reversible con tratamiento médico o quirúrgico (A.D.A.M. Medical Encyclopedia, 2024)

- Hipoacusia Neurosensorial:

  Se produce debido a daños en las células ciliadas del oído interno o en las vías nerviosas que transmiten la información auditiva al cerebro. Los dispositivos como audífonos e implantes cocleares son las principales herramientas para mejorar la calidad de vida de

las personas con este tipo de pérdida auditiva (Organización Mundial de la Salud, 2026).

- Hipoacusia Mixta:

  Este tipo combina características de las hipoacusias conductiva y neurosensorial, afectando tanto la conducción como la percepción del sonido. (APADA-ASTURIAS, 2018).

- Hipoacusia Central:

  Es menos frecuente y ocurre debido a problemas en las vías auditivas centrales o en las áreas del cerebro responsables del procesamiento del sonido. Aunque el oído físico puede estar en buen estado, la interpretación de los estímulos sonoros está alterada.(Merck Manual Professional Edition, 2024)

2.1.1.2 Clasificación Según el Momento de Aparición

La clasificación de la discapacidad auditiva según el momento de aparición se centra en identificar si la pérdida auditiva ocurre antes o después de que una persona haya adquirido el lenguaje hablado.(Merck Manual Professional Edition, 2024)

- Hipoacusia Prelocutiva:

  Se presenta antes de que la persona haya adquirido el lenguaje hablado, generalmente en los primeros años de vida. Esta condición tiene un impacto significativo en el desarrollo lingüístico y cognitivo, ya que limita la exposición a los estímulos auditivos necesarios para estructurar el lenguaje. (Manual MSD, 2024)

- Hipoacusia Postlocutiva:

  Aparece después de que la persona ha adquirido habilidades básicas de lenguaje. Aunque el impacto en la comunicación es menor, puede generar dificultades para

comprender conversaciones en ambientes ruidosos o distinguir ciertos tonos de voz (A.D.A.M. Medical Encyclopedia, 2024)

2.1.1.3 Clasificación Según la Gravedad

La severidad de la pérdida auditiva se mide según el nivel de decibeles (dB) necesarios para percibir sonidos en el mejor oído (Organización Mundial de la Salud, 2026)

- Leve (20-40 dB): Dificultad para escuchar sonidos suaves.
- Moderada (41-70 dB): Problemas para comprender el habla normal.
- Severa (71-90 dB): Gran dificultad para entender el habla, incluso en ambientes tranquilos.
- Profunda (>90 dB): Dependencia total de herramientas visuales y dispositivos auditivos.

2.1.2 Impacto y Necesidades Educativas en la Infancia

La discapacidad auditiva tiene un impacto profundo en diversas áreas del desarrollo infantil, incidiendo directamente en los aspectos lingüísticos, sociales y educativos. Esta condición adquiere una relevancia crítica en el rango etario de 6 a 8 años, etapa en la que se consolidan habilidades fundamentales como la lectoescritura y la comunicación. (Sánchez Fuentes et al., 2019)

2.1.2.1 Desarrollo Lingüístico y Cognitivo

El desarrollo del lenguaje en niños con discapacidad auditiva está estrechamente relacionado con la capacidad de percibir sonidos y asociarlos con significados. En los casos de hipoacusia prelocutiva, la ausencia o disminución de estímulos sonoros limita la capacidad del infante para asociar sonidos con palabras y conceptos. Esto genera retrasos significativos en la adquisición

de vocabulario y estructuras gramaticales, comprometiendo la comprensión lectora y la producción oral(Organización Mundial de la Salud, 2026).

Según (Herrera Fernández et al., 2016) la integración de sistemas visuales como el lenguaje de señas puede ayudar a compensar estas limitaciones y facilitar la comprensión del lenguaje escrito.

2.1.2.2 Interacción Social y Barreras Comunicativas

Las limitaciones en la comunicación dificultan la participación en actividades grupales, lo que puede derivar en aislamiento social y afectar la autoestima del infante. Aunque los niños con discapacidad auditiva tienden a desarrollar estrategias visuales y kinestésicas para adaptarse a su entorno, muchas veces esto resulta insuficiente sin un ambiente inclusivo. Es fundamental promover actividades que fomenten la empatía y reduzcan las barreras de comunicación con las personas oyentes.(Sánchez Fuentes et al., 2019)

2.1.2.3 Educativo Rendimiento Académico y Estrategias Tecnológicas

En el ámbito educativo, los infantes enfrentan mayores desafíos debido a la dependencia tradicional del lenguaje hablado en el currículo. Los métodos de enseñanza convencionales (como el uso de tarjetas, repetición y dibujos) a menudo carecen de la interactividad necesaria para mantener la atención sostenida. Para mitigar estas dificultades y atender sus necesidades de procesamiento visual, es imprescindible incorporar recursos lúdicos y tecnologías interactivas. El uso de herramientas basadas en inteligencia artificial y aplicaciones visuales permite que el estudiante practique de manera autónoma, reciba retroalimentación inmediata y fortalezca su aprendizaje de forma motivadora y personalizada (Organización Mundial de la Salud, 2026).

2.2 Léxico Dactilológico y Sistemas de Lengua de Señas

El léxico dactilológico, comúnmente conocido como alfabeto manual o deletreo, constituye un sistema de representación viso-gestual en el cual cada letra del alfabeto escrito se asocia con una configuración manual específica (Moreno-Pérez et al., 2015) Es imperativo destacar que las lenguas de señas no son representaciones mímicas del lenguaje oral; poseen una estructura gramatical, sintáctica y un vocabulario tridimensional completamente independientes. Dentro de este complejo ecosistema lingüístico, el dactilológico actúa como un sistema complementario y un puente de transcodificación. Su uso se reserva estratégicamente para la articulación de nombres propios, neologismos, términos técnicos o vocablos que carecen de un lexema gestual equivalente en el idioma (Marín García, 2021)

En el ámbito de la educación infantil, la adquisición temprana del léxico dactilológico resulta un componente crítico para el desarrollo de la lectoescritura en aprendices con discapacidad auditiva. Investigaciones recientes en psicolingüística demuestran que el deletreo manual facilita la decodificación ortográfica, permitiendo a los infantes establecer correspondencias directas entre las configuraciones espaciales de sus manos y los grafemas del lenguaje escrito. Esta bimodalidad fonológica-visual acelera la alfabetización y reduce significativamente las brechas de comprensión lectora en edades tempranas(Gutiérrez Leguizamón et al., 2022)

2.2.1 Definición y origen del léxico dactilológico

El léxico dactilológico tiene su origen en la evolución de los sistemas de comunicación visual desarrollados por comunidades sordas. Estos sistemas surgieron como respuesta a la necesidad de establecer una comunicación efectiva en entornos donde el lenguaje oral no era accesible. Según (Gutiérrez Leguizamón et al., 2022), el léxico dactilológico no solo es una herramienta funcional, sino que también representa un puente entre el lenguaje de señas y el lenguaje escrito, promoviendo la alfabetización y el desarrollo lingüístico de las personas sordas.

2.2.2 ASL como referencia dactilológica y adaptaciones locales

En el ámbito de la visión artificial y el aprendizaje automático (*Machine Learning*), el alfabeto manual del *American Sign Language* (ASL) se ha consolidado como un estándar técnico de facto. La inmensa mayoría de arquitecturas de redes neuronales y repositorios de datos (*datasets*) de código abierto están pre-entrenados utilizando la morfología del ASL, debido a su influencia histórica y a la vasta disponibilidad de muestras estandarizadas (Gutiérrez Leguizamón et al., 2022). Sin embargo, la extrapolación directa de estos modelos foráneos a entornos educativos latinoamericanos representa un error arquitectónico y pedagógico severo, ya que ignora el contexto lingüístico y cultural de los usuarios finales (Quinta Lipe & Guerrero, 2025)

En el Ecuador, la comunidad sorda se comunica mediante la Lengua de Señas Ecuatoriana (LSEC). Si bien el alfabeto dactilológico de la LSEC comparte una raíz morfológica y topológica significativa con el ASL —lo que viabiliza el uso de técnicas computacionales de *Transfer Learning*—, presenta adaptaciones y adiciones estrictamente diseñadas para satisfacer las necesidades ortográficas de la lengua española y el entorno local (Ministerio de Educación del Ecuador, 2019)

2.3 Estrategias Lúdicas e Interactivas en el Aprendizaje Infantil

El diseño de interfaces tecnológicas orientadas a la infancia requiere trascender la mera presentación de información para convertirse en un ecosistema de participación activa. En el contexto del software educativo para estudiantes con discapacidad auditiva, la interactividad se define como el ciclo continuo de acción y respuesta entre el usuario y la máquina(Alotaibi, 2024). Durante la etapa de 6 a 8 años, el aprendizaje basado en juegos digitales (*Digital Game-Based Learning*) actúa como un catalizador cognitivo, transformando la repetición sistemática

de gestos —una tarea inherentemente monótona— en un proceso dinámico de descubrimiento (Behnamnia et al., 2023)

La integración de tecnologías de visión artificial en estas estrategias permite que el cuerpo del infante se convierta en el controlador del sistema. Esta bimodalidad, que conecta el movimiento físico (kinestésico) con la validación en pantalla, refuerza la memoria muscular de las configuraciones manuales y promueve una curva de aprendizaje autónoma, reduciendo la dependencia de la instrucción directa tradicional (Torres et al., 2021)

2.3.1 Diseño de Interfaces, Gamificación y Retroalimentación Visual

El diseño de juegos educativos orientados a infantes con discapacidad auditiva requiere interfaces intuitivas que estructuren la información mediante animaciones y gráficos claros, reduciendo la carga cognitiva derivada de la lectura de textos complejos (Zainuddin et al., 2020). Como se observa en la **Figura 3**, estructurar el contenido en distintos niveles interactivos permite al usuario acceder al conocimiento de manera progresiva y autónoma a través de dispositivos electrónicos.

Figura 3
Interfaz interactiva en el diseño de juegos educativos.



Nota. Adaptado de Drimlike (2023.), *Desarrollo de juegos didácticos y gamificación* (https://www.drimlike.com/es/gamificacion/desarrollo-de-juegos-didacticos-y.html).

Para sostener la motivación del infante frente al dispositivo, la arquitectura del *software* debe implementar técnicas de gamificación. Este concepto consiste en aplicar mecánicas propias del diseño de videojuegos —tales como la acumulación de puntajes y la obtención de

recompensas— en entornos de aprendizaje, incrementando el compromiso del estudiante (Zainuddin et al., 2020). En la **Figura 4** se puede ver cómo la gamificación se organiza para incentivar la práctica constante y el refuerzo positivo.

Figura 4
Gamificación



Nota: Adaptado de Net-Learning, 2023

En infantes sordos, cuyo canal primario de procesamiento es estrictamente visual, la gamificación debe estar unida a la retroalimentación visual (*visual feedback*). Cuando el algoritmo de Inteligencia Artificial procesa la postura de la mano, el sistema traduce inmediatamente ese cálculo en un estímulo gráfico (Chan et al., 2022). Al observar que la pantalla valida su seña (cambios de color o barras de progreso), el niño ajusta instintivamente sus movimientos hasta que el sistema lo reconoce, consolidando la precisión geométrica mediante un bucle de corrección autónoma.

2.3.2 Personalización y Adaptabilidad del Sistema

Las herramientas tecnológicas inclusivas no pueden ser estáticas; deben permitir la personalización del contenido para responder a las curvas de aprendizaje individuales de cada infante (Herzig & Allen, 2023). Esta personalización es un factor crítico en casos de discapacidad auditiva, ya que los estudiantes presentan distintos niveles de desarrollo motor y comprensión espacial.

La adaptabilidad del software implica la capacidad de ajustar el nivel de dificultad, modificar los tiempos de espera para la ejecución de un gesto, o calibrar los umbrales de exigencia del

algoritmo de reconocimiento (Chiotaki et al., 2023). Como se detalla en la **Figura 5**, la personalización en los videojuegos educativos no solo previene la frustración, sino que optimiza la experiencia de aprendizaje al asegurar que el desafío tecnológico sea acorde a la destreza actual del infante.

Figura 5
Beneficios de la personalización en los videojuegos para niños



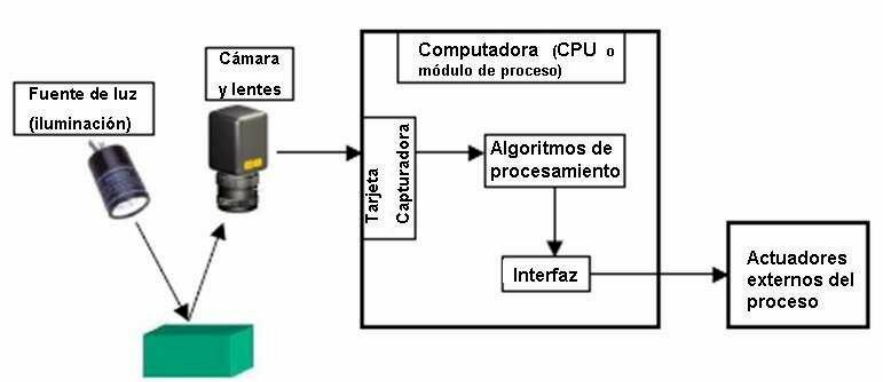
Nota. Adaptado de *Beneficios de la personalización en los videojuegos*, por Kokoro Kids, 2021

2.4 Inteligencia Artificial y Visión Artificial Aplicada

La visión artificial es una disciplina de la ingeniería que intenta replicar el proceso biológico de la vista, permitiendo a las máquinas extraer información útil de imágenes digitales para automatizar procesos o tomar decisiones (Szeliski, 2022). Cuando se analiza la literatura técnica sobre estos sistemas —ya sea para inspección industrial o para el reconocimiento interactivo de gestos— se observa una arquitectura física estandarizada.

Como se detalla en la **Figura 6**, el flujo de funcionamiento básico de un sistema de visión artificial se compone de los siguientes elementos secuenciales

Figura 6
Esquema de un sistema de visión artificial.



Nota. Adaptado de Duque, A. *Sistema de atención visual para la detección de puntos topológicos de referencia*.

- **Fuente de luz o iluminación:** Es el pilar físico del sistema. Su función es iluminar el entorno de manera uniforme para que la cámara pueda captar la imagen con claridad. Una iluminación óptima es fundamental para resaltar bordes, formas y texturas, evitando sombras que puedan confundir a los algoritmos (Oudah et al., 2020).
- **Objeto a inspeccionar:** Es el elemento físico que se somete a análisis. Mientras que en la industria suele ser una pieza fabricada, en sistemas de accesibilidad y *software* educativo, el "objeto" corresponde a las manos del usuario y sus configuraciones biométricas.(Rastgoo et al., 2021)
- **Cámara y lentes:** Constituyen el sistema óptico. La cámara es el sensor que captura la luz rebotada por el objeto, mientras que los lentes se encargan de enfocar la imagen y ajustar el campo de visión adecuado.(Rastgoo et al., 2021)
- **Tarjeta capturadora:** Es el puente de *hardware* que recibe la señal cruda proveniente de la cámara y la transforma (digitaliza) en información binaria o digital, dejándola lista para ser interpretada por la computadora.(Rastgoo et al., 2021)
- **Computadora o CPU:** Representa el núcleo lógico del sistema. Aquí es donde se procesa la imagen digitalizada utilizando algoritmos matemáticos y modelos de Inteligencia Artificial. En el reconocimiento de señas, esta unidad se encarga de

detectar la posición geométrica de la mano y clasificar el gesto realizado (Goodfellow et al., 2016c)

- **Interfaz:** Es el medio de comunicación que traduce y transmite el resultado del análisis (la predicción de la IA) hacia otros equipos, bases de datos o el propio usuario.
- **Actuadores externos del proceso:** Son los dispositivos periféricos que ejecutan una acción física o lógica en respuesta al resultado del análisis. En un entorno lúdico interactivo, los actuadores pueden ser la propia pantalla del sistema emitiendo una retroalimentación visual (cambio de nivel, puntaje) o un altavoz reproduciendo un estímulo positivo.(Berrezueta-Guzman et al., 2025)

2.4.1 Técnicas de extracción de características espaciales

Cuando se revisa la literatura sobre reconocimiento visual, existen dos enfoques principales para analizar una imagen. El primer método es el análisis basado en la apariencia o en píxeles. Aquí, se toma la fotografía completa y se introduce en redes neuronales muy pesadas (como las convolucionales o CNN). El algoritmo evalúa colores, texturas y sombras. Aunque este enfoque es muy preciso, resulta computacionalmente costoso porque procesa millones de píxeles por segundo (Oudah et al., 2020)

El segundo enfoque, que ha ganado terreno en sistemas de bajos recursos, es la estimación topológica o extracción de características geométricas (landmarks). En lugar de analizar toda la imagen, este método utiliza algoritmos previos para detectar únicamente el esqueleto o las articulaciones de la mano. Al aislar solo las puntas de los dedos y las falanges, la imagen pesada se descarta y se transforma en un vector ligero de coordenadas tridimensionales (X, Y, Z). Al trabajar con distancias relativas en lugar de píxeles, los sistemas logran una mayor velocidad de procesamiento y se vuelven inmunes a problemas comunes como la mala iluminación o el color de la piel (Sharma & Singh, 2021)

2.4.2 Algoritmos de clasificación para gestos estáticos

Cuando un usuario realiza el gesto de una letra o un número, la información espacial no cambia en el tiempo; todo el contexto necesario para entender la seña está contenido en un solo instante o *frame*. A esto se le conoce como un gesto estático.(Rastgoo et al., 2021)

Para procesar este tipo de información plana y numérica (como los vectores de coordenadas extraídos previamente), se utilizan redes neuronales densas (DNN) o perceptrones multicapa. En esta arquitectura, cada valor de entrada está conectado a varias capas ocultas de "neuronas" matemáticas. A medida que los datos cruzan estas capas, la red multiplica los valores por distintos pesos que aprendió durante su entrenamiento, intentando encontrar patrones geométricos invisibles. Al final del proceso, la última capa arroja una distribución de probabilidades para decidir, por ejemplo, si la postura de los dedos se parece en un 95% a la letra "A" o a la letra "E" (Goodfellow et al., 2016c) .La ventaja de usar redes densas sobre datos estructurados es que ocupan poquísimo espacio y dan respuestas casi instantáneas.

2.4.3 Algoritmos secuenciales para gestos dinámicos

La complejidad matemática explota cuando se intenta interpretar palabras completas o saludos. Estos son gestos "dinámicos", lo que significa que el significado no radica en una sola postura, sino en la trayectoria y dirección del movimiento a lo largo del tiempo.(Rastgoo et al., 2021)

Si le pasamos estos datos en movimiento a una red neuronal estática (como una DNN), el sistema fracasa por una razón simple: no tiene memoria. Para resolver este cuello de botella, la ingeniería informática propone arquitecturas diseñadas específicamente para secuencias temporales:

- **El enfoque recurrente:** Las Redes Neuronales Recurrentes (RNN) poseen lazos de retroalimentación interna. A diferencia de las redes tradicionales, el resultado del *frame* anterior afecta al *frame* actual.
- **La arquitectura LSTM (*Long Short-Term Memory*):** Es la variante más robusta para video. Incorpora "compuertas lógicas" que deciden qué información del inicio del movimiento debe retenerse y cuál debe olvidarse. Al analizar un bloque de imágenes (por ejemplo, 100 *frames* consecutivos), la red logra entender el contexto y la dirección temporal de la seña para clasificar palabras en movimiento (Rastgoo et al., 2021)

2.5 Hardware para sistemas de procesamiento visual

El desarrollo de un sistema de procesamiento visual no depende únicamente del algoritmo utilizado, sino también de la plataforma física donde se ejecuta. En aplicaciones que trabajan con imágenes o video, el hardware debe permitir la captura de información visual, el procesamiento de datos y la entrega de una respuesta al usuario en un tiempo adecuado. Por esta razón, al estudiar sistemas de reconocimiento de gestos o señas, es necesario considerar aspectos como la capacidad de cómputo, el consumo energético, el tamaño del dispositivo, la compatibilidad con cámaras y la posibilidad de funcionar sin depender completamente de servicios externos.(Amin et al., 2022)

En los últimos años, el procesamiento local ha tomado importancia en aplicaciones de inteligencia artificial, especialmente cuando se requiere reducir la latencia y evitar el envío constante de datos hacia la nube. Este enfoque resulta útil en sistemas educativos e interactivos, ya que la respuesta del dispositivo debe presentarse durante la práctica del usuario y no después de un procesamiento remoto. (Chen & Ran, 2019) señalan que el procesamiento en el borde permite acercar la ejecución de modelos de aprendizaje profundo al lugar donde se generan los datos, reduciendo la dependencia de la red y mejorando el tiempo de respuesta. De forma

similar, (Merenda et al., 2020) indican que los dispositivos IoT con aprendizaje automático en el borde permiten ejecutar tareas de inferencia en equipos compactos, aunque su desempeño depende de los recursos disponibles en el hardware.

2.5.1 Plataformas de procesamiento: móviles, microcontroladores y SBC

Las plataformas de procesamiento utilizadas en sistemas de visión artificial pueden variar según el tipo de aplicación y el nivel de complejidad del reconocimiento. En un extremo se encuentran los dispositivos móviles, como teléfonos inteligentes y tabletas, los cuales integran cámaras, sensores, pantalla y capacidad de procesamiento en un solo equipo. Estas características los convierten en herramientas accesibles para aplicaciones educativas o de asistencia. Sin embargo, su uso depende del sistema operativo, de la compatibilidad de las bibliotecas y de las restricciones propias del dispositivo.(Merenda et al., 2020)

Los microcontroladores representan otra alternativa dentro de los sistemas embebidos. Estos dispositivos se utilizan con frecuencia en proyectos de bajo consumo, control de sensores, comunicación inalámbrica y activación de actuadores. Su ventaja principal está en el costo reducido y en la eficiencia energética; no obstante, su capacidad de procesamiento suele ser limitada cuando se trabaja con video, imágenes de alta resolución o modelos de inteligencia artificial. Por esta razón, en sistemas de reconocimiento visual suelen emplearse como apoyo para tareas simples, pero no como unidad principal de procesamiento de imágenes (Merenda et al., 2020)

Las computadoras de placa única, conocidas como SBC por sus siglas en inglés, ofrecen una capacidad mayor que los microcontroladores, ya que pueden ejecutar sistemas operativos completos, bibliotecas de visión artificial y modelos de aprendizaje automático. En esta categoría se encuentran plataformas utilizadas en aplicaciones de visión artificial, robótica, educación e inteligencia artificial embebida. Estas placas permiten trabajar con cámaras,

pantallas y periféricos externos, además de ejecutar procesamiento local sin depender necesariamente de un servidor remoto. Según (Sze et al., 2017), la ejecución eficiente de modelos de inteligencia artificial en hardware limitado requiere considerar no solo la precisión del modelo, sino también la memoria, el consumo energético y la velocidad de inferencia.

Como se evidencia en la **Tabla *1***, cada plataforma presenta ventajas y limitaciones que deben analizarse de acuerdo con el tipo de aplicación. Mientras los dispositivos móviles ofrecen facilidad de transporte e integración de componentes, los microcontroladores destacan por su bajo consumo, aunque presentan restricciones para tareas de visión artificial. En cambio, las computadoras de placa única y las plataformas con aceleración gráfica ofrecen mejores condiciones para el procesamiento local de imágenes y modelos de inteligencia artificial.

Tabla 1
Comparación general de plataformas de procesamiento para visión artificial

Plataforma	Características generales	Ventajas	Limitaciones
Dispositivos móviles	Integran cámara, pantalla, sensores y sistema operativo.	Son accesibles y fáciles de transportar.	Dependencia del sistema operativo y restricciones de compatibilidad.
Microcontroladores	Equipos de bajo consumo usados para sensores y control.	Bajo costo y eficiencia energética.	Capacidad limitada para video e inteligencia artificial.
Computadoras de placa única	Ejecutan sistemas operativos, bibliotecas y modelos de IA.	Permiten procesamiento local y conexión con periféricos.	Requieren mayor consumo energético y control térmico.
Plataformas con aceleración gráfica	Incorporan GPU o unidades especializadas para IA.	Mejor desempeño en visión artificial e inferencia local.	Mayor costo y necesidad de configuración técnica.

Nota. Elaboración propia con base en Chen y Ran (2019), Merenda et al. (2020) y Sze et al. (2017).

2.5.2 Cámaras y sensores para captura de movimiento

La captura de movimiento es una etapa esencial en el reconocimiento de señas, debido a que la calidad de la información inicial influye en el procesamiento posterior. Para este tipo de sistemas se pueden utilizar cámaras RGB, cámaras con profundidad, sensores inerciales y sensores electromiográficos.

Las cámaras RGB son una alternativa frecuente porque permiten capturar imágenes o video sin colocar dispositivos sobre el cuerpo del usuario. Esta característica es importante en aplicaciones educativas con niños, ya que facilita una interacción más natural. En cambio, los sensores IMU y EMG pueden aportar información sobre movimiento o actividad muscular, pero requieren dispositivos vestibles, contacto físico o procesos de calibración. Como se muestra en la **Tabla *2***, las cámaras presentan una ventaja en comodidad de uso, mientras que otros sensores pueden aumentar la complejidad del sistema (Rastgoo et al., 2021)

Tabla 2
Tecnologías de captura utilizadas en reconocimiento de señas

Tecnología	Descripción	Limitación principal
Cámara RGB	Captura imágenes o video en color sin contacto físico.	Puede verse afectada por iluminación o encuadre.
Cámara RGB-D	Captura imagen y profundidad.	Mayor costo y complejidad de instalación.
Sensores IMU	Registran orientación, aceleración y movimiento.	Requieren dispositivos vestibles.

Sensores EMG	Registran señales musculares.	Requieren contacto con la piel y calibración.

Nota. Elaboración propia con base en Oudah et al. (2020), Rastgoo et al. (2021) y Amin et al. (2022).

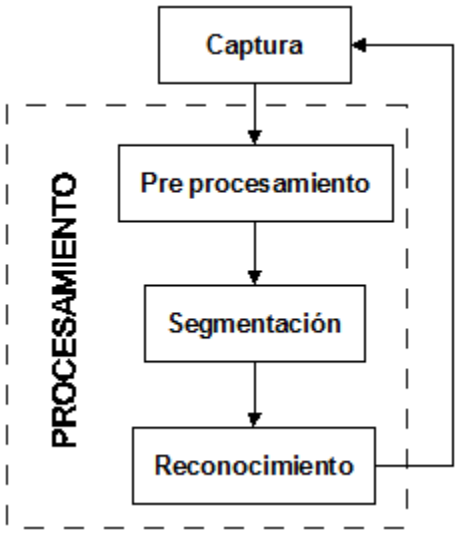
2.5.3 Procesamiento local en sistemas embebidos

El procesamiento local consiste en realizar la captura, análisis y reconocimiento dentro del propio dispositivo o cerca del lugar donde se generan los datos. Este enfoque reduce la necesidad de enviar imágenes o video a servidores externos, disminuye el uso de ancho de banda y permite una respuesta más rápida durante la interacción con el usuario (Chen & Ran, 2019)

En sistemas embebidos, este tipo de procesamiento también presenta retos, ya que el hardware tiene límites de memoria, temperatura, energía y capacidad de cómputo. Por esta razón, se recomienda utilizar modelos ligeros, reducir la cantidad de datos procesados y seleccionar características relevantes de la imagen, en lugar de trabajar siempre con información visual completa(Merenda et al., 2020)

Como se observa en la **Figura** *7*, un sistema de visión artificial sigue un flujo general que inicia con la captura de la imagen, continúa con el procesamiento y análisis de la información visual, y finaliza con una etapa de reconocimiento o respuesta del sistema. Aunque el ejemplo proviene de una aplicación industrial, el esquema permite comprender la lógica general que también se aplica en sistemas interactivos de reconocimiento visual.

Figura 7
Flujo general del procesamiento local en un sistema de reconocimiento visual



Nota. Adaptado de *Diseño e implementación de un sistema de reconocimiento y manipulación de frutas utilizando visión artificial y brazo robótico industrial*, por (A. J. Valencia Armijos & Idrovo Urgilés, 2016), citado en ResearchGate.

2.5.4 Software para reconocimiento visual

El software para reconocimiento visual permite unir la captura de imágenes con el procesamiento, la extracción de características y la presentación de resultados al usuario. En este tipo de sistemas se utilizan varias herramientas, ya que cada una cumple una función específica dentro del flujo de trabajo. En aplicaciones de reconocimiento de señas, estas herramientas deben permitir trabajar con video, detectar la mano, preparar los datos y ejecutar modelos de inteligencia artificial de forma local.(Á. J. Valencia Armijos & Idrovo Urgilés, 2016)

2.5.4.1 Herramientas de visión artificial

Las bibliotecas de visión artificial permiten trabajar con imágenes y video capturados por una cámara. OpenCV se utiliza para tareas como lectura de imágenes, captura de video, conversión de color y manejo de frames. Por otra parte, MediaPipe Hands permite detectar la mano y obtener puntos de referencia sobre la palma y los dedos, lo que facilita representar el gesto mediante datos numéricos en lugar de analizar toda la imagen directamente (Zhang et al., 2020)

Como se evidencia en la **Tabla *3***, cada herramienta cumple una función diferente dentro del sistema, desde la programación general hasta la detección de la mano y la ejecución de modelos.

Tabla 3
Herramientas de software utilizadas en sistemas de reconocimiento visual

Herramienta	Función principal	Aporte en el reconocimiento de señas
Python	Lenguaje de programación.	Integra las bibliotecas del sistema.
OpenCV	Procesamiento de imágenes y video.	Permite capturar y preparar frames.
MediaPipe	Detección de manos.	Obtiene puntos de referencia de la mano.
TensorFlow/Keras	Desarrollo de modelos de IA.	Permite entrenar modelos de clasificación.
TensorFlow Lite	Inferencia local.	Facilita ejecutar modelos en dispositivos con recursos limitados.

Nota. Elaboración propia con base en(Zhang et al., 2020), (Moreno-Pérez et al., 2015)

2.5.4.2 Frameworks de aprendizaje automático

Los frameworks de aprendizaje automático permiten construir, entrenar y ejecutar modelos de inteligencia artificial. TensorFlow es una herramienta utilizada para implementar modelos en diferentes plataformas, mientras que Keras facilita la creación de redes mediante capas. Para sistemas embebidos, TensorFlow Lite permite ejecutar modelos optimizados, reduciendo el consumo de memoria y procesamiento durante la inferencia (Sánchez Fuentes et al., 2019)

En el reconocimiento de señas, estas herramientas permiten transformar los datos procesados en una predicción comprensible para el usuario. Como se muestra en la **Tabla *4***, algunas herramientas se orientan al entrenamiento del modelo y otras a su ejecución local.

Tabla 4
Frameworks de inteligencia artificial y función general

Framework	Uso principal	Aplicación general
TensorFlow	Entrenamiento y ejecución de modelos.	Desarrollo de modelos de clasificación.
Keras	Creación de modelos por capas.	Construcción de redes neuronales.
TensorFlow Lite	Inferencia local.	Ejecución de modelos optimizados en dispositivos embebidos.

Nota. Elaboración propia con base en Abadi et al. (2016), Keras (2026) y David et al. (2021).

2.5.4.3 Aplicaciones locales, base de datos y reportes

Además del reconocimiento visual, el sistema necesita mostrar resultados y guardar la información generada durante las prácticas. Para esto se pueden usar herramientas como Flask, SQLite y archivos CSV. Flask permite crear aplicaciones web ligeras, SQLite funciona como una base de datos local que no requiere un servidor independiente, y el formato CSV permite exportar información tabular para revisarla en hojas de cálculo (Pallets Projects, 2026)

Como se observa en la **Tabla *5***, estas herramientas complementan el sistema porque permiten presentar información, guardar registros y generar reportes para el seguimiento docente.

Tabla 5
Herramientas para interfaz, almacenamiento y reportes

Herramienta	Función	Aporte en el sistema
Flask	Aplicación web local.	Permite crear interfaces y paneles de consulta.
SQLite	Base de datos local.	Guarda estudiantes, sesiones y resultados.
CSV	Formato de intercambio de datos.	Permite exportar reportes para revisión externa.

Nota. Elaboración propia con base en (Pallets Projects, 2026)

CAPÍTULO III. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

En este apartado se presenta el diseño del sistema propuesto para apoyar la práctica del léxico dactilológico en el entorno educativo. Este recurso busca ofrecer un medio interactivo que permita a los estudiantes ejercitar señas de forma visual y sencilla, complementando el trabajo que realiza la docente dentro del aula.

El prototipo está orientado al reconocimiento de letras, números y algunas palabras dinámicas mediante visión artificial e inteligencia artificial. Para ello, utiliza una cámara que captura los gestos realizados por el estudiante, una pantalla táctil para la interacción con las actividades y un panel de seguimiento que permite revisar los avances obtenidos durante la práctica. Además, el procesamiento se realiza de manera local, sin depender de conexión a internet, lo que facilita su uso en distintos espacios educativos. De esta forma, el diseño integra recursos tecnológicos y criterios pedagógicos para favorecer una práctica más directa, accesible y adaptada al ritmo de cada estudiante.

3.1 Metodología de desarrollo

Para el desarrollo del sistema se utilizó la metodología en cascada, ya que permite organizar el proyecto en etapas consecutivas y facilita el control de cada actividad antes de pasar a la siguiente. Este enfoque resultó adecuado porque el prototipo necesitaba una planificación ordenada: primero identificar las necesidades, luego diseñar la arquitectura, después implementar los módulos y finalmente comprobar su funcionamiento.

El proceso se dividió en cinco fases principales:

- **Fase de análisis:** En esta etapa se identificaron las necesidades del entorno educativo y las características que debía cumplir el sistema. Se consideró que debía funcionar sin

conexión a internet, ser fácil de usar para los estudiantes y permitir a la docente revisar el progreso de las actividades. También se tomaron en cuenta criterios pedagógicos relacionados con el aprendizaje del léxico dactilológico, a partir de la información obtenida mediante entrevistas con docentes especialistas en lengua de señas.

- **Fase de diseño:** Con base en los requerimientos definidos, se estableció la estructura general del proyecto. Se decidió utilizar una placa NVIDIA Jetson Nano como unidad principal de procesamiento, debido a que permite ejecutar modelos de visión artificial de forma local. Además, se diseñaron los módulos principales: captura de imagen mediante cámara, reconocimiento de señas, almacenamiento de resultados, interfaz táctil para el estudiante y dashboard web para la docente.
- **Fase de implementación:** En esta fase se construyó el prototipo físico dentro de una carcasa de madera con un diseño lúdico, pensado para hacerlo más atractivo y seguro durante su uso. También se desarrolló el software en Python, se incorporaron la cámara y la pantalla táctil, y se entrenaron los modelos de inteligencia artificial para reconocer letras, números y palabras dinámicas. Durante esta etapa se articularon los componentes de hardware y software para obtener una primera versión funcional.
- **Fase de verificación:** Después de la implementación, se realizaron pruebas para comprobar el reconocimiento de los gestos. Se revisó el flujo completo de funcionamiento, desde la captura de la seña realizada por el estudiante hasta la visualización de los resultados en el panel de la docente. También se verificó la generación de reportes en formato CSV, con el fin de facilitar el registro y análisis del avance de cada práctica.

3.2 Fase de análisis

En esta fase se analizaron las condiciones iniciales del proyecto, considerando el contexto educativo, los usuarios que utilizarán el prototipo y las funciones que debía cumplir la

herramienta para apoyar la práctica del léxico dactilológico. Para ello, se estudio el contexto del problema, se identifican los actores involucrados y se recopila información mediante diferentes técnicas de levantamiento.

Asimismo, se revisan trabajos relacionados que permiten conocer soluciones similares y aspectos técnicos relevantes para el proyecto. A partir de la información obtenida, se determinan las necesidades del sistema y se establecen los requerimientos funcionales, no funcionales y de arquitectura que servirán como base para las siguientes fases de desarrollo.

3.2.1 Situación Actual

El aprendizaje del léxico dactilológico en niños con discapacidad auditiva, cuando se basa en métodos de enseñanza tradicionales, enfrenta varias limitaciones. Martínez y Gómez (2021) señalan que, si bien existen recursos como libros y videos, estos suelen centrarse en la instrucción directa y presentan desventajas significativas en términos de interactividad y personalización.

Como consecuencia, los niños no tienen la oportunidad de practicar de manera autónoma ni de recibir la retroalimentación inmediata sobre sus errores, un aspecto que González y Villalobos (2019) destacan que las tecnologías digitales pueden solventar eficazmente.

Los métodos educativos tradicionales, al realizar la enseñanza de manera uniforme, no suelen considerar las diferencias cognitivas o el ritmo de aprendizaje individual de cada niño. Esta falta de adaptación puede llevar a que los estudiantes se sientan frustrados o desmotivados. De hecho, Mella-Norambuena et al. (2016) sostienen que las estrategias lúdicas, a menudo ausentes en estos métodos, son las que favorecen la motivación y la participación activa del estudiante.

La falta de herramientas interactivas y adaptativas evidencia la necesidad de desarrollar una solución tecnológica. Como lo indica la Organización Mundial de la Salud (OMS, 2021), la integración de tecnologías interactivas, como aplicaciones móviles, puede mitigar estas dificultades al proporcionar entornos de aprendizaje personalizados y accesibles.

3.2.2 Identificación de Actores Involucrados (Stakeholders)

La identificación de los actores involucrados permite establecer quiénes participan en el uso, desarrollo y seguimiento del prototipo. En este proyecto, los principales usuarios son los niños de 6 a 8 años, ya que ellos interactúan directamente con la herramienta durante las actividades de práctica del léxico dactilológico. También se considera a la docente, quien cumple un rol importante al orientar el proceso de aprendizaje y revisar los resultados obtenidos por los estudiantes.

Para el desarrollo del dispositivo lúdico, los stakeholders se han clasificado según su nivel de interacción con el sistema, tal como se puede ver en la **Tabla *6*** a continuación:

Tabla 6.
Descripción de los Stakeholders involucrados

Lista de Stakeholders	
Usuarios Directos	Niños de 6 a 8 años y Docentes.
Director de Tesis	MSc. Jaime Michilena
Asesor de Tesis	MSc. Fabián Cuzme
Desarrolladora	Srta. Alfonsina Chamorro.

Nota: Elaboración propia, 2026.

3.2.3 Metodología y Técnicas de Recolección

Para obtener información útil para el diseño del prototipo, se aplican dos técnicas de recolección: una entrevista semiestructurada y una revisión de trabajos relacionados. La

entrevista permite conocer, desde la experiencia de una docente especialista, las dificultades que se presentan durante la enseñanza de señas y las características que debe tener una herramienta de apoyo dirigida a niños de 6 a 8 años.

Por otra parte, la revisión de trabajos relacionados sirve para identificar soluciones similares y criterios técnicos vinculados con visión artificial, aprendizaje automático, interfaces interactivas y recursos lúdicos. De esta manera, ambas técnicas aportan a la definición de las necesidades del sistema, considerando tanto el enfoque pedagógico como los aspectos tecnológicos del prototipo.

3.2.3.1 Entrevista Semiestructurada

Con el propósito de identificar las necesidades y requerimientos asociados al sistema propuesto, se realizó una entrevista semiestructurada a una docente especialista. La selección de la participante se basó en su experiencia directa en los procesos de enseñanza dirigidos al público objetivo de este proyecto.

La entrevista estuvo conformada por preguntas abiertas orientadas a identificar las principales dificultades existentes en el aula, las necesidades de los usuarios y las características que debería incorporar la solución tecnológica. Las interrogantes planteadas fueron:

1. ¿Qué tipo de recursos didácticos o estrategias pedagógicas utiliza actualmente para enseñar el abecedario y las señas básicas a niños de entre 6 y 8 años?
2. Desde su experiencia, ¿cuáles son las principales dificultades o desafíos que presentan los niños de esta edad al momento de coordinar y memorizar las posturas exactas de las manos?
3. Al trabajar con un grupo de niños simultáneamente, ¿qué tan complejo resulta evaluar y corregir de manera individual la ejecución precisa de cada seña?

4. ¿Considera que los materiales tradicionales (tarjetas, dibujos, repetición) logran mantener la atención sostenida de los niños en estas edades, o percibe que hace falta mayor interactividad?
5. Considerando el avance de la tecnología, ¿cree que hace falta implementar herramientas digitales interactivas enfocadas específicamente en la lengua de señas dentro del contexto educativo ecuatoriano?
6. ¿Qué características indispensables cree usted que debería tener una herramienta tecnológica para que le sea realmente útil tanto a usted como a los niños?
7. ¿Cree que sería beneficioso para el proceso cognitivo del niño interactuar con un sistema que le indique de forma visual, amigable e inmediata si está realizando la seña de manera correcta?

La información recolectada mediante este instrumento resulta fundamental para asegurar que el diseño interactivo y lúdico responda a una necesidad real del entorno educativo. La evidencia física del cuestionario, junto con las respuestas manuscritas originales emitidas por la especialista, se encuentran documentadas en el **Anexo A:** Formato Entrevista semiestructurada a usuario especialista.

3.2.3.2 Revisión de Trabajos Relacionados

Esta técnica se enfoca en la identificación de los requerimientos técnicos necesarios para el diseño e implementación del modelo de visión artificial y el hardware. Mediante la recopilación de fichas bibliográficas, se analizó el estado del arte de sistemas similares para establecer las mejores prácticas tecnológicas, tal como se estructura en la Tabla 7.

Tabla 7.
Trabajos relacionados analizados para la identificación de requerimientos

Código	Autor(es)	Título de la investigación	Año	Fuente
T1	Quinta, D., & Guerrero, P.	Traducción inteligente de lenguaje de señas mediante aprendizaje automático y visión artificial	2025	Revista Criterio / Universidad Pública de El Alto

T2	Plazas, J., Gutiérrez, J., Suárez, M., & González, J.	Reconocimiento de lengua de señas colombiana mediante redes neuronales convolucionales y captura de movimiento	2022	Revista Tecnura / Universidad Pedagógica y Tecnológica de Colombia
T3	Hinojosa, D.	Traductor de lenguaje de señas para personas con discapacidad auditiva basado en visión artificial	2024	Universidad Técnica del Norte
T4	Caicedo, J.	Juguete electrónico lúdico como herramienta educativa para la adquisición de la lectoescritura en niños de 5 a 8 años con NEE	2019	Universidad Técnica del Norte
T5	Agudo, M.	Desarrollo de un juego interactivo para niños con necesidades educativas especiales	2016	Universidad Politécnica de Madrid

Nota: Elaboración propia, 2026.

Los trabajos relacionados fueron seleccionados considerando dos enfoques principales: el componente técnico asociado al reconocimiento de señas mediante visión artificial y aprendizaje automático, y el componente educativo relacionado con herramientas lúdicas e interactivas dirigidas a niños. El detalle de los trabajos analizados se presenta en el **Anexo B:** Matriz de revisión de trabajos relacionados

3.2.4 Necesidades del Sistema

Una vez aplicadas las técnicas de levantamiento de información, se procede al análisis de los datos obtenidos con el propósito de identificar las principales necesidades que debe cubrir el sistema propuesto. Para ello, se consideran dos fuentes principales: la entrevista semiestructurada realizada a la docente especialista y la revisión de trabajos relacionados.

La entrevista permite conocer las dificultades presentes en el proceso de enseñanza y aprendizaje desde una perspectiva pedagógica y práctica. Por otra parte, la revisión de trabajos

relacionados permite identificar criterios técnicos, herramientas, metodologías y características utilizadas en soluciones similares. A partir de ambas fuentes se establecen los hallazgos más relevantes y se determinan las necesidades que servirán como base para la posterior definición de requerimientos.

3.2.4.1 Resultados de la entrevista semiestructurada

La entrevista semiestructurada permitió identificar aspectos relacionados con la metodología de enseñanza, las dificultades de aprendizaje de los niños y las características esperadas en una herramienta tecnológica de apoyo. Las preguntas estuvieron orientadas a conocer los recursos didácticos utilizados, las limitaciones de los métodos tradicionales, la complejidad de realizar correcciones individuales y la utilidad de una herramienta interactiva con retroalimentación inmediata.

Los resultados evidencian que el proceso de aprendizaje requiere recursos visuales, actividades dinámicas y mecanismos que permitan al niño recibir una orientación clara durante la práctica. Además, se identificó la necesidad de que la herramienta sea sencilla de utilizar, amigable para el usuario infantil y capaz de apoyar al docente en el seguimiento de la ejecución de las señas, tal como se puede observar en la Tabla 8.

Tabla 8.
Resultados de la entrevista semiestructurada

Pregunta	Aspecto analizado	Hallazgo identificado	Necesidad relacionada
P1	Recursos didácticos utilizados	Se identificó el uso de recursos tradicionales como imágenes, tarjetas, repetición y apoyo visual.	Incorporar recursos visuales e interactivos que refuercen el aprendizaje.
P2	Dificultades de aprendizaje	Los niños pueden presentar dificultad para memorizar y coordinar correctamente las posturas de las manos.	Brindar apoyo visual y guía durante la ejecución de las señas.

P3	Corrección individual	La evaluación simultánea de varios niños dificulta la corrección personalizada de cada seña.	Proporcionar retroalimentación individual sobre la ejecución del gesto.
P4	Limitaciones de métodos tradicionales	Los materiales tradicionales pueden no mantener la atención de los niños durante toda la actividad.	Incorporar actividades lúdicas e interactivas.
P5	Uso de herramientas digitales	Se reconoce la utilidad de integrar herramientas tecnológicas en el proceso educativo.	Diseñar una solución tecnológica orientada al aprendizaje de señas.
P6	Características esperadas	La herramienta debe ser fácil de usar, amigable y adecuada al contexto educativo.	Diseñar una interfaz sencilla, visual y comprensible para niños de 6 a 8 años.
P7	Retroalimentación inmediata	Se considera beneficioso que el niño reciba una respuesta visual inmediata sobre su desempeño.	Implementar retroalimentación inmediata ante aciertos y errores.

Nota: Elaboración propia, 2026.

A partir de estos resultados, se determina que el sistema no debe limitarse únicamente al reconocimiento de gestos, sino que también debe responder a necesidades pedagógicas. Por esta razón, la solución propuesta debe integrar elementos de interacción visual, retroalimentación inmediata y actividades amigables que favorezcan la participación del niño durante el aprendizaje.

3.2.4.2 Resultados de la revisión de trabajos relacionados

La revisión de trabajos relacionados permitió identificar antecedentes técnicos y educativos vinculados con el reconocimiento de señas, el uso de visión artificial y el desarrollo de herramientas lúdicas para niños. Los trabajos analizados fueron clasificados en dos enfoques principales: el componente técnico, relacionado con aprendizaje automático, visión artificial, redes neuronales, captura de gestos e inferencia en tiempo real; y el componente educativo,

relacionado con el diseño de actividades interactivas, recursos visuales, retroalimentación y estrategias lúdicas, así como se puede observar en la *Tabla 9.*

Los trabajos de Quinta Lipe y Guerrero (2025), Plazas López et al. (2022) e Hinojosa González (2024) aportan criterios relevantes para el componente de visión artificial, debido a que abordan el reconocimiento de señas mediante modelos de aprendizaje automático, captura de imágenes, procesamiento de gestos y uso de herramientas como OpenCV, MediaPipe, TensorFlow, YOLOv8 y plataformas de procesamiento local. Por otro lado, los trabajos de Caicedo Sandoval (2019) y Agudo Ballesteros (2016) aportan criterios pedagógicos e interactivos, ya que presentan herramientas educativas orientadas a niños con necesidades educativas especiales mediante actividades lúdicas, audio, recursos visuales, repetición, modos de juego y retroalimentación.

Tabla 9.
Resultados de la revisión de trabajos relacionados

Código	Trabajo relacionado	Enfoque principal	Hallazgo identificado	Necesidad relacionada
T1	Quinta Lipe y Guerrero (2025)	Aprendizaje automático y visión artificial	Uso de inteligencia artificial y procesamiento visual para traducir señas en entornos con recursos limitados.	Incorporar visión artificial y procesamiento eficiente de gestos.
T2	Plazas López et al. (2022)	Redes neuronales convolucionales y captura en tiempo real	Aplicación de CNN, captura de movimiento, OpenCV, TensorFlow y MediaPipe para reconocer señas.	Procesar gestos en tiempo real mediante herramientas de visión artificial.
T3	Hinojosa González (2024)	Traductor de señas con visión artificial	Uso de Jetson Nano, cámara, OpenCV, MediaPipe y YOLOv8 para inferencia local en tiempo real.	Definir una arquitectura tecnológica capaz de ejecutar reconocimiento local.

T4	Caicedo Sandoval (2019)	Herramienta educativa lúdica	Uso de actividades lúdicas, recursos visuales y auditivos para apoyar el aprendizaje en niños.	Incorporar actividades interactivas y recursos visuales adaptados a niños.
T5	Agudo Ballesteros (2016)	Juego interactivo para niños con NEE	Implementación de modos de juego, audio, puntuación, repetición y validación funcional.	Integrar retroalimentación, dinámica de juego y repetición de actividades.

Nota: Elaboración propia, 2026.

El análisis de estos trabajos permitió reconocer que una solución tecnológica orientada al aprendizaje de señas debe considerar tanto el desempeño técnico del sistema como la experiencia del usuario infantil. Por lo tanto, el sistema debe ser capaz de reconocer gestos de manera automática, entregar una respuesta inmediata y mantener una interacción clara, visual y motivadora.

3.2.4.3 Necesidades identificadas

Con base en los resultados obtenidos de la entrevista semiestructurada y de la revisión de trabajos relacionados, se establecieron las necesidades principales del sistema, como se puede ver en la Tabla 10. Estas necesidades representan las condiciones que la solución propuesta debe atender para responder adecuadamente al problema identificado.

Tabla 10.
Necesidades identificadas del sistema

Código	Necesidad identificada	Fuente de identificación
N-01	Incorporar recursos visuales que apoyen la memorización y ejecución de las señas.	Entrevista semiestructurada, T4
N-02	Proporcionar retroalimentación inmediata al usuario sobre la ejecución correcta o incorrecta de la seña.	Entrevista semiestructurada, T5
N-03	Permitir la evaluación individual de cada intento realizado por el niño.	Entrevista semiestructurada
N-04	Incluir actividades interactivas y lúdicas que favorezcan la atención y participación del usuario.	Entrevista semiestructurada, T4, T5

N-05	Reconocer automáticamente los gestos mediante técnicas de visión artificial.	T1, T2, T3
N-06	Procesar los gestos capturados en tiempo real o en un tiempo adecuado para la interacción educativa.	T2, T3
N-07	Utilizar una arquitectura tecnológica adecuada para el procesamiento de imágenes y reconocimiento de gestos.	T1, T2, T3
N-08	Diseñar una interfaz sencilla, amigable y comprensible para niños de 6 a 8 años.	Entrevista semiestructurada, T4, T5
N-09	Facilitar el uso del sistema en un entorno educativo, considerando la participación del docente como apoyo en el proceso de aprendizaje.	Entrevista semiestructurada, T4
N-10	Permitir el registro o visualización del desempeño del usuario para apoyar el seguimiento del aprendizaje.	Entrevista semiestructurada, T4, T5

Nota: Elaboración propia, 2026.

Las necesidades identificadas permiten establecer una base clara para la definición de los requerimientos del sistema. Estas necesidades serán transformadas posteriormente en requerimientos de stakeholders, requerimientos funcionales, requerimientos no funcionales y requerimientos de arquitectura, siguiendo una estructura trazable y coherente con el proceso de análisis realizado.

3.2.5 Requerimientos del Sistema

Para la especificación de los requerimientos del sistema se toma como referencia la norma ISO/IEC/IEEE 29148, la cual establece lineamientos para la definición, organización y documentación de requisitos. Su aplicación permite estructurar los requerimientos de manera clara, verificable y trazable, facilitando su relación con las necesidades identificadas durante el levantamiento de información.

3.2.5.1 Nomenclatura de requerimientos

En esta sección se establece la nomenclatura utilizada para la identificación y organización de los requerimientos del sistema. Esta clasificación permite diferenciar los requerimientos según su propósito dentro del proyecto, facilitando su análisis, seguimiento y posterior validación.

Los requerimientos se clasifican en necesidades identificadas, requerimientos de stakeholders, requerimientos funcionales, requerimientos no funcionales, requerimientos de arquitectura y requerimientos del dataset. Esta organización permite relacionar las necesidades obtenidas durante el levantamiento de información con las características que deberá cumplir el sistema propuesto.

Como se muestra en la Tabla *11*, se emplean abreviaturas específicas para cada tipo de requerimiento, con el fin de facilitar su referencia a lo largo del documento.

Tabla 11.
Abreviaturas de Requerimientos

Requerimientos	Abreviatura
Necesidad identificada	N
Requerimiento de stakeholder	StSR
Requerimiento funcional	RF
Requerimiento no funcional	RNF
Requerimiento de arquitectura	RArq
Requerimiento del dataset	RD

Nota: Elaboración propia, 2026.

De igual manera, se establecieron niveles de prioridad para cada requerimiento , tal como se muestra en la **Tabla *12***. Esta clasificación toma en cuenta la importancia de cada requerimiento dentro del diseño y la implementación del sistema. De esta forma, se identifican aquellos

elementos esenciales para el correcto funcionamiento de la solución, así como los que corresponden a mejoras adicionales que complementan el sistema.

Tabla 12
Prioridades de los Requerimientos

Prioridad	Descripción
Alta	Requerimientos esenciales para el funcionamiento del sistema. Deben cumplirse para garantizar la funcionalidad principal de la solución propuesta
Media	Requerimientos importantes para mejorar el funcionamiento del sistema, pero que no comprometen su operación básica.
Baja	Requerimientos complementarios que aportan mejoras a la experiencia del usuario o al sistema, pero que no son indispensables en la fase inicial.

Nota: Elaboración propia, 2026.

3.2.5.2 Requerimientos de stakeholders

Los requerimientos de stakeholders representan las necesidades y expectativas de las personas involucradas en el uso del sistema. Para su definición se consideraron los usuarios finales, el personal docente y el desarrollador del sistema, como se puede evidenciar en la *Tabla 13.* Estos requerimientos permiten establecer una relación directa entre las necesidades identificadas y las funcionalidades que deberá cumplir la herramienta propuesta.

Tabla 13.
Requerimientos de stakeholders

REQUERIMIENTOS DEL STAKEHOLDERS (StSR)					
Código	**Requerimiento**	**Necesidad asociada**	**PRIORIDAD**		
			Alta	**Media**	**Baja**
Requerimientos orientados al aprendizaje					

StSR-01	El sistema debe apoyar el aprendizaje de señas mediante actividades visuales	N-01, N-04, N-08	X	
StSR-02	El sistema debe presentar al niño la seña que debe practicar de forma clara y comprensible.	N-01, N-08	X	
StSR-03	El sistema debe permitir la práctica de señas estáticas, como letras del abecedario y números.	N-05, N-06	X	
StSR-04	El sistema debe permitir la práctica de señas dinámicas asociadas a palabras seleccionadas.	N-05, N-06	X	
StSR-05	El sistema debe brindar una respuesta visual cuando el niño realice una seña de forma correcta o incorrecta.	N-02	X	
StSR-06	El sistema debe ofrecer apoyo visual cuando el niño presente dificultad durante la ejecución de una seña.	N-01, N-02		X
StSR-07	El sistema debe incorporar elementos lúdicos que favorezcan la motivación y participación del niño durante la sesión.	N-04	X	
Requerimientos de uso en el entorno educativo				
StSR-08	El sistema debe funcionar como apoyo al docente durante el proceso de enseñanza-aprendizaje de lengua de señas.	N-09	X	
StSR-09	El sistema debe permitir que el docente supervise el desarrollo de las actividades realizadas por el niño.	N-09, N-10	X	
StSR-10	El sistema debe ser fácil de utilizar dentro del aula, sin requerir conocimientos técnicos avanzados por parte del docente.	N-08, N-09	X	
StSR-11	El sistema debe adaptarse a las condiciones del entorno educativo donde se realicen las sesiones de práctica.	N-07, N-09		X

StSR-12	El sistema debe contar con un medio de captura que permita observar los gestos realizados por el usuario durante la actividad.	N-05, N-07	X		
StSR-13	El sistema debe operar de manera local para facilitar su uso en espacios donde la conexión a internet pueda ser limitada.	N-07, N-09	X		
StSR-14	El docente debe seleccionar la categoría de señas específica antes de iniciar la actividad.	N-09, N-10			X
Requerimientos de seguimiento y validación					
StSR-15	El sistema debe registrar los resultados obtenidos por el niño durante cada actividad realizada.	N-10	X		

Nota: Elaboración propia, 2026.

3.2.5.3 Requerimientos funcionales

Los requerimientos funcionales describen las acciones principales que debe realizar el sistema para cumplir con las necesidades identificadas durante el levantamiento de información. En el caso de la herramienta propuesta, estos requerimientos se relacionan con el registro de usuarios, la captura del gesto, el reconocimiento de señas, la retroalimentación visual, el seguimiento del desempeño y la generación de resultados para el docente.

Como se muestra en la *Tabla 14.* los requerimientos funcionales se organizaron por bloques con el fin de facilitar su comprensión. Estos bloques corresponden a la gestión de usuarios y sesiones, captura y procesamiento de gestos, interacción con el niño, retroalimentación visual y seguimiento de resultados.

Tabla 14.
Requerimientos funcionales

REQUERIMIENTOS FUNCIONALES (RF)					
Código	**Requerimiento funcional**	**Necesidad asociada**	**PRIORIDAD**		
			Alta	**Media**	**Baja**
Gestión de usuarios y sesiones					

RF-01	El sistema debe permitir registrar o seleccionar al niño que realizará la actividad de práctica.	StSR-14, StSR-15	X		
RF-02	El sistema debe permitir iniciar una sesión de práctica asociada a un niño y a una actividad específica.	StSR-08, StSR-14	X		
RF-03	El sistema debe permitir que el docente seleccione la categoría de señas a practicar antes de iniciar la actividad, considerando abecedario, números o palabras.	StSR-14			X
Captura y procesamiento de gestos					
RF-04	El sistema debe capturar de forma continua el flujo de video del usuario mediante una cámara para su posterior procesamiento.	StSR-12	X		
RF-05	El sistema debe detectar la mano del usuario durante la actividad de práctica.	StSR-12	X		
RF-06	El sistema debe extraer los puntos característicos de la mano para utilizarlos como entrada del modelo de reconocimiento.	StSR-12	X		
RF-07	El sistema debe reconocer señas estáticas correspondientes al abecedario dactilológico.	StSR-03	X		
RF-08	El sistema debe reconocer señas estáticas correspondientes a números.	StSR-03	X		
RF-09	El sistema debe reconocer señas dinámicas correspondientes a palabras seleccionadas.	StSR-04		X	
RF-10	El sistema debe comparar la seña detectada con la seña solicitada en la actividad.	StSR-05	X		
Interacción y retroalimentación					
RF-11	El sistema debe mostrar en la interfaz la seña que el niño debe realizar durante la actividad.	StSR-02	X		
RF-12	El sistema debe mostrar una respuesta visual cuando la seña realizada sea reconocida correctamente.	StSR-05	X		
RF-13	El sistema debe mostrar apoyo visual guiado cuando la seña objetivo no sea detectada correctamente después de un tiempo de espera definido.	StSR-06	X		

RF-14	El sistema debe incorporar elementos lúdicos durante la práctica, como mensajes de refuerzo, avances o indicadores visuales.	StSR-07		X	
Panel docente y seguimiento					
RF-15	El sistema debe permitir que el docente supervise la actividad desde un dispositivo.	StSR-09, StSR-10	X		
RF-16	El sistema debe registrar el resultado final realizado por el niño durante la actividad.	StSR-14	X		
RF-17	El sistema debe almacenar información de la sesión	StSR-15	X		
RF-18	El sistema debe permitir consultar los resultados registrados durante las sesiones de práctica.	StSR-15		X	
RF-19	El sistema debe permitir descargar los resultados	StSR-15		X	
RF-20	El sistema debe generar datos que permitan identificar aciertos, saltos, total de actividades y porcentaje de aciertos por seña..	StSR-15	X		
RF-21	El sistema debe permitir al docente consultar un resumen de los resultados por sesión, categoría y seña practicada	StSR-15			X

Nota: Elaboración propia, 2026.

Los requerimientos funcionales presentados en la *Tabla 14.* permiten establecer las funciones que debe cumplir el sistema durante su operación. De esta manera, se asegura que la herramienta no solo realice el reconocimiento de señas, sino que también apoye el proceso educativo mediante retroalimentación visual, actividades interactivas y registro de resultados para el seguimiento del aprendizaje

3.2.5.4 Requerimientos no funcionales

Los requerimientos no funcionales establecen las condiciones de calidad que debe cumplir el sistema durante su funcionamiento. A diferencia de los requerimientos funcionales, estos no

describen acciones específicas, sino características relacionadas con el rendimiento, usabilidad, estabilidad, almacenamiento, accesibilidad y mantenimiento de la herramienta.

En el caso del sistema propuesto, estos requerimientos son importantes porque la herramienta será utilizada por niños de 6 a 8 años dentro de un entorno educativo. Por esta razón, el sistema debe ofrecer una interacción fluida, una interfaz comprensible, tiempos de respuesta adecuados y un funcionamiento estable durante las sesiones de práctica. Como se muestra en la *Tabla 15*, los requerimientos no funcionales se organizan considerando los aspectos de usabilidad, rendimiento, operación local, almacenamiento y mantenibilidad.

Tabla 15
Requerimientos no funcionales

Requerimientos no funcionales (RNF)					
Código	**Requerimiento no funcional**	**Necesidad asociada**	**PRIORIDAD**		
			Alta	**Media**	**Baja**
Usabilidad e interacción					
RNF-01	La interfaz debe ser sencilla y visual para facilitar el uso por parte de los niños.	StSR-01, StSR-02	X		
RNF-02	Los recursos gráficos deben ser claros y adecuados para la edad del usuario.	StSR-02, StSR-06	X		
RNF-03	La herramienta debe mostrar mensajes o señales visuales comprensibles durante la práctica.	StSR-05, RF-12, RF-13	X		
RNF-04	El sistema debe ser fácil de manipular por el docente durante la actividad.	StSR-10, RF-15	X		
Requerimientos de rendimiento					
RNF-05	El sistema debe mantener una interacción fluida durante la captura y reconocimiento de señas.	StSR-05, RF-04, RF-05	X		

RNF-06	El sistema debe evitar interrupciones durante el desarrollo de las actividades principales.	StSR-08, StSR-11	X		
RNF-07	El sistema debe utilizar adecuadamente los recursos de la plataforma donde sea implementado.	StSR-13, RF-04, RF-10		X	
Requerimientos de operación					
RNF-08	El sistema debe funcionar de manera local para realizar las actividades sin depender de internet.	StSR-13	X		
RNF-09	El sistema debe adaptarse a las condiciones básicas del entorno educativo de uso.	StSR-11, StSR-12		X	
RNF-10	El sistema debe ser compatible con el uso de cámara, pantalla y red local para el desarrollo de la actividad.	StSR-11, StSR-12, RF-15		X	
Requerimientos de información					
RNF-11	La información generada durante las actividades debe almacenarse de forma estructurada.	StSR-14, StSR-15, RF-16, RF-17	X		
RNF-12	Los resultados deben presentarse de forma comprensible para el docente.	StSR-15, RF-18, RF-20		X	
RNF-13	Los datos registrados deben conservarse localmente dentro del sistema.	StSR-13, StSR-14, RF-17		X	
Requerimientos de mantenimiento					
RNF-14	El sistema debe permitir la actualización de recursos, actividades o modelos de reconocimiento.	StSR-11		X	
RNF-15	La estructura del sistema debe facilitar mejoras futuras en la interfaz o actividades educativas.	StSR-11			X

Nota: Elaboración propia, 2026

Los requerimientos no funcionales presentados en la *Tabla 15* permiten establecer condiciones generales para que el sistema funcione de manera adecuada en el entorno educativo. Estos aspectos complementan los requerimientos funcionales, ya que no describen acciones específicas del sistema, sino las características de calidad esperadas durante su uso. Los requerimientos RNF-05, RNF-06 y RNF-07 serán revisados durante las pruebas del sistema integrado. Para ello, se observará la continuidad de la captura y el reconocimiento, además de registrar los FPS procesados, la temperatura y el uso de memoria durante la ejecución. Estas mediciones permitirán describir el comportamiento del prototipo en la Jetson Nano bajo las condiciones del escenario de pruebas.

3.2.5.5 Requerimientos de arquitectura

Los requerimientos de arquitectura describen los componentes tecnológicos necesarios para el funcionamiento del sistema. Estos requerimientos permiten definir, de manera general, la estructura de hardware, software, procesamiento, almacenamiento y comunicación que deberá integrar la herramienta propuesta.

En la *Tabla 16* se presentan los requerimientos de arquitectura establecidos para el sistema, considerando que la solución debe realizar reconocimiento de señas mediante visión artificial, operar de forma local y permitir la interacción entre el niño, el docente y la herramienta.

Tabla 16
Requerimientos de arquitectura

REQUERIMIENTOS DE ARQUITECTURA (RArq)					
Código	**Requerimiento**	**Necesidad asociada**	**PRIORIDAD**		
			Alta	**Media**	**Baja**
Requerimientos de hardware					
RArq-01	El sistema debe contar con una plataforma de procesamiento local para ejecutar la herramienta.	StSR-13, RNF-08	X		

RArq-02	El sistema debe integrar una cámara para la captura de los gestos realizados por el usuario.	RF-04, StSR-12	X	
RArq-03	El sistema debe contar con una pantalla o medio de visualización para presentar la actividad y la retroalimentación.	RF-11, RF-12, RNF-01	X	
RArq-04	El sistema debe disponer de una fuente de alimentación adecuada para el funcionamiento de sus componentes.	StSR-11	X	
RArq-05	El sistema debe permitir la conexión de los dispositivos necesarios para la captura, visualización y control de la herramienta.	RF-04, RF-15		X
Requerimientos de software				
RArq-06	El sistema debe utilizar un entorno de programación compatible con el procesamiento de imágenes y modelos de reconocimiento.	RF-05, RF-06	X	
RArq-07	El sistema debe incluir librerías de visión artificial para la detección y análisis de la mano del usuario.	RF-05, RF-06	X	
RArq-08	El sistema debe integrar modelos de reconocimiento para señas estáticas y dinámicas.	RF-07, RF-08, RF-09	X	
RArq-09	El sistema debe contar con una interfaz gráfica para mostrar instrucciones, recursos visuales y resultados de la actividad.	RF-11, RF-12, RF-13	X	
RArq-10	El sistema debe contar con un servidor local que permita el acceso al panel del docente dentro de la red local.	RF-15, RNF-08	X	
Requerimientos de procesamiento				
RArq-11	El sistema debe procesar las imágenes capturadas para obtener información útil de la mano del usuario.	RF-05, RF-06	X	
RArq-12	El sistema debe comparar la seña reconocida con la seña solicitada durante la actividad.	RF-10	X	

RArq-13	El sistema debe permitir el uso de modelos optimizados para facilitar la ejecución local del reconocimiento.	RNF-05, RNF-07	X	
Requerimientos de almacenamiento y resultados				
RArq-14	El sistema debe contar con una base de datos local para almacenar usuarios, sesiones, intentos y resultados.	RF-16, RF-17, RNF-11	X	
RArq-15	El sistema debe permitir la consulta de la información registrada durante las sesiones de práctica.	RF-18, RNF-12	X	
RArq-16	El sistema debe permitir la exportación de resultados para su análisis posterior.	RF-19, RF-20	X	
Requerimientos de organización y mejora futura				
RArq-17	El sistema debe organizar sus módulos de manera que facilite el mantenimiento de la herramienta.	RNF-14, RNF-15	X	
RArq-18	El sistema debe permitir la incorporación de nuevas categorías de señas o actividades en futuras versiones.	RNF-15		X

Nota: Elaboración propia, 2026

Los requerimientos de arquitectura presentados en la *Tabla 16* permiten establecer una visión general de los componentes necesarios para implementar la herramienta. Estos requerimientos complementan los requerimientos funcionales y no funcionales, ya que definen la base tecnológica sobre la cual se ejecutará el reconocimiento de señas, la interacción con el usuario y el almacenamiento de resultados.

3.2.5.6 Requerimientos del dataset

Los requerimientos del dataset describen las condiciones que deben cumplir los datos utilizados para el entrenamiento, validación y prueba de los modelos de reconocimiento de señas. Estos requerimientos permiten organizar la información necesaria para que el sistema pueda identificar señas estáticas y dinámicas de manera adecuada.

En la *Tabla 17* se presentan los requerimientos del dataset definidos para el sistema propuesto, considerando la necesidad de contar con muestras organizadas, etiquetadas y compatibles con los modelos de reconocimiento empleados en la herramienta.

Tabla 17
Requerimientos del dataset

REQUERIMIENTOS DE DATASET (RD)					
Código	**Requerimiento**	**Necesidad asociada**	**PRIORIDAD**		
			Alta	**Media**	**Baja**
Requerimientos de contenido					
RD-01	El dataset debe incluir muestras de señas estáticas correspondientes al abecedario dactilológico.	RF-07	X		
RD-02	El dataset debe incluir muestras de señas estáticas correspondientes a números.	RF-08	X		
RD-03	El dataset debe incluir muestras de señas dinámicas correspondientes a palabras seleccionadas.	RF-09	X		
RD-04	Cada muestra del dataset debe estar asociada a una etiqueta que identifique la seña correspondiente.	RF-07, RF-08, RF-09	X		
Requerimientos de organización					
RD-05	El dataset debe organizarse por categorías de señas, como abecedario, números y palabras.	RF-03, RF-07, RF-08, RF-09	X		
RD-06	Los datos deben mantenerse ordenados para facilitar su uso durante el entrenamiento de los modelos.	RArq-08	X		
RD-07	Las etiquetas deben conservar una estructura clara para relacionar la salida del modelo con la seña reconocida.	RF-07, RF-08, RF-09	X		
Requerimientos de procesamiento					

RD-08	Las muestras deben contener información útil de la mano del usuario para el reconocimiento de la seña.	RF-05, RF-06	X		
RD-09	Los datos deben pasar por un proceso de preparación antes de ser utilizados por los modelos de reconocimiento.	RF-06, RArq-11	X		
RD-10	Las muestras de señas dinámicas deben conservar información relacionada con la secuencia del movimiento.	RF-09	X		
Requerimientos de entrenamiento y evaluación					
RD-11	El dataset debe dividirse en conjuntos de entrenamiento y prueba.	RArq-08	X		
RD-12	El dataset debe permitir evaluar el desempeño de los modelos de reconocimiento implementados.	RF-20	X		
RD-13	El dataset debe permitir identificar posibles errores de reconocimiento entre las diferentes señas.	RF-20		X	
Requerimientos de compatibilidad y mejora futura					
RD-14	El dataset debe ser compatible con el entorno de entrenamiento y con los modelos utilizados por el sistema.	RArq-06, RArq-08	X		
RD-15	El dataset debe permitir la incorporación de nuevas muestras.	RArq-18, RNF-15		X	
RD-16	El dataset debe incorporar muestras adicionales bajo diferentes condiciones de iluminación, posición o distancia para mejorar futuras versiones del sistema.	RNF-15			X

Nota: Elaboración propia, 2026

Los requerimientos del dataset presentados en la *Tabla 17* permiten definir las condiciones mínimas que deben cumplir los datos utilizados por el sistema. Estos requerimientos son importantes porque el desempeño del reconocimiento de señas depende no solo del modelo

empleado, sino también de la calidad, organización y etiquetado de las muestras utilizadas durante el entrenamiento y evaluación.

3.2.5.7 Matriz de trazabilidad

La matriz de trazabilidad permite relacionar las necesidades identificadas con los diferentes requerimientos definidos para el sistema. Esta relación facilita verificar que cada necesidad corresponde a cada uno de los requerimientos de stakeholders, funcionales, no funcionales, de arquitectura y del dataset.

En la Tabla 18 se presenta la matriz de trazabilidad del sistema propuesto. Esta matriz permite evidenciar la relación entre las necesidades obtenidas durante el levantamiento de información y los requerimientos establecidos para el desarrollo de la herramienta.

Tabla 18
Matriz de trazabilidad de requerimientos

Necesidad	Requerimientos de stakeholders	Requerimientos funcionales	Requerimientos no funcionales	Requerimientos de arquitectura	Requerimientos del dataset
N-01	StSR-01, StSR-02, StSR-06	RF-11, RF-13	RNF-01, RNF-02, RNF-03	RArq-03, RArq-09	—
N-02	StSR-05, StSR-06	RF-12, RF-13	RNF-03, RNF-05	RArq-09, RArq-12	—
N-03	StSR-09, StSR-15	RF-16, RF-17, RF-18, RF-20, RF-21	RNF-11, RNF-12	RArq-14, RArq-15, RArq-16	RD-12, RD-13
N-04	StSR-01, StSR-07	RF-14	RNF-04	RArq-09	—
N-05	StSR-03, StSR-04, StSR-12	RF-04, RF-05, RF-06, RF-07, RF-08, RF-09, RF-10	RNF-05, RNF-06	RArq-02, RArq-06, RArq-07, RArq-08, RArq-11, RArq-12	RD-01, RD-02, RD-03, RD-04, RD-08, RD-10, RD-12

N-06	StSR-03, StSR-04	RF-04, RF-05, RF-06, RF-07, RF-08, RF-09, RF-10	RNF-05, RNF-06, RNF-07	RArq-01, RArq-08, RArq-11, RArq-13	RD-09, RD-11, RD-14
N-07	StSR-11, StSR-12, StSR-13	RF-04, RF-05, RF-06, RF-15	RNF-07, RNF-08, RNF-09, RNF-10	RArq-01, RArq-02, RArq-04, RArq-05, RArq-06, RArq-07, RArq-10, RArq-13	RD-14, RD-15, RD-16
N-08	StSR-01, StSR-02, StSR-10	RF-11, RF-12, RF-13, RF-14, RF-15	RNF-01, RNF-02, RNF-03, RNF-04, RNF-12	RArq-03, RArq-09, RArq-10	—
N-09	StSR-08, StSR-09, StSR-10, StSR-11, StSR-13, StSR-14	RF-01, RF-02, RF-03, RF-15	RNF-04, RNF-08, RNF-09, RNF-10	RArq-03, RArq-10, RArq-17, RArq-18	RD-05, RD-15
N-10	StSR-09, StSR-14, StSR-15	RF-16, RF-17, RF-18, RF-19, RF-20, RF-21	RNF-11, RNF-12, RNF-13	RArq-14, RArq-15, RArq-16	RD-11, RD-12, RD-13

Nota: Elaboración propia, 2026

La matriz presentada en la Tabla 18 permite comprobar que las necesidades identificadas se encuentran cubiertas por los requerimientos definidos para el sistema. De esta manera, se mantiene una relación ordenada entre el análisis inicial, las expectativas de los usuarios, las funciones del sistema, las condiciones de calidad, la arquitectura tecnológica y los datos necesarios para el entrenamiento y evaluación de los modelos de reconocimiento.

3.3 Fase de Diseño

Una vez definidos los requerimientos del sistema y establecida la matriz de trazabilidad, se procede con la fase de diseño. En esta etapa se determinan los componentes tecnológicos necesarios para el desarrollo de la herramienta, considerando los requerimientos funcionales, no funcionales, de arquitectura y del dataset definidos previamente.

El diseño del sistema contempla la selección de hardware, software y herramientas de desarrollo que permitan la captura de gestos, el procesamiento de imágenes, el reconocimiento de señas, la retroalimentación visual y el almacenamiento de resultados. Además, esta fase permite establecer la estructura general de funcionamiento de la herramienta antes de iniciar su implementación.

3.3.1 Selección de componentes tecnológicos

La selección de componentes tecnológicos tiene como finalidad identificar los elementos de hardware y software más adecuados para el desarrollo del sistema propuesto. Esta selección no se realiza únicamente a partir de características técnicas individuales, sino considerando la relación de cada componente con los requerimientos definidos en la fase anterior.

Para el sistema propuesto, los componentes seleccionados deben permitir el procesamiento local de imágenes, la ejecución de modelos de reconocimiento de señas, la interacción visual con el niño, el acceso del docente a la información y el almacenamiento de los resultados generados durante las sesiones de práctica.

Por esta razón, en esta sección se establecen los criterios de selección que servirán como base para comparar las alternativas disponibles. Posteriormente, estos criterios serán aplicados en la elección de los componentes de hardware, software, herramientas de entrenamiento, almacenamiento y visualización del sistema.

3.3.1.1 Criterios de selección

Los criterios de selección permiten evaluar las alternativas tecnológicas de forma ordenada y coherente con los requerimientos del sistema. Para ello, se consideraron aspectos relacionados con compatibilidad, funcionamiento local, facilidad de integración, disponibilidad, uso educativo y posibilidad de mantenimiento.

Como se muestra en la *Tabla 19*, los criterios definidos permiten orientar la elección de los componentes que formarán parte del sistema, procurando que estos se adapten a las necesidades técnicas y pedagógicas del proyecto.

Tabla 19
Criterios de selección de componentes tecnológicos

Código	Criterio de selección	Descripción
C-01	Compatibilidad con visión artificial	El componente debe permitir el uso de herramientas relacionadas con captura, procesamiento de imágenes y detección de la mano.
C-02	Compatibilidad con modelos de reconocimiento	El componente debe permitir la ejecución o integración de modelos utilizados para el reconocimiento de señas estáticas y dinámicas.
C-03	Funcionamiento local	El componente debe permitir que las funciones principales del sistema se ejecuten sin depender de servicios externos o conexión permanente a internet.
C-04	Facilidad de integración	El componente debe poder conectarse o trabajar adecuadamente con los demás elementos del sistema, como cámara, pantalla, base de datos o panel docente.
C-05	Disponibilidad	El componente debe estar disponible para el desarrollo, pruebas e implementación del prototipo.
C-06	Costo accesible	El componente debe ajustarse a los recursos disponibles para el proyecto, sin comprometer el funcionamiento principal del sistema.
C-07	Facilidad de uso y desarrollo	El componente debe contar con documentación, librerías o herramientas que faciliten su configuración, programación y depuración.
C-08	Adecuación al entorno educativo	El componente debe ser apropiado para su uso dentro de un ambiente adecuado, considerando la interacción con niños y la supervisión del docente.
C-09	Almacenamiento y gestión de datos	El componente debe permitir organizar, guardar o exportar la información generada durante las sesiones de práctica.
C-10	Escalabilidad y mejora futura	El componente debe permitir futuras actualizaciones, incorporación de nuevas señas, mejoras de interfaz o cambios en los modelos de reconocimiento.

Nota: Elaboración propia, 2026

Para la selección de los componentes se utilizará una valoración basada en el cumplimiento de los criterios establecidos. En la Tabla 20 , el valor **1** indicará que la alternativa cumple con el criterio evaluado, mientras que el valor **0** indicará que no lo cumple. La opción con mayor valoración será considerada como la alternativa más adecuada, siempre que responda a las necesidades principales del sistema.

Tabla 20
Escala de valoración para la selección de componentes

Valor	Descripción
1	Cumple con el criterio evaluado.
0	No cumple con el criterio evaluado.

Nota: Elaboración propia, 2026

Los criterios presentados en la *Tabla 19* y la escala de valoración de la Tabla 20 servirán como base para la selección de los componentes tecnológicos del sistema. De esta manera, la elección de hardware y software se realiza de forma justificada y relacionada con los requerimientos definidos previamente.

3.3.1.2 Selección de Hardware

La selección de hardware se realizó a partir de los requerimientos definidos previamente, con el propósito de identificar los componentes físicos más adecuados para el funcionamiento del sistema. Para ello, se consideraron elementos relacionados con procesamiento local, captura visual, interacción con el usuario, almacenamiento, conectividad, alimentación y estructura física.

El sistema propuesto requiere una placa embebida capaz de ejecutar tareas de visión artificial, una cámara para capturar los gestos del niño, una pantalla para mostrar la interfaz gráfica, un

medio de almacenamiento local, conectividad para el panel docente, una fuente de alimentación estable y una estructura física que proteja los componentes internos.

Para la evaluación de alternativas se utilizó la escala definida anteriormente, donde el valor **1** indica que la opción cumple con el requerimiento evaluado y el valor **0** indica que no lo cumple completamente. Las fichas técnicas de los componentes seleccionados se presentan en el **Anexo C**: .

a. Selección de la placa embebida

La placa embebida constituye el componente principal del sistema, debido a que se encarga de ejecutar los procesos de captura, procesamiento de imagen, reconocimiento de señas, visualización de resultados y almacenamiento local de información. Por este motivo, la placa seleccionada debe ser compatible con herramientas de visión artificial, modelos de reconocimiento y operación local. En la *Tabla 21* se presenta la comparación de alternativas consideradas.

Tabla 21
Evaluación y selección de la placa embebida

Requerimiento / criterio evaluado	Raspberry Pi 4	Raspberry Pi 5	Coral Dev Board	Jetson Nano eMMC	Jetson Orin Nano	Orange Pi 5
RArq-01	1	1	1	**1**	1	1
RArq-05	1	1	1	**1**	1	1
RArq-06	1	1	1	**1**	1	1
RArq-07	1	1	1	**1**	1	1
RArq-08	0	1	1	**1**	1	1
RArq-10	1	1	1	**1**	1	1
RArq-11	1	1	1	**1**	1	1
RArq-12	1	1	1	**1**	1	1
RArq-13	0	0	1	**1**	1	1
RArq-14	1	1	1	**1**	1	1
RArq-17	1	1	1	**1**	1	1
RNF-05	0	1	1	**1**	1	1
RNF-07	1	1	1	**1**	1	1
RNF-08	1	1	1	**1**	1	1
RNF-10	1	1	1	**1**	1	1
Subtotal de requerimientos	**12**	**14**	**15**	**15**	**15**	**15**

C-05	1	0	0	**1**	0	0
C-06	1	1	0	**1**	0	1
C-07	1	1	0	**1**	1	0
C-08	1	1	1	**1**	0	1
C-09	0	0	1	**1**	1	1
C-10	0	1	1	**1**	1	1
Subtotal criterios de selección	**4**	**5**	**3**	**6**	**3**	**5**
Puntaje final	**16**	**18**	**18**	**21**	**18**	**19**

Nota: Elaboración propia, 2026

Con base en la evaluación realizada, se selecciona la NVIDIA Jetson Nano Kit 16 GB eMMC como placa embebida para el desarrollo del prototipo. Esta alternativa cumple con los requerimientos asociados al procesamiento local, conexión de periféricos, compatibilidad con visión artificial, integración de modelos de reconocimiento, ejecución de un servidor local, almacenamiento de datos y funcionamiento sin dependencia de internet.

Las especificaciones técnicas ampliadas de las placas consideradas se presentan en el **Anexo C**: .

b. Selección del dispositivo de captura visual

La selección del dispositivo de captura visual es importante dentro del diseño del sistema, debido a que este componente permite obtener el flujo de video necesario para observar los gestos realizados por el usuario. A partir de esta captura, el sistema realiza la detección de la mano, la extracción de características y el posterior reconocimiento de la seña.

En la *Tabla 22* se presentan y comparan las alternativas consideradas para la captura visual del sistema, tomando en cuenta criterios como: compatibilidad con la placa embebida, captura continua de video, detección de la mano, extracción de puntos característicos y facilidad de integración con el entorno de procesamiento.

Tabla 22
Selección del dispositivo de captura visual

Requerimiento / criterio evaluado	Cámara USB 720p	Cámara CSI IMX219 8 MP	Raspberry Pi Camera Module 3	Intel RealSense D435
StSR-12	1	1	1	1
RArq-02	1	1	1	1
RArq-05	1	1	0	1
RF-04	1	1	1	1
RF-05	1	1	1	1
RF-06	0	1	1	1
RArq-11	1	1	1	1
RNF-05	0	1	1	1
RNF-09	1	1	1	0
RNF-10	1	1	0	1
RD-08	0	1	1	1
Subtotal de requerimientos	**8**	**11**	**9**	**10**
C-01	1	1	1	1
C-04	1	1	0	1
C-05	1	1	1	0
C-06	1	1	1	0
C-07	1	1	0	0
C-08	0	1	1	0
Puntaje final	**13**	**17**	**13**	**12**

Nota: Elaboración propia, 2026

A partir de la evaluación realizada, se seleccionó la Cámara CSI IMX219, debido a que presenta un equilibrio adecuado entre calidad de imagen, facilidad de integración, disponibilidad, costo accesible y compatibilidad con la placa embebida utilizada en el prototipo.

c. Medio de interacción visual

La selección del medio de interacción visual es necesaria para presentar al niño la interfaz gráfica, la seña que debe practicar, los mensajes de apoyo y la retroalimentación generada durante la actividad. Debido a que el sistema está orientado a niños de 6 a 8 años, se requiere un medio de visualización claro, amigable y compatible con la placa embebida.

En la *Tabla 23* se presentan y comparan las alternativas consideradas para la interacción visual del sistema, tomando en cuenta criterios como: visualización de la interfaz, presentación de actividades, retroalimentación visual, facilidad de uso y adecuación al usuario infantil.

Tabla 23
Selección del Medio de interacción visual

Requerimiento / criterio evaluado	Monitor HDMI	Pantalla HDMI táctil	Pantalla táctil DSI Raspberry Pi	Tablet
StSR-01	1	**1**	1	1
StSR-02	1	**1**	1	1
StSR-05	1	**1**	1	1
StSR-06	1	**1**	1	1
StSR-07	1	**1**	1	1
StSR-10	1	**1**	0	0
RArq-03	1	**1**	1	1
RArq-05	1	**1**	0	0
RArq-09	1	**1**	1	1
RF-11	1	**1**	1	1
RF-12	1	**1**	1	1
RF-13	1	**1**	1	1
RF-14	1	**1**	1	1
RNF-01	1	**1**	1	1
RNF-02	1	**1**	1	1
RNF-03	1	**1**	1	1
RNF-04	1	**1**	0	0
RNF-10	1	**1**	0	1
Subtotal de requerimientos	**18**	**18**	**15**	**15**
C-04	1	**1**	0	0
C-05	0	**1**	1	1
C-06	1	**1**	0	0
C-07	1	**1**	0	0
C-08	1	**1**	1	1
Puntaje final	**23**	**24**	**17**	**17**

Nota: Elaboración propia, 2026

Con base en la evaluación realizada, se selecciona la pantalla HDMI táctil como medio de interacción visual principal para el prototipo. Esta alternativa permite mostrar la interfaz gráfica del estudiante, presentar instrucciones, visualizar la seña objetivo, mostrar retroalimentación y facilitar la interacción del niño con la herramienta. Además, su función táctil representa una ventaja para el uso didáctico, ya que permite seleccionar opciones o iniciar actividades sin depender únicamente de periféricos externos.

d. Medio de almacenamiento

La selección del medio de almacenamiento es necesaria para alojar los archivos del sistema, librerías, modelos de reconocimiento, recursos visuales y base de datos local. Este componente permite que el sistema conserve la información generada durante las sesiones de práctica y funcione sin depender de servicios externos de almacenamiento.

En la *Tabla 24* se presentan y comparan las alternativas consideradas para el almacenamiento del sistema, tomando en cuenta criterios como: almacenamiento de datos, conservación local de resultados, compatibilidad con el entorno de ejecución y capacidad para alojar los recursos necesarios del prototipo.

Tabla 24
Selección del medio de almacenamiento

Requerimiento / criterio evaluado	Tarjeta microSD GB	Memoria eMMC integrada de 16 GB	Memoria USB externa	SSD externo	Almacenamiento en la nube
RArq-10	**1**	1	1	1	1
RArq-13	**1**	1	1	1	0
RArq-14	**1**	1	1	1	0
RArq-17	**1**	1	1	1	0
RNF-07	**1**	1	0	1	0
RNF-08	**1**	1	1	1	0
RNF-10	**1**	1	0	0	1
Subtotal de requerimientos	**7**	**7**	**5**	**6**	**2**
C-04	**1**	1	1	0	0
C-05	**1**	0	1	0	0
C-06	**1**	1	1	0	0
C-07	**1**	1	1	0	0
C-09	**1**	1	0	1	0
C-10	**1**	0	1	1	1
Puntaje final	**13**	**11**	**10**	**8**	**3**

Nota: Elaboración propia, 2026

Con base en la evaluación realizada, se selecciona la tarjeta microSD como medio de almacenamiento principal del prototipo.

e. Medio de conectividad

La selección del medio de conectividad es importante para permitir el acceso al panel docente desde un dispositivo conectado a la misma red local del sistema. Esta comunicación permite supervisar actividades, consultar resultados y descargar información sin depender necesariamente de una conexión externa a internet.

En la Tabla 25 se presentan y comparan las alternativas consideradas para la conectividad del sistema, tomando en cuenta criterios como: acceso local al panel docente, supervisión de actividades, funcionamiento sin internet y compatibilidad con los dispositivos del prototipo.

Tabla 25
Selección del medio de conectividad

Requerimiento / criterio evaluado	**Cable Ethernet**	**Adaptador Wi-Fi USB**	**Bluetooth**
StSR-09	1	**1**	0
StSR-10	1	**1**	0
StSR-13	1	**1**	1
RF-15	1	**1**	0
RF-18	1	**1**	0
RF-19	1	**1**	0
RNF-04	1	**1**	0
RNF-08	1	**1**	1
RNF-09	1	**1**	1
RNF-10	1	**1**	0
RArq-05	1	**1**	1
RArq-10	1	**1**	0
Subtotal de requerimientos	**12**	**12**	**4**
C-04	1	**1**	1
C-05	1	**1**	1
C-06	1	**1**	1
C-07	1	**1**	0
C-08	0	**1**	0
C-10	1	**1**	0
Subtotal de criterios complementarios	**5**	**6**	**3**
Puntaje final	**17**	**18**	**7**

Nota: Elaboración propia, 2026

Se selecciona el adaptador Wi-Fi USB como medio de conectividad principal del prototipo. Esta alternativa permite conectar la placa embebida a una red local inalámbrica, facilita el acceso del docente al dashboard desde un dispositivo externo y mantiene el funcionamiento local del sistema sin depender de una conexión permanente a internet.

f. Fuente de alimentación

La selección de la fuente de alimentación es necesaria para garantizar el funcionamiento estable de la placa embebida y de los componentes conectados al sistema. Una alimentación inadecuada puede generar interrupciones durante la captura de video, el procesamiento de imágenes o la ejecución de modelos de reconocimiento. Por esta razón, la fuente seleccionada debe proporcionar el voltaje y la corriente necesarios para mantener la operación continua del prototipo.

En la *Tabla 26* se presentan y comparan las alternativas consideradas para la alimentación del sistema, tomando en cuenta criterios como: estabilidad energética, continuidad de funcionamiento y adecuación a las condiciones del entorno educativo

Tabla 26
Selección de la fuente de alimentación

Requerimiento / criterio evaluado	Alimentación por puerto USB de computadora	Batería externa	Cargador conectado al puerto de fuente
RArq-04	0	1	**1**
RArq-05	0	1	**1**
RNF-06	0	0	**1**
RNF-07	0	1	**1**
RNF-09	0	1	**1**
RNF-10	0	0	**1**
Subtotal de requerimientos	**0**	**4**	**6**

C-04	1	1	**1**
C-05	1	1	**1**
C-06	1	1	**1**
C-07	0	1	**1**
C-08	0	1	**1**
Subtotal de criterios complementarios	**3**	**5**	**5**
Puntaje final	**3**	**9**	**11**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, el cargador AC/DC de 5 V – 4 A conectado al puerto de fuente obtiene el mayor puntaje final, debido a que cumple con todos los requerimientos considerados y con los criterios complementarios relacionados con facilidad de integración, disponibilidad, costo accesible, facilidad de uso, adecuación al entorno educativo y estabilidad energética.

La alimentación por puerto USB de computadora se descarta como alternativa principal porque no garantiza la corriente necesaria para alimentar de forma estable la placa embebida junto con los periféricos conectados. La batería externa puede ser útil en aplicaciones portátiles; sin embargo, su autonomía depende de la capacidad disponible y puede presentar variaciones durante el uso prolongado. Por esta razón, se selecciona el cargador conectado al puerto de fuente como fuente de alimentación principal del prototipo.

g. Integración física del prototipo

La selección de la estructura física es importante para proteger los componentes internos del prototipo y adaptar la herramienta al entorno infantil. En este caso, la estructura no solo cumple una función de protección, sino que también contribuye a que el sistema tenga una apariencia amigable y adecuada para niños.

En la *Tabla 27* se presentan y comparan las alternativas consideradas para la estructura física del sistema, tomando en cuenta criterios como: apoyo visual al aprendizaje, incorporación de elementos lúdicos, claridad de los recursos gráficos y facilidad de manipulación por parte del docente.

Tabla 27
Selección de la estructura física

Requerimiento / criterio evaluado	**Montaje directo sobre mesa**	**Caja cerrada de acrílico o plástico**	**Base de madera**
StSR-10	0	0	**1**
StSR-12	0	1	**1**
RF-04	0	1	**1**
RF-11	1	1	**1**
RF-12	1	1	**1**
RNF-01	0	1	**1**
RNF-02	1	1	**1**
RNF-04	0	0	**1**
RNF-09	0	1	**1**
RNF-10	0	1	**1**
RArq-05	1	1	**1**
Subtotal de requerimientos	**4**	**9**	**11**
C-04	1	0	**1**
C-05	1	1	**1**
C-06	1	0	**1**
C-07	0	0	**1**
C-08	0	1	**1**
C-10	0	0	**1**
Subtotal de criterios complementarios	**3**	**2**	**6**
Puntaje final	**7**	**11**	**17**

Nota: Elaboración propia, 2026

Con base en la evaluación realizada, se selecciona la carcasa de madera como estructura física del prototipo. Esta alternativa permite ubicar la pantalla táctil en una posición visible para el

niño, fijar la cámara CSI hacia el área de reconocimiento de gestos, colocar la Jetson Nano en una zona accesible y organizar el cableado de alimentación y conectividad.

Por lo tanto, la carcasa de madera representa una alternativa adecuada para el sistema propuesto, ya que proporciona estabilidad, bajo costo, facilidad de construcción, disponibilidad del material y adaptación al entorno educativo.

h. Resumen de hardware seleccionado

Una vez evaluadas las alternativas de hardware para cada bloque del sistema, en la Tabla 28 se presenta el resumen de los componentes seleccionados para la implementación del prototipo.

Tabla 28
Resumen de hardware seleccionado

Elemento del sistema	Hardware seleccionado	Función dentro del prototipo
Placa embebida	Jetson Nano	Ejecutar el sistema local, procesar la captura visual, ejecutar el modelo de reconocimiento y alojar la aplicación.
Dispositivo de captura visual	Cámara CSI IMX219 de 8 MP	Capturar imágenes o video de la mano del usuario para el reconocimiento de señas dactilológicas.
Medio de interacción visual	Pantalla HDMI táctil de 7 pulgadas	Mostrar la interfaz del estudiante, la seña objetivo, instrucciones, recursos visuales y retroalimentación.
Medio de almacenamiento	Tarjeta microSD de 64 GB	Alojar el sistema operativo clonado, la aplicación, los modelos de reconocimiento, la base de datos local, recursos gráficos y reportes.
Medio de conectividad	Adaptador Wi-Fi USB	Conectar la placa embebida a una red local para permitir el acceso al dashboard docente desde otro dispositivo.
Fuente de alimentación	Cargador AC/DC de 5 V – 4 A conectado al puerto de fuente	Suministrar energía estable a la Jetson Nano y a los periféricos conectados al prototipo.
Estructura física	Base de madera en forma de nube blanca	Integrar y sostener los componentes del prototipo de forma estable, accesible y adecuada para el entorno infantil.

Nota: Elaboración propia, 2026

El hardware seleccionado permite el funcionamiento local del sistema y su uso en el aula. La Jetson Nano procesa la información, la cámara CSI captura los gestos, la pantalla táctil muestra la interfaz, la microSD almacena el sistema, el adaptador Wi-Fi permite la conexión local y la fuente AC/DC suministra energía.

La base de madera organiza los componentes y mejora la presentación del prototipo, cumpliendo con los requerimientos de captura, procesamiento, visualización, almacenamiento, conectividad, alimentación e integración física.

3.3.1.3 Selección de Software

La selección de software se realizó considerando los requerimientos funcionales, no funcionales, de arquitectura y del dataset definidos para el sistema. En esta etapa se analizaron las herramientas necesarias para la ejecución local del prototipo, el procesamiento de imágenes, la detección de la mano, el reconocimiento de señas, la interfaz gráfica, el panel docente, el almacenamiento de resultados y la exportación de información.

Las alternativas consideradas fueron evaluadas mediante una escala de cumplimiento, donde el valor 1 indica que la herramienta cumple con el requerimiento evaluado y el valor 0 indica que no lo cumple completamente. La configuración e instalación de las herramientas seleccionadas se presenta en el Anexo D.

a. Sistema operativo

La selección del entorno base es importante, debido a que este permite ejecutar el sistema operativo, librerías, controladores y herramientas necesarias para el funcionamiento de la placa embebida. Al tratarse de un sistema basado en visión artificial e inteligencia artificial, el

entorno debe ser compatible con la plataforma seleccionada y permitir la ejecución local de los procesos del sistema.

En la *Tabla 29* se presentan y comparan las alternativas consideradas para el entorno base del sistema, tomando en cuenta criterios como: compatibilidad con la placa embebida, soporte para visión artificial, ejecución de modelos, funcionamiento local y facilidad de integración con los componentes del prototipo.

Tabla 29
Selección del Sistema Operativo

Requerimiento / criterio evaluado	NVIDIA JetPack SDK / Jetson Linux	Ubuntu Desktop / Server genérico	Raspberry Pi OS	Windows
RF-04	**1**	1	1	1
RF-05	**1**	1	1	1
RF-06	**1**	1	1	1
RF-15	**1**	1	1	1
RF-18	**1**	1	1	1
RF-19	**1**	1	1	1
RNF-07	**1**	1	0	0
RNF-08	**1**	1	1	1
RNF-10	**1**	0	0	0
RArq-05	**1**	0	0	0
RArq-10	**1**	1	1	1
Subtotal de requerimientos	**11**	**9**	**8**	**8**
C-01	**1**	1	1	1
C-04	**1**	0	0	0
C-05	**1**	1	0	0
C-06	**1**	1	1	0
C-07	**1**	1	1	1
C-08	**1**	1	0	0
C-10	**1**	1	0	0

Subtotal de criterios complementarios	**7**	**6**	**3**	**2**
Puntaje final	**18**	**15**	**11**	**10**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, NVIDIA JetPack SDK / Jetson Linux obtiene el mayor puntaje final, debido a que cumple con los requerimientos relacionados con captura de imágenes, procesamiento de visión artificial, ejecución local, compatibilidad con periféricos, almacenamiento local y despliegue del dashboard docente.

b. Lenguaje de programación

La selección del lenguaje de programación es necesaria para integrar los procesos de captura de video, procesamiento de imágenes, ejecución de modelos, almacenamiento de resultados y funcionamiento del panel docente. El lenguaje seleccionado debe permitir el desarrollo de aplicaciones de visión artificial, inteligencia artificial y servicios web locales.

En la **Tabla *30*** se presentan y comparan los lenguajes considerados para el desarrollo del sistema, tomando en cuenta criterios como: compatibilidad con librerías de visión artificial, integración con modelos de reconocimiento, desarrollo del servidor local, manejo de base de datos y facilidad de mantenimiento.

Tabla 30
Selección del lenguaje de programación

Requerimiento / criterio evaluado	Python 3	C++	JavaScript
RF-04	**1**	1	1
RF-05	**1**	1	0
RF-06	**1**	1	0
RF-07	**1**	1	0
RF-15	**1**	0	1
RF-18	**1**	1	1
RF-19	**1**	1	1
RNF-07	**1**	1	1

RNF-08	**1**	1	1
RNF-10	**1**	1	0
RArq-10	**1**	0	1
RArq-14	**1**	1	1
Subtotal de requerimientos	**12**	**10**	**8**
C-01	**1**	1	0
C-04	**1**	0	1
C-05	**1**	1	1
C-06	**1**	1	1
C-07	**1**	0	1
C-10	**1**	1	1
Subtotal de criterios complementarios	**6**	**4**	**5**
Puntaje final	**18**	**14**	**13**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, Python 3 obtiene el mayor puntaje final, debido a que cumple con los requerimientos relacionados con captura de imágenes, detección de la mano, extracción de puntos característicos, reconocimiento de señas, almacenamiento local, generación de reportes y despliegue del dashboard docente.

C++ representa una alternativa de alto rendimiento y puede utilizarse en sistemas embebidos; sin embargo, presenta mayor complejidad de desarrollo e integración para el alcance del prototipo. JavaScript / Node.js resulta adecuado para el desarrollo web y la construcción de interfaces interactivas, pero no constituye la opción principal para los módulos de visión artificial y reconocimiento de señas. Java permite desarrollar aplicaciones multiplataforma y servicios, pero no ofrece la misma facilidad de integración con las librerías utilizadas para visión artificial en el prototipo.

c. Procesamiento de imágenes

La selección de la herramienta de procesamiento de imágenes es fundamental para obtener, tratar y visualizar el flujo de video capturado por la cámara. Esta herramienta permite adquirir los cuadros de video, mostrar la interfaz del estudiante, dibujar elementos visuales y preparar la imagen para la detección de la mano.

En la *Tabla 31* se presentan y comparan las alternativas consideradas para el procesamiento de imágenes, tomando en cuenta criterios como: captura continua de video, integración con la cámara, procesamiento de imagen, respuesta fluida y compatibilidad con el entorno de ejecución.

Tabla 31
Selección de la herramienta de procesamiento de imágenes

Requerimiento / criterio evaluado	**PIL/Pillow**	**MATLAB**	**OpenCV**
RF-04	0	1	**1**
RArq-07	0	1	**1**
RArq-11	1	1	**1**
RNF-05	0	0	**1**
RNF-10	1	0	**1**
Subtotal de requerimientos	**2**	**3**	**5**
C-01	0	1	**1**
C-04	1	0	**1**
C-05	1	0	**1**
C-06	1	0	**1**
C-07	1	0	**1**
C-10	0	1	**1**
Subtotal de criterios complementarios	**4**	**2**	**6**
Puntaje final	**6**	**5**	**11**

Nota: Elaboración propia, 2026

Mediante la evaluación realizada, se seleccionó **OpenCV** como herramienta de procesamiento de imágenes para el sistema propuesto. Esta elección se basa en su cumplimiento de los requerimientos definidos, como la captura continua del flujo de video (RF-04), el uso de

librerías de visión artificial para el análisis de la mano (RArq-07), el procesamiento de imágenes capturadas para obtener información útil (RArq-11), la interacción fluida durante la captura y reconocimiento de señas (RNF-05) y la compatibilidad con la cámara y la placa embebida (RNF-10).

OpenCV se utiliza para capturar el video proveniente de la cámara, mostrar la imagen en pantalla, dibujar elementos de apoyo visual, presentar mensajes de retroalimentación y construir la interfaz visual observada por el niño durante la actividad.

d. Detección de mano y extracción de características

La selección de la herramienta para detección de mano y extracción de características es necesaria para identificar la mano del usuario y obtener los puntos de referencia utilizados por los modelos de reconocimiento. Esta etapa transforma la imagen capturada en información útil para clasificar la seña realizada.

En la *Tabla 32* se presentan y comparan las alternativas consideradas para la detección de mano, tomando en cuenta criterios como: detección de la mano, extracción de puntos característicos, procesamiento local, respuesta fluida y compatibilidad con el entorno de visión artificial.

Tabla 32
Selección de la herramienta para detección de mano

Requerimiento / criterio evaluado	OpenPose	YOLO Pose / Keypoints	MediaPipe Hands
RF-05	1	1	**1**
RF-06	1	1	**1**
RF-07	1	1	**1**
RArq-07	1	1	**1**
RArq-08	1	1	**1**
RArq-11	0	1	**1**
RNF-05	0	1	**1**
RNF-10	0	1	**1**

Subtotal de requerimientos	**5**	**8**	**8**
C-01	1	1	**1**
C-04	0	0	**1**
C-05	0	1	**1**
C-06	1	1	**1**
C-07	0	0	**1**
C-10	1	1	**1**
Subtotal de criterios complementarios	**3**	**4**	**6**
Puntaje final	**8**	**12**	**14**

Nota: Elaboración propia, 2026

Mediante la evaluación realizada, se seleccionó MediaPipe Hands como herramienta para la detección de la mano y extracción de características. Esta elección se basa en su cumplimiento de los requerimientos definidos, como la detección de la mano del usuario (RF-05), la extracción de puntos característicos para el modelo de reconocimiento (RF-06), el uso de librerías de visión artificial para análisis de la mano (RArq-07), el procesamiento de imágenes capturadas (RArq-11) y la interacción fluida durante el reconocimiento de señas (RNF-05).

MediaPipe Hands permite obtener puntos de referencia de la mano, conocidos como landmarks, los cuales sirven como entrada para los modelos de reconocimiento. Esta herramienta resulta adecuada para el sistema debido a su integración con Python y OpenCV, además de su eficiencia para aplicaciones de visión artificial en tiempo real.

e. Framework de inteligencia artificial

La selección del framework de inteligencia artificial permite definir la herramienta utilizada para entrenar, cargar y ejecutar los modelos de reconocimiento de señas. Este componente es necesario para clasificar señas estáticas, como letras y números, y señas dinámicas asociadas a palabras seleccionadas.

En la *Tabla 33* se presentan y comparan las alternativas consideradas para el desarrollo de modelos de inteligencia artificial, tomando en cuenta criterios como: reconocimiento de señas

estáticas, reconocimiento de señas dinámicas, compatibilidad con datos procesados, facilidad de entrenamiento y despliegue local.

Tabla 33
Selección del framework de inteligencia artificial

Requerimiento / criterio evaluado	TensorFlow/Keras	PyTorch	Scikit-learn	MATLAB Deep Learning Toolbox
RF-07	**1**	1	1	1
RArq-08	**1**	1	1	1
RArq-11	**1**	1	1	0
RNF-05	**1**	1	1	0
RNF-07	**1**	0	1	0
RNF-08	**1**	1	1	0
RNF-10	**1**	1	0	0
Subtotal de requerimientos	**7**	**6**	**6**	**2**
C-01	**1**	1	0	1
C-04	**1**	1	1	0
C-05	**1**	1	1	0
C-06	**1**	1	1	0
C-07	**1**	0	1	0
C-10	**1**	1	0	1
Subtotal de criterios complementarios	**6**	**5**	**4**	**2**
Puntaje final	**13**	**11**	**10**	**4**

Nota: Elaboración propia, 2026

Mediante la evaluación realizada, se seleccionó TensorFlow/Keras como framework de inteligencia artificial para el sistema propuesto. Esta elección se basa en su cumplimiento de los requerimientos definidos, como el reconocimiento de señas estáticas del abecedario (RF-07), el reconocimiento de señas estáticas correspondientes a números (RF-08), el reconocimiento de señas dinámicas correspondientes a palabras seleccionadas (RF-09), la integración de modelos de reconocimiento para señas estáticas y dinámicas (RArq-08) y la compatibilidad con el entorno de entrenamiento y modelos utilizados por el sistema (RD-14).

TensorFlow/Keras permite desarrollar modelos ligeros para señas estáticas y modelos secuenciales para señas dinámicas. En el caso del sistema propuesto, esta herramienta permite

trabajar con los puntos de referencia obtenidos mediante MediaPipe y generar modelos capaces de clasificar las señas practicadas por el niño.

f. Motor de inferencia local

La selección del motor de inferencia local permite ejecutar los modelos de reconocimiento dentro de la placa embebida durante el uso del sistema. Esta etapa es importante porque el prototipo debe responder en tiempo adecuado y funcionar de manera local, sin depender de servicios externos.

En la *Tabla 34* se presentan y comparan las alternativas consideradas para la inferencia local, tomando en cuenta criterios como: ejecución local, optimización de modelos, rendimiento en la placa embebida, compatibilidad con los modelos desarrollados y funcionamiento sin conexión a internet.

Tabla 34
Selección del motor de inferencia local

Requerimiento / criterio evaluado	TensorFlow/Keras	TensorFlow Lite	ONNX Runtime	PyTorch
RF-07	**1**	1	1	1
RF-08	**1**	1	1	1
RF-12	**1**	1	1	1
RArq-08	**1**	1	1	1
RArq-11	**1**	1	1	1
RNF-05	**1**	1	1	1
RNF-07	**1**	1	1	0
RNF-08	**1**	1	1	1
RNF-10	**1**	1	0	1
Subtotal de requerimientos	**9**	**9**	**8**	**8**
C-01	**1**	1	1	1
C-04	**1**	0	0	0

C-05	**1**	1	1	1
C-06	**1**	1	1	1
C-07	**1**	0	0	0
C-10	**1**	1	1	1
Subtotal de criterios complementarios	**6**	**4**	**4**	**4**
Puntaje final	**15**	**13**	**12**	**12**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, TensorFlow/Keras obtiene el mayor puntaje final, debido a que cumple con los requerimientos relacionados con reconocimiento de señas, comparación de la seña realizada con la seña objetivo, generación de retroalimentación visual, ejecución local y compatibilidad con el flujo de software del prototipo.

TensorFlow Lite representa una alternativa adecuada para inferencia en dispositivos embebidos; sin embargo, requiere convertir el modelo a un formato específico y realizar ajustes adicionales para su ejecución. ONNX Runtime permite ejecutar modelos en formato ONNX y facilita la interoperabilidad entre frameworks, pero no fue seleccionado porque el modelo del prototipo se desarrolló directamente en TensorFlow/Keras. PyTorch también permite realizar inferencia local, pero no mantiene continuidad con el modelo implementado en el sistema.

g. Servidor local

La selección del servidor local es necesaria para permitir que el docente acceda al panel de supervisión desde un dispositivo conectado a la misma red local. Este servidor permite consultar información de sesiones, visualizar resultados y descargar reportes sin requerir una conexión permanente a internet.

En la *Tabla 35* se presentan y comparan las alternativas consideradas para el servidor local, tomando en cuenta criterios como: acceso al panel docente, consulta de resultados, descarga de información, funcionamiento local y facilidad de integración con Python.

Tabla 35
Selección del servidor local

Requerimiento / criterio evaluado	Flask	Django	FastAPI	Express.js
RF-15	**1**	1	1	1
RF-18	**1**	1	1	1
RF-19	**1**	1	1	1
RArq-10	**1**	1	1	1
RArq-11	**1**	1	1	1
RArq-14	**1**	1	1	1
RNF-04	**1**	0	1	0
RNF-08	**1**	1	1	1
RNF-10	**1**	1	1	0
Subtotal de requerimientos	**9**	**8**	**9**	**7**
C-04	**1**	0	1	0
C-05	**1**	1	1	1
C-06	**1**	1	1	1
C-07	**1**	0	1	0
C-08	**1**	0	1	0
C-10	**1**	1	1	1
Subtotal de criterios complementarios	**6**	**3**	**6**	**3**
Puntaje final	**15**	**11**	**15**	**10**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, Flask y FastAPI cumplen con la mayoría de los requerimientos considerados para el servidor local. Sin embargo, Flask se selecciona como alternativa principal debido a su facilidad de integración con Python 3, su estructura ligera y su adecuación para desarrollar un dashboard web local con rutas, plantillas HTML, archivos estáticos y conexión con una base de datos local.

Django representa una alternativa robusta para aplicaciones web más completas; no obstante, para el alcance del prototipo puede resultar más complejo de configurar y mantener. FastAPI es una alternativa moderna y eficiente para el desarrollo de APIs; sin embargo, el sistema propuesto requiere principalmente un dashboard local con páginas web, por lo que Flask resulta más directo para el desarrollo planteado. Express.js permite desarrollar servidores web

mediante JavaScript y Node.js, pero no mantiene la misma continuidad con el entorno principal del prototipo, desarrollado en Python 3.

h. Base de datos

La selección de la base de datos permite definir el mecanismo encargado de almacenar la información generada durante las sesiones de práctica. Esta información debe conservarse de forma local y estructurada, permitiendo su consulta por parte del docente.

En la **Tabla *36*** se presentan y comparan las alternativas consideradas para la base de datos, tomando en cuenta criterios como base de datos local, consulta de información, almacenamiento estructurado, conservación local de datos y compatibilidad con el entorno del sistema.

Tabla 36
Selección de base de datos y exportación de resultados

Requerimiento / criterio evaluado	SQLite	MySQL / MariaDB	PostgreSQL	MongoDB
RF-16	**1**	1	1	1
RF-17	**1**	1	1	1
RF-18	**1**	1	1	1
RF-19	**1**	1	1	1
RArq-10	**1**	1	1	1
RArq-11	**1**	1	1	1
RArq-13	**1**	1	1	1
RArq-14	**1**	1	1	1
RNF-07	**1**	0	0	0
RNF-08	**1**	1	1	1
Subtotal de requerimientos	**10**	**9**	**9**	**9**
C-04	**1**	0	0	0

C-05	**1**	1	1	1
C-06	**1**	1	1	1
C-07	**1**	0	0	0
C-09	**1**	1	1	1
C-10	**1**	1	1	1
Subtotal de criterios complementarios	**6**	**4**	**4**	**4**
Puntaje final	**16**	**13**	**13**	**13**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, SQLite obtiene el mayor puntaje final, debido a que cumple con los requerimientos relacionados con registro de sesiones, almacenamiento de resultados, consulta de información, descarga de reportes, funcionamiento local e integración con el servidor Flask.

MySQL / MariaDB y PostgreSQL son alternativas robustas para aplicaciones con mayor cantidad de usuarios, concurrencia y administración avanzada; sin embargo, requieren la instalación y configuración de un servidor de base de datos, lo que incrementa la complejidad del prototipo. MongoDB permite almacenar documentos flexibles y puede utilizarse en aplicaciones web modernas, pero no resulta necesario para el tipo de información estructurada que maneja el sistema, como usuarios, sesiones, actividades y resultados.

i. Resumen de software seleccionado

Una vez evaluadas las alternativas de software para cada bloque del sistema, en la **Tabla *37*** se presenta el resumen de las herramientas seleccionadas para la implementación del prototipo.

Tabla 37
Resumen de software seleccionado

Componente de software	Herramienta seleccionada	Función dentro del sistema
Sistema operativo	NVIDIA JetPack SDK / Jetson Linux basado en Ubuntu	Permitir la ejecución local del sistema en la Jetson Nano, gestionando periféricos, librerías y

		recursos de la placa embebida.
Lenguaje de programación	Python 3	Implementar los módulos de captura, procesamiento, reconocimiento, base de datos, servidor local e interfaz del sistema.
Procesamiento de imágenes	OpenCV	Capturar fotogramas desde la cámara, procesar imágenes y preparar los datos visuales para las etapas de detección y reconocimiento.
Detección de mano y extracción de características	MediaPipe Hands	Detectar la mano del usuario y extraer puntos característicos de la palma y los dedos para el reconocimiento de señas.
Framework de inteligencia artificial	TensorFlow/Keras	Construir, entrenar y evaluar el modelo de reconocimiento de señas dactilológicas.
Motor de inferencia local	TensorFlow Lite/Keras	Cargar el modelo entrenado y ejecutar predicciones localmente a partir de los datos generados por MediaPipe Hands.
Servidor local	Flask	Alojar el dashboard docente dentro de la red local, permitiendo consultar resultados y descargar reportes.
Base de datos	SQLite	Almacenar de forma local usuarios, sesiones, actividades, resultados y registros generados por el sistema.

Nota: Elaboración propia, 2026

La selección de software realizada permite definir las herramientas que cumplen con los requerimientos del sistema. Esta selección constituye una base técnica para las etapas posteriores de diseño e implementación, sin desarrollar todavía aspectos relacionados con instalación, configuración, programación, pruebas o validación.

3.3.1.4 Selección de Dataset

La selección del dataset se realizó considerando los requerimientos del dataset, funcionales, no funcionales y de arquitectura definidos para el sistema. En esta etapa se analizaron las alternativas necesarias para disponer de datos adecuados para el entrenamiento y evaluación de los modelos de reconocimiento de señas.

El dataset debe permitir trabajar con señas estáticas, como letras y números, así como con señas dinámicas asociadas a palabras seleccionadas. Además, debe contar con una estructura organizada por categorías y etiquetas, ser compatible con herramientas de visión e inteligencia artificial, y permitir la incorporación de nuevas muestras en futuras versiones del sistema.

Las alternativas consideradas fueron evaluadas mediante una escala de cumplimiento, donde el valor **1** indica que la alternativa cumple con el requerimiento evaluado y el valor **0** indica que no lo cumple completamente.

a. Origen del dataset

La selección del origen del dataset permite definir la procedencia de las muestras utilizadas para el entrenamiento de los modelos de reconocimiento. Esta decisión es importante, debido a que el conjunto de datos debe cubrir las categorías del sistema, permitir el etiquetado de cada seña y adaptarse a posibles variaciones propias de la Lengua de Señas.

En la **Tabla *38*** se presentan y comparan las alternativas consideradas para el origen del dataset, tomando en cuenta requerimientos relacionados con señas estáticas, señas dinámicas, etiquetado, ampliación futura y variabilidad de muestras.

Tabla 38
Selección del origen del dataset

Requerimiento / criterio evaluado	Synthetic ASL Numbers	ASL Hand Signs Dataset	ASL Alphabet Dataset	WLASL	Sign Language MNIST	ASL Fingerspelling Dataset	Captura propia	Dataset híbrido
RD-01	0	1	1	0	1	1	1	**1**
RD-02	1	1	0	0	0	0	1	**1**
RD-03	0	0	0	1	0	0	1	**1**
RD-04	1	1	1	1	1	1	1	**1**
RD-14	1	1	1	1	1	1	1	**1**
RD-15	0	0	0	0	0	0	1	**1**
RD-16	1	1	1	1	1	1	0	**1**
Subtotal de requerimientos	**4**	**5**	**4**	**4**	**4**	**4**	**6**	**7**
C-04	1	1	1	0	1	1	1	**1**
C-05	1	1	1	1	1	1	1	**1**
C-06	1	1	1	1	1	1	1	**1**
C-07	1	1	1	0	1	1	0	**1**
C-10	0	0	0	0	0	0	1	**1**
Subtotal de criterios complementarios	**4**	**4**	**4**	**2**	**4**	**4**	**4**	**5**
Puntaje final	**8**	**9**	**8**	**6**	**8**	**8**	**10**	**12**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, el **dataset híbrido** obtiene el mayor puntaje final, debido a que permite cubrir de manera más completa las categorías requeridas por el sistema. Esta alternativa combina datasets externos específicos, como *Synthetic ASL Numbers*, *ASL Hand Signs Dataset*, *ASL Alphabet Dataset* y *WLASL*, con capturas propias que permiten adaptar el conjunto de datos a las condiciones reales del prototipo.

Synthetic ASL Numbers resulta útil para la categoría de números, pero no cubre abecedario ni palabras dinámicas. ASL Hand Signs Dataset y ASL Alphabet Dataset aportan muestras para el abecedario dactilológico, aunque no incluyen señas dinámicas correspondientes a palabras. Por su parte, WLASL contiene señas a nivel de palabra en formato de video; sin embargo, puede no coincidir completamente con las palabras seleccionadas para el sistema y presenta mayor complejidad de integración. En contraste, la captura propia permite generar muestras

ajustadas al prototipo y a las condiciones reales de uso, aunque puede presentar menor variabilidad si se utiliza como única fuente.

b. Representación de los datos

La selección de la representación de los datos permite definir el tipo de información que será utilizada por los modelos de reconocimiento. En sistemas basados en visión artificial, las muestras pueden representarse mediante imágenes completas, videos completos o características extraídas de la mano.

En la **Tabla *39*** se presentan y comparan las alternativas consideradas para la representación de los datos, tomando en cuenta criterios como extracción de información útil de la mano, preparación de datos antes del entrenamiento, conservación de información temporal, procesamiento de imágenes y uso adecuado de recursos computacionales.

Tabla 39
Selección de la representación de los datos

Requerimiento / criterio evaluado	Imagen completa	Imagen recortada o preprocesada	Descriptores geométricos manuales	Puntos característicos de la mano
RD-05	1	1	1	**1**
RD-06	0	1	1	**1**
RD-07	0	1	1	**1**
RD-14	1	1	1	**1**
RD-16	0	0	1	**1**
Subtotal de requerimientos	**2**	**4**	**5**	**5**
C-04	1	1	0	**1**
C-05	1	1	1	**1**
C-06	1	1	1	**1**
C-07	0	1	0	**1**
C-10	0	0	0	**1**
Subtotal de criterios complementarios	**3**	**4**	**2**	**5**
Puntaje final	**5**	**8**	**7**	**10**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, la representación mediante **puntos característicos de la mano** obtiene el mayor puntaje final. Esta alternativa permite representar la seña mediante coordenadas numéricas correspondientes a puntos clave de la mano, también conocidos como *landmarks*. De esta manera, el modelo no trabaja directamente con todos los píxeles de la imagen, sino con información estructurada relacionada con la posición de la palma y los dedos.

La imagen completa conserva toda la información visual capturada por la cámara; sin embargo, también incluye elementos que no son relevantes para el reconocimiento, como el fondo, iluminación, sombras u objetos externos. La imagen recortada o preprocesada mejora esta condición porque reduce parte de la información innecesaria, aunque todavía depende de características visuales que pueden variar según el entorno de captura. Por otra parte, los descriptores geométricos manuales permiten trabajar con medidas numéricas, pero requieren definir manualmente qué distancias, ángulos o relaciones serán utilizadas, lo que puede limitar la flexibilidad del sistema.

c. Organización del dataset

La organización del dataset permite estructurar las muestras de acuerdo con las categorías de reconocimiento definidas para el sistema. Esta organización es necesaria para mantener una relación clara entre las muestras, sus etiquetas y las salidas esperadas de los modelos.

En la ***Tabla 40*** se presentan y comparan las alternativas consideradas para la organización del dataset, tomando en cuenta criterios como separación por categorías, ordenamiento para entrenamiento, relación entre etiquetas y salidas del modelo, evaluación del desempeño y posibilidad de ampliación futura.

Tabla 40
Selección de la organización del dataset

Requerimiento / criterio evaluado	Archivos sin estructura definida	Organización únicamente por carpetas	Organización mediante archivos de etiquetas	Organización híbrida estructurada por carpetas, landmarks y etiquetas
RD-04	0	1	1	**1**
RD-08	0	1	0	**1**
RD-09	0	1	1	**1**
RD-10	0	0	1	**1**
RD-11	0	0	1	**1**
RD-14	0	1	1	**1**
RD-15	0	1	1	**1**
Subtotal de requerimientos	**0**	**5**	**6**	**7**
C-04	0	1	1	**1**
C-05	1	1	1	**1**
C-06	1	1	1	**1**
C-07	0	1	0	**1**
C-10	0	0	1	**1**
Subtotal de criterios complementarios	**2**	**4**	**4**	**5**
Puntaje final	**2**	**9**	**10**	**12**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, la organización híbrida estructurada por carpetas, landmarks y etiquetas obtiene el mayor puntaje final. Esta alternativa permite organizar el dataset en dos fases principales: la organización en crudo de las muestras originales y la organización procesada de los datos numéricos utilizados por los modelos de reconocimiento.

La opción de archivos sin estructura definida no resulta adecuada, debido a que dificulta la identificación de clases, el etiquetado y la preparación de datos para el entrenamiento. La organización únicamente por carpetas permite clasificar las muestras originales, pero no es suficiente cuando se requiere almacenar landmarks, arreglos numéricos y etiquetas codificadas. Por otra parte, la organización mediante archivos de etiquetas permite relacionar rutas y clases, aunque resulta menos clara si no se complementa con una estructura física ordenada de carpetas.

Con base en la evaluación realizada, se selecciona la organización híbrida estructurada por carpetas, landmarks y etiquetas. Esta alternativa permite mantener ordenados los datos originales y, al mismo tiempo, generar una representación procesada compatible con TensorFlow/Keras.

d. División del dataset

La división del dataset permite separar las muestras en conjuntos destinados al entrenamiento y evaluación del modelo. Esta etapa es necesaria para que el modelo aprenda a partir de un conjunto de datos y, posteriormente, sea evaluado con muestras no utilizadas durante el ajuste directo de sus pesos.

En la **Tabla *41*** se presentan y comparan las alternativas consideradas para la división del dataset, tomando en cuenta criterios como separación de datos para entrenamiento y evaluación, medición del desempeño del modelo, identificación de errores por seña, generación de resultados métricos y uso adecuado de recursos.

Tabla 41
Selección de la división del dataset

Requerimiento / criterio evaluado	70/30	80/20	90/10	60/20/20	k-fold
RD-04	1	**1**	1	1	1
RD-09	1	**1**	1	1	1
RD-12	0	**1**	1	0	1
RD-13	1	**1**	0	1	1
RD-14	1	**1**	1	1	1
RD-16	1	**1**	0	1	1
Subtotal de requerimientos	**5**	**6**	**4**	**5**	**6**
C-04	1	**1**	1	1	0
C-05	1	**1**	1	1	1
C-06	1	**1**	1	1	1
C-07	1	**1**	1	0	0
C-10	0	**1**	0	1	1
Subtotal de criterios complementarios	**4**	**5**	**4**	**4**	**3**

Puntaje final	**9**	**11**	**8**	**9**	**9**

Nota: Elaboración propia, 2026

De acuerdo con la matriz de evaluación, la división **80% entrenamiento y 20% prueba** obtiene el mayor puntaje final. Esta alternativa permite que la mayor parte de las muestras sea utilizada para el aprendizaje del modelo, mientras que una proporción suficiente del dataset se reserva para evaluar el desempeño con datos no vistos durante el entrenamiento.

La división 70/30 permite contar con un conjunto de prueba más amplio; sin embargo, reduce la cantidad de muestras disponibles para el entrenamiento, lo cual puede afectar el aprendizaje del modelo, especialmente cuando el dataset no es muy grande. La división 90/10 proporciona más datos para entrenar, pero deja un conjunto de prueba pequeño, lo que puede hacer que la evaluación sea menos representativa. Por otra parte, la división 60/20/20 permite separar entrenamiento, validación y prueba, pero disminuye considerablemente las muestras destinadas al entrenamiento. Finalmente, la validación cruzada k-fold puede ofrecer una evaluación más robusta, pero requiere entrenar el modelo varias veces, aumentando el tiempo de procesamiento y la complejidad del desarrollo.

e. Resumen del dataset seleccionado

Una vez evaluadas las alternativas relacionadas con el origen, representación, organización y división del dataset, en la **Tabla *42*** se presenta el resumen de la selección realizada.

Tabla 42
Resumen del dataset seleccionado

Elemento	Alternativa seleccionada	Función dentro del sistema
Origen del dataset	Dataset híbrido	Combinar fuentes externas y capturas propias para cubrir las categorías del sistema.

Representación de datos	Puntos característicos de la mano	Representar la posición de la mano y los dedos como entrada para los modelos.
Organización	Categorías y etiquetas	Separar las muestras según el tipo de seña y su etiqueta correspondiente.
División	Entrenamiento 80% y evaluación 20%	Entrenar el modelo y evaluar su desempeño con un conjunto reservado de datos.

Nota: Elaboración propia, 2026

La selección del dataset permite definir la base de datos necesaria para el entrenamiento y evaluación de los modelos de reconocimiento de señas. El origen híbrido facilita cubrir las categorías requeridas por el sistema; la representación mediante puntos característicos reduce la carga computacional; la organización por categorías y etiquetas mantiene el orden de las muestras; y la división entrenamiento/evaluación permite medir el desempeño del modelo con datos reservados para este propósito.

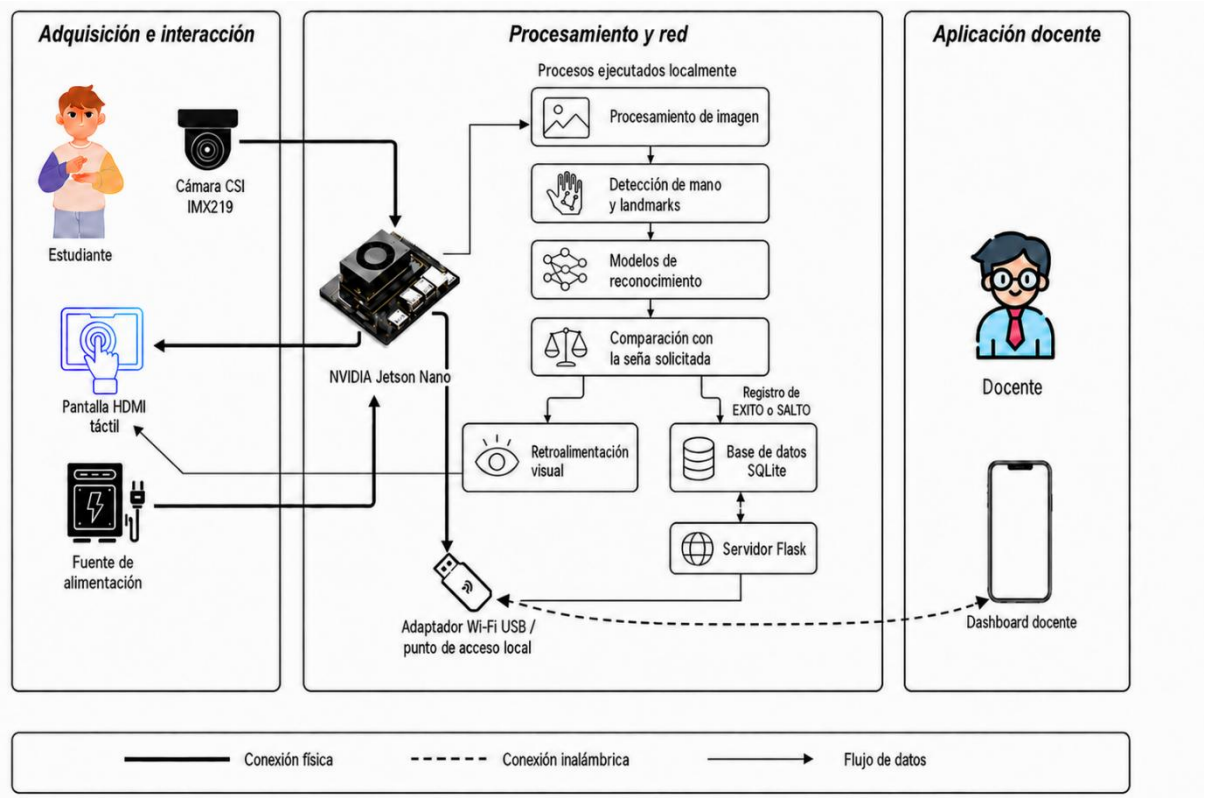
3.3.2 Arquitectura del sistema

La arquitectura del sistema define la forma en que se relacionan los componentes físicos y los procesos necesarios para el funcionamiento de la herramienta didáctica. La NVIDIA Jetson Nano actúa como unidad central, ya que recibe la información de la cámara, ejecuta el reconocimiento de las señas, presenta la retroalimentación visual, almacena los resultados y mantiene disponible el dashboard docente.

Los procesos principales se ejecutan localmente, sin depender de servicios externos ni de una conexión permanente a internet. Para el acceso docente, el adaptador Wi-Fi USB se configura como punto de acceso local, permitiendo que el celular se conecte directamente a la red generada por el prototipo.

La Figura 8 presenta la arquitectura general, organizada en tres áreas: adquisición e interacción, procesamiento y red, y aplicación docente.

Figura 8
Arquitectura general del sistema



Nota. La línea continua representa las conexiones físicas, la línea discontinua corresponde a la comunicación inalámbrica y las flechas indican el flujo de datos. Elaboración propia, 2026.

Como se observa en la **Figura *9*** la cámara captura la seña realizada por el estudiante y envía la información a la Jetson Nano. En esta placa se ejecutan el procesamiento de imagen, la detección de la mano, la extracción de landmarks y los modelos de reconocimiento. El resultado se compara con la seña solicitada para generar la retroalimentación que se presenta en la pantalla táctil.

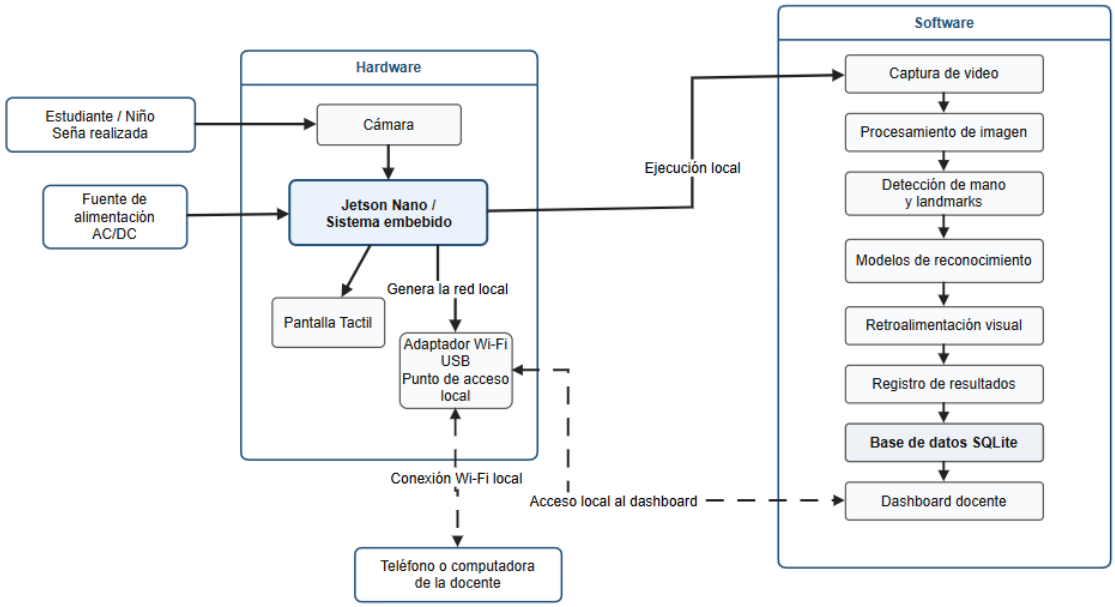
Cuando la actividad finaliza con estado EXITO o SALTO, la información se almacena en la base de datos SQLite. El servidor Flask permite consultar estos registros desde el dashboard, al cual la docente accede mediante un celular conectado al punto de acceso Wi-Fi local. De esta

manera, el reconocimiento, el almacenamiento y la supervisión funcionan dentro del mismo entorno.

Para representar la arquitectura mediante bloques funcionales, la Figura 9 organiza los elementos en hardware y software. Esta representación permite observar la relación entre la interacción del estudiante, el procesamiento local y el acceso docente.

Figura 9

Diagrama de bloques de la arquitectura del sistema



Nota: Elaboración propia, 2026

Como se puede ver en la **Figura *9***, el proceso comienza cuando el estudiante realiza una seña frente a la cámara. Esta seña es capturada y enviada a la Jetson Nano, donde se ejecutan todos los módulos principales del sistema. Primero se realiza la captura del video y el procesamiento de la imagen; después, se detecta la mano y se obtienen los puntos característicos o landmarks. Con esta información, el modelo de reconocimiento analiza la seña y determina cuál fue la realizada por el estudiante.

Una vez obtenida la predicción, el sistema muestra una respuesta visual en la pantalla conectada al prototipo. Esta retroalimentación permite que el estudiante sepa si la seña fue reconocida

correctamente o si necesita volver a intentarlo. De esta manera, el sistema no solo cumple con la tarea de reconocer señas, sino que también sirve como apoyo durante el proceso de aprendizaje.

Al mismo tiempo, los resultados generados durante la sesión se almacenan en una base de datos local SQLite. En ella se registra información como la actividad realizada, la seña solicitada, la seña reconocida por el sistema, la cantidad de intentos, así como los aciertos y errores obtenidos. Estos datos son importantes porque permiten llevar un seguimiento del progreso del estudiante y facilitan la generación de reportes para la docente.

Por otro lado, la supervisión se realiza mediante un dashboard web local al que se puede acceder desde un teléfono, una tableta o una computadora conectada a la misma red. Para lograrlo, la Jetson Nano utiliza un adaptador Wi-Fi USB configurado como punto de acceso local. El celular de la docente se conecta a esta red para acceder al dashboard servido mediante Flask. La comunicación se mantiene dentro de la red local y no requiere un router externo ni acceso a Internet.

Desde el punto de vista físico, la Jetson Nano actúa como el núcleo principal de la arquitectura. Esta recibe energía desde una fuente AC/DC y se conecta con los diferentes dispositivos que forman parte del prototipo. La cámara se encarga de capturar las señas realizadas por el estudiante, la pantalla muestra la interfaz y la retroalimentación visual, la memoria local almacena la información generada y el router permite la comunicación necesaria para el acceso al dashboard. Esta integración hace posible distinguir dos tipos de interacción: la del estudiante con la herramienta de aprendizaje y la de la docente con el sistema de monitoreo.

En resumen, la arquitectura propuesta reúne en un mismo entorno la captura de imágenes, el procesamiento local, el reconocimiento de señas, la retroalimentación al estudiante, el almacenamiento de información y la supervisión docente. Todo esto permite que el sistema

funcione de manera organizada y acorde con los requerimientos definidos, además de facilitar su implementación en un contexto educativo sin depender de plataformas externas.

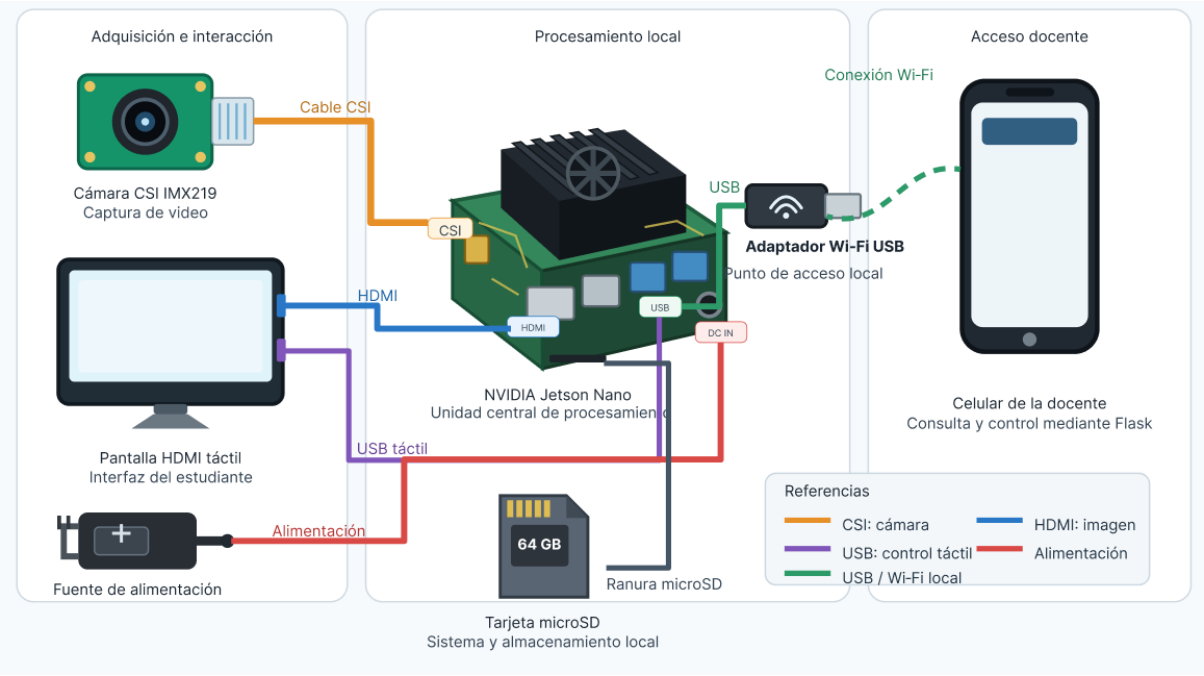
3.3.2.1 Conexiones físicas y comunicación local del prototipo

Después de definir los bloques que conforman la arquitectura, se establecieron las conexiones físicas necesarias para integrar los componentes del prototipo. La NVIDIA Jetson Nano funciona como la unidad central, debido a que recibe la información capturada por la cámara, ejecuta el procesamiento local y permite presentar la interfaz del estudiante y el dashboard docente.

La cámara IMX219 se conecta al puerto CSI de la Jetson Nano para realizar la captura continua de video. La pantalla táctil utiliza una conexión HDMI para mostrar la interfaz y una conexión USB para controlar la función táctil. La fuente de alimentación se conecta directamente a la placa, mientras que la tarjeta microSD contiene el sistema operativo, los modelos y los archivos necesarios para la ejecución.

Para el acceso docente, el adaptador Wi-Fi USB se conecta a uno de los puertos de la Jetson Nano y funciona como punto de acceso local. De esta manera, la docente puede conectarse desde un celular y abrir el dashboard mediante el servidor Flask, sin requerir acceso a Internet. La Figura 10 presenta las conexiones utilizadas en el prototipo.

Figura 10
Conexiones físicas y comunicación local del prototipo



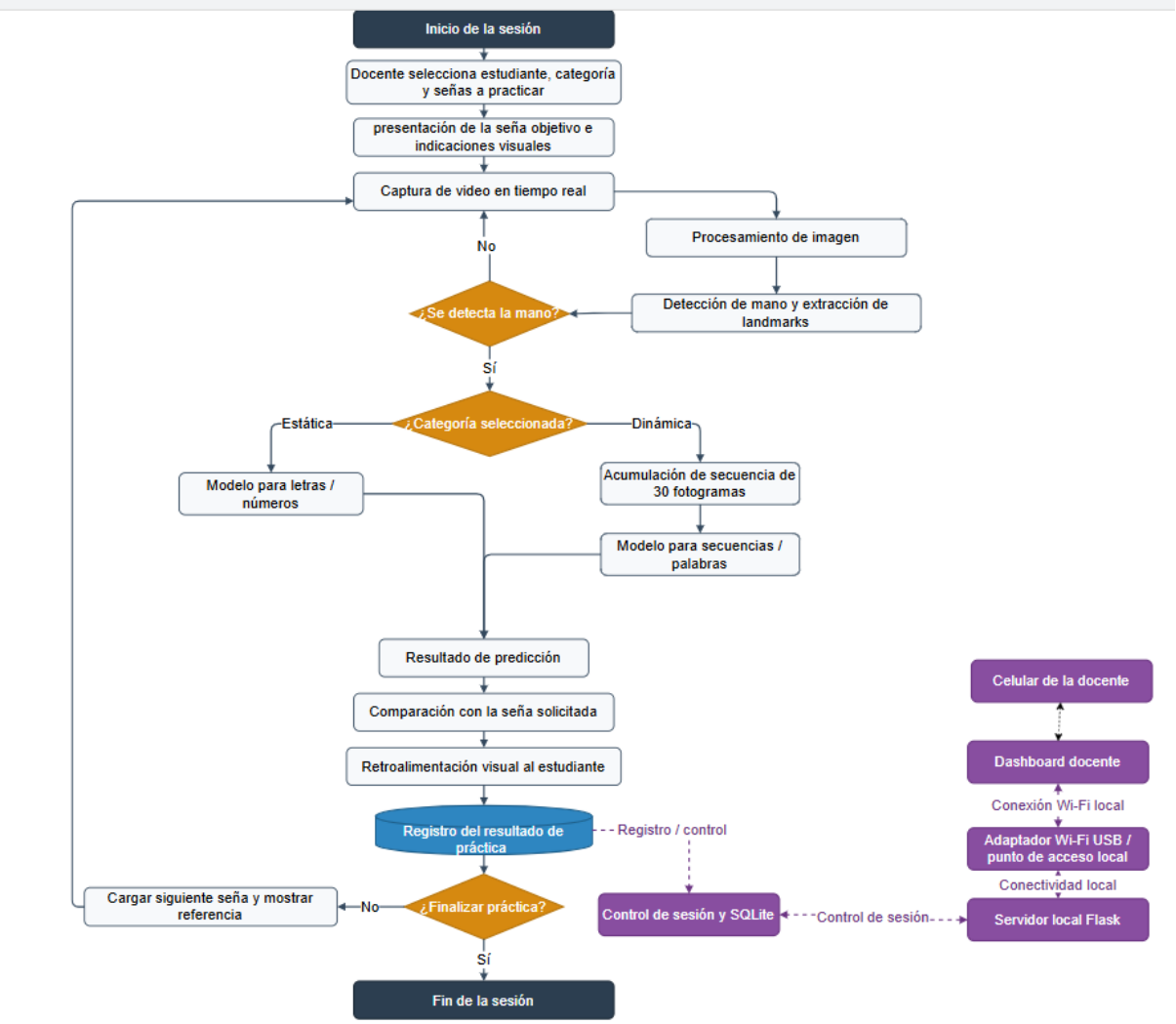
Nota: Elaboración propia, 2026

Como se observa en la Figura 10, las conexiones físicas se concentran en la Jetson Nano. La cámara envía el video para su procesamiento, la pantalla presenta la actividad al estudiante y el adaptador Wi-Fi permite la comunicación con el celular de la docente. Esta distribución mantiene el reconocimiento, el almacenamiento y la comunicación dentro del entorno local.

3.3.2.2 Funcionamiento lógico y conectividad del sistema

El funcionamiento lógico del sistema se organiza como una secuencia de procesamiento local orientada al reconocimiento de señas dactilológicas y al registro de resultados de práctica. Esta secuencia inicia con la configuración de la sesión por parte de la docente y finaliza con el almacenamiento de la información generada y la posibilidad de consultarla mediante el dashboard docente. La **Figura *11*** presenta el flujo lógico de funcionamiento y conectividad del sistema.

Figura 11
Flujo lógico de funcionamiento y conectividad del sistema



Nota: Elaboración propia, 2026

Como se observa en la **Figura *11***, el proceso inicia cuando la docente selecciona el estudiante, la categoría y la seña que será practicada. Posteriormente, el sistema presenta la seña objetivo junto con las indicaciones visuales necesarias para guiar al niño durante la actividad. Esta etapa permite contextualizar la práctica antes de iniciar el reconocimiento automático.

Una vez iniciada la actividad, la cámara captura video en tiempo real y envía los frames hacia la Jetson Nano, donde se ejecuta el procesamiento de imagen. En esta etapa se analiza el flujo visual capturado con el propósito de identificar la presencia de la mano. Si no se detecta una mano dentro del área de captura, el sistema retorna a la etapa de adquisición de video y continúa

esperando una entrada válida. Esta condición evita que el modelo de reconocimiento procese imágenes que no contienen información útil para la clasificación.

Cuando la mano es detectada, el sistema realiza la extracción de landmarks, es decir, puntos característicos que representan la posición de la palma y los dedos. Estos puntos permiten transformar la imagen capturada en datos estructurados, los cuales son utilizados como entrada para el reconocimiento de la seña.

Luego, el sistema verifica la categoría seleccionada para determinar la ruta de análisis correspondiente. Para señas estáticas, como letras y números, los landmarks extraídos son enviados al modelo de reconocimiento correspondiente. En cambio, para señas dinámicas, asociadas a palabras o gestos que dependen del movimiento, el sistema acumula una secuencia de treinta fotogramas antes de enviarla al modelo encargado de analizar secuencias temporales.

Después de ejecutar el modelo correspondiente, el sistema obtiene un resultado de predicción. Este resultado se compara con la seña solicitada al inicio de la actividad. A partir de esta comparación, se genera una retroalimentación visual para el estudiante, indicando si la seña fue reconocida correctamente o si es necesario repetir el intento. Esta respuesta inmediata permite que la herramienta cumpla una función didáctica durante la práctica.

El resultado de la interacción se almacena en la base de datos local SQLite. En este registro se conserva información relacionada con la sesión, la seña practicada, la predicción generada, los intentos realizados y el resultado obtenido. Esta información permite mantener un historial de práctica y sirve como base para el seguimiento docente.

La conectividad del sistema se representa mediante la comunicación entre la Jetson Nano, el servidor local Flask, el router de red local y el dispositivo de la docente. El servidor local permite habilitar el dashboard docente dentro de la misma red, de manera que la docente pueda consultar resultados y enviar comandos de control de sesión desde un teléfono, tableta o

computadora. Esta comunicación se realiza en la red local, sin necesidad de depender de servicios en la nube o conexión permanente a internet.

Finalmente, el sistema evalúa si la práctica ha terminado. Si la sesión continúa, se carga la siguiente seña y se muestra nuevamente la referencia visual para que el estudiante realice un nuevo intento. Si la práctica finaliza, el sistema cierra la sesión y conserva los registros generados para su posterior consulta desde el dashboard docente.

En conjunto, este flujo permite integrar el reconocimiento automático, la retroalimentación al estudiante, el almacenamiento local y la supervisión docente dentro de una misma arquitectura de funcionamiento local.

3.3.2.3 Consideraciones y dimensionamiento energético de operación

El dimensionamiento energético del prototipo permite verificar que la fuente de alimentación seleccionada sea capaz de suministrar la corriente necesaria para el funcionamiento estable de los componentes principales del sistema. Esta consideración es importante debido a que la herramienta ejecuta procesos de captura de video, visión artificial, reconocimiento de señas, visualización de resultados y almacenamiento local, los cuales requieren una alimentación continua y estable.

Aunque el sistema se plantea como una herramienta transportable, esta característica se relaciona con la posibilidad de trasladar el prototipo e instalarlo en diferentes espacios educativos. Sin embargo, el dispositivo no se considera autónomo energéticamente, ya que su operación depende de una fuente externa AC/DC conectada a la red eléctrica. Por esta razón, se realiza una estimación de consumo que permite justificar la selección de una fuente de 5V–4A frente a alternativas como una batería portátil convencional.

Para el cálculo se consideran los componentes principales que intervienen durante una sesión de uso: placa embebida, cámara, pantalla, ventilador y periféricos menores. La **Tabla *43*** presenta la estimación de corriente y potencia requerida por cada elemento.

Tabla 43
Estimación de consumo eléctrico de los componentes principales del sistema

Componente	Voltaje de operación	Corriente estimada	Potencia estimada
NVIDIA Jetson Nano	5 V	2,00 A	10,00 W
Cámara IMX219 CSI	5 V	0,20 A	1,00 W
Pantalla HDMI	5 V	1,00 A	5,00 W
Ventilador de enfriamiento	5 V	0,10 A	0,50 W
Periféricos de red	5 V	0,25 A	1,25 W
Total, estimado	**5 V**	**3,55 A**	**17,75 W**

Nota: Los valores nominales corresponden al escenario de máxima exigencia de hardware, validados con las hojas de datos oficiales (datasheets) de los respectivos fabricantes. Las mediciones en estado de reposo serán inferiores a esta estimación.

La corriente total estimada se obtiene mediante la sumatoria de las corrientes individuales de los componentes, como se muestra en la **Ecuación 1**.

Ecuación 1
Corriente total estimada

$$[I_T = \sum I_i]$$

Donde:

$$[I_T = \text{corriente total estimada del sistema}]$$

$$[I_i = \text{corriente individual de cada componente}]$$

Aplicando la ecuación:

$$[I_T = 2{,}00A + 0{,}20A + 1{,}00A + 0{,}10A + 0{,}25A]$$

$$[I_T = 3{,}55A]$$

Con base en este resultado, el sistema requiere una corriente aproximada de 3,55 A a 5 V durante su funcionamiento. La potencia eléctrica requerida se obtiene mediante la relación entre voltaje y corriente, expresada en la **Ecuación *2***.

Ecuación 2
Potencia eléctrica

$$[P = V \cdot I]$$

Donde:

$$[P = potencia\ eléctrica]$$

$$[V = \text{voltaje de operación}]$$

$$[I = \text{corriente eléctrica}]$$

Para el sistema:

$$[P_{sistema} = 5V \cdot 3{,}55A]$$

$$[P_{sistema} = 17{,}75W]$$

La fuente seleccionada entrega 5V–4A, por lo que su potencia máxima disponible es:

$$[P_{fuente} = 5V \cdot 4A]$$

$$[P_{fuente} = 20W]$$

A partir de estos valores, se determina el margen de potencia disponible:

$$[P_{margen} = P_{fuente} - P_{sistema}]$$

$$[P_{margen} = 20W - 17{,}75W]$$

$$[P_{margen} = 2{,}25W]$$

El margen obtenido permite justificar la selección de la fuente AC/DC de 5V–4A, ya que su capacidad es superior al consumo estimado del sistema. Este margen contribuye a reducir el riesgo de caídas de tensión, reinicios o interrupciones durante la ejecución de los procesos de visión artificial y reconocimiento de señas.

Además, se analiza la alternativa de alimentación mediante batería portátil. Una batería convencional puede ofrecer capacidad energética suficiente en términos de autonomía; sin embargo, no siempre entrega la corriente requerida de forma estable. Por ejemplo, una batería con salida de 5V–2A solo puede entregar una potencia máxima de:

$$[P = 5V \cdot 2A]$$

$$[P = 10W]$$

Mientras que una batería con salida de 5V–3A entrega:

$$[P = 5V \cdot 3A]$$

$$[P = 15W]$$

Estos valores se encuentran por debajo de la potencia estimada del sistema, correspondiente a 17,75 W. Por lo tanto, una batería portátil convencional de 5V–2A o 5V–3A no sería suficiente para alimentar de forma estable el conjunto de componentes considerados.

Para estimar la autonomía teórica de una batería portátil, se puede considerar una batería comercial de 10000 mAh. Debido a que la capacidad nominal de este tipo de baterías suele expresarse respecto al voltaje interno de sus celdas, se calcula primero la energía almacenada mediante la **Ecuación *3***.

Ecuación 3
Energía nominal de una batería

$$[E = V_{bat} \cdot C]$$

Donde:

$$[E = energía\ almacenada\ en\ Wh]$$

$$[V_{bat} = \text{voltaje nominal interno de la batería}]$$

$$[C = \text{capacidad de la batería en Ah}]$$

Para una batería de 10000 mAh, equivalente a 10 Ah, con voltaje nominal interno de 3,7 V:

$$[E = 3{,}7V \cdot 10Ah]$$

$$[E = 37Wh]$$

Considerando una eficiencia aproximada del 85 % en la conversión hacia 5 V, la energía útil sería:

$$[E_{útil} = 37Wh \cdot 0{,}85]$$

$$[E_{útil} = 31{,}45Wh]$$

El tiempo de autonomía teórica se obtiene mediante la **Ecuación *4***.

Ecuación 4
Tiempo estimado de autonomía

$$[T = \frac{E_{útil}}{P_{sistema}}]$$

$$\left[T = \frac{31{,}45Wh}{17{,}75W}\right]$$

$$[T = 1{,}77h0]$$

l cálculo anterior muestra que una batería de 10000 mAh podría ofrecer una autonomía teórica aproximada de **1,77 horas**, siempre que sea capaz de entregar la corriente requerida por el sistema. Sin embargo, el aspecto crítico no es únicamente la capacidad de la batería, sino su corriente máxima de salida. Si la batería no entrega al menos la corriente estimada de **3,55 A** de forma estable, el prototipo podría presentar reinicios, fallos de captura o interrupciones durante el procesamiento.

Por esta razón, para el prototipo actual se prioriza la alimentación mediante una fuente externa AC/DC de **5V–4A**. Esta decisión permite mantener una operación más estable durante las sesiones de práctica y evita confundir la transportabilidad física del sistema con autonomía energética. En consecuencia, el prototipo se define como una herramienta transportable de funcionamiento local, alimentada mediante fuente externa, dejando la incorporación de una batería portátil de alta corriente como una posible mejora futura.

3.3.3 Diseño del dataset

El diseño del dataset define la forma en que se organizan, etiquetan y representan las muestras utilizadas para el reconocimiento de señas dactilológicas dentro del sistema. Esta etapa permite estructurar los datos que serán empleados por los modelos de inteligencia artificial, diferenciando entre señas estáticas, como letras y números, y señas dinámicas, asociadas a palabras o gestos que requieren movimiento.

A partir de la selección realizada previamente, el dataset se diseñó bajo un enfoque híbrido, combinando muestras obtenidas de fuentes externas con capturas propias necesarias para adaptar el sistema a las categorías consideradas en el prototipo. Esta organización permite

disponer de datos para el entrenamiento de los modelos y mantener una estructura flexible para futuras ampliaciones.

3.3.3.1 Estructura general del dataset

El dataset se organiza por categorías y etiquetas. Las categorías permiten diferenciar el tipo de seña que será reconocida, mientras que las etiquetas identifican cada clase específica dentro del conjunto de datos.

Las señas estáticas se representan a partir de un frame o imagen individual, debido a que su reconocimiento depende principalmente de la configuración de la mano. En cambio, las señas dinámicas se representan mediante secuencias de frames, ya que requieren conservar información del movimiento realizado durante la seña, tal como se puede observar en la **Tabla *44***.

Tabla 44
Estructura general del dataset

Categoría	Tipo de seña	Representación	Modelo asociado
Letras	Estática	Landmarks de un frame	Modelo estático
Números	Estática	Landmarks de un frame	Modelo estático
Palabras seleccionadas	Dinámica	Secuencias de landmarks	Modelo dinámico

Nota. Elaboración propia, 2026.

En su organización inicial, los datos originales se almacenan en carpetas organizadas por categoría y clase. El directorio raíz se divide en tres categorías principales: abecedario, números y palabras. Dentro de cada categoría se ubican subcarpetas correspondientes a cada clase o etiqueta.

La estructura general del dataset en crudo se organiza de la siguiente manera:

```
dataset/
├── DATASET_ABECEDARIO/
│   ├── A/
```

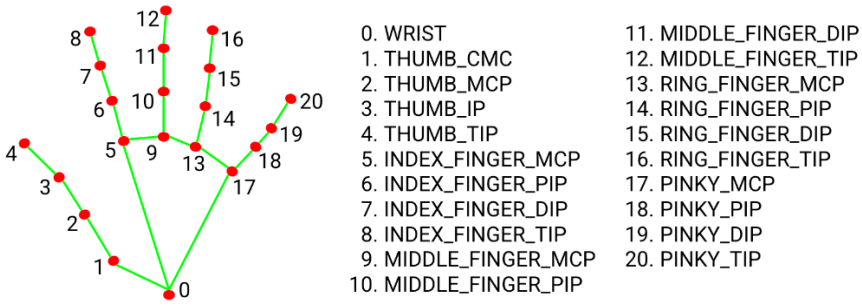
```
    ├── B/
    ├── C/
    └── ...
├── DATASET_NUMEROS/
    ├── 0/
    ├── 1/
    ├── 2/
    └── ...
└── DATASET_PALABRAS/
    ├── ADIOS/
        ├── 0.jpg
        ├── 1.jpg
        ├── ...
        └── 29.jpg
    ├── HOLA/
        ├── 0.jpg
        ├── 1.jpg
        ├── ...
        └── 29.jpg
    └── ...
```

En esta estructura, DATASET_ABECEDARIO almacena las muestras correspondientes a letras dactilológicas, DATASET_NUMEROS contiene las muestras correspondientes a números y DATASET_PALABRAS almacena las secuencias de imágenes asociadas a señas dinámicas. Para las palabras, los frames se enumeran de forma secuencial, por ejemplo desde 0.jpg hasta 29.jpg, con el fin de conservar el orden temporal del movimiento.

3.3.3.2 Representación mediante landmarks

Para reducir la complejidad del procesamiento, las muestras no se trabajan únicamente como imágenes completas, sino como puntos característicos de la mano o landmarks. En este sistema, cada mano detectada se representa mediante 21 puntos clave, los cuales describen la ubicación de la palma y las articulaciones de los dedos, los puntos se pueden observar en la

Figura 12
Representación de los 21 landmarks



Nota: Tomado de MediaPipe. (s. f.). *21 hand landmarks*

La representación mediante 21 landmarks se adopta porque corresponde a la estructura generada por MediaPipe Hands para describir la mano. Estos puntos permiten identificar zonas relevantes como la palma, las articulaciones y las puntas de los dedos. De esta manera, cada seña puede representarse mediante información geométrica de la mano, sin necesidad de procesar directamente toda la imagen.

Cada landmark contiene coordenadas X, Y y Z. Las coordenadas X e Y representan la posición del punto dentro del plano de la imagen, mientras que Z proporciona una referencia relativa de profundidad. De esta forma, cada frame queda representado por 63 valores numéricos, considerando los 21 landmarks y sus tres coordenadas.

Esta representación permite trabajar con información estructurada y de menor dimensión, reduciendo la influencia de elementos externos como el fondo, la iluminación o el tamaño de la imagen. Por esta razón, los landmarks constituyen la entrada principal para los modelos de reconocimiento.

3.3.3.3 Organización procesada para señas estáticas y dinámicas

Después de organizar los datos originales, las imágenes o frames son procesados mediante MediaPipe Hands. Esta herramienta permite detectar la mano y extraer los 21 landmarks por cada imagen. A partir de esta transformación, el modelo no trabaja directamente con los píxeles de la imagen, sino con una representación numérica de la posición de la mano, la palma y los dedos.

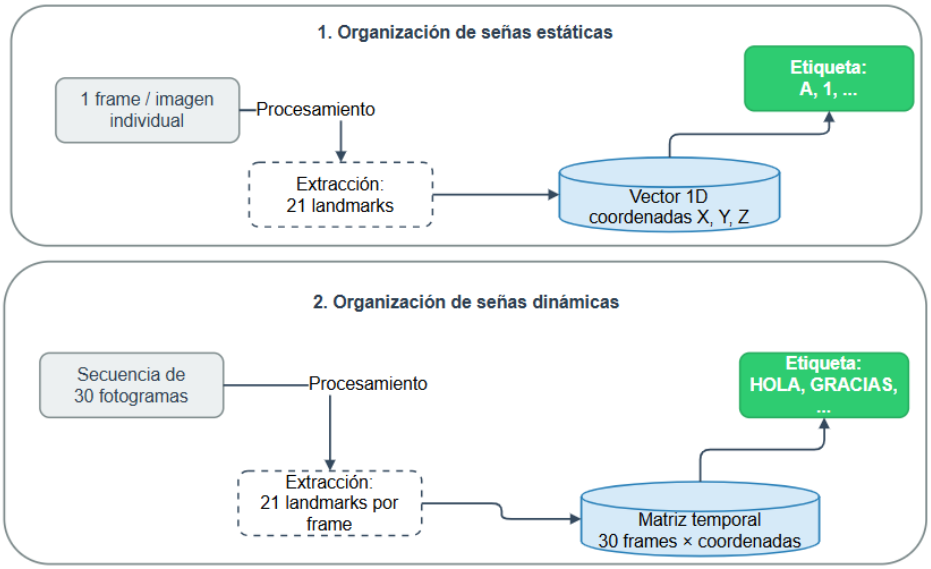
Debido a que el sistema reconoce señas estáticas y dinámicas, el dataset procesado se organiza en dos estructuras de datos. Para las señas estáticas, correspondientes a letras y números, cada muestra se representa como un vector unidimensional de 63 valores numéricos. La forma general del arreglo es:

(Total_de_muestras, 63)

Esta estructura se utiliza porque el modelo estático solo necesita analizar una posición de la mano en un instante determinado. Por ello, resulta adecuada para modelos de clasificación de señas estáticas, como redes neuronales densas o MLP.

La **Figura *13*** presenta la organización general de las muestras estáticas y dinámicas del dataset.

Figura 13
Organización de datos para señas estáticas y dinámicas



Nota. Elaboración propia, 2026.

Para las señas dinámicas, correspondientes a palabras seleccionadas, cada muestra se organiza como una secuencia temporal de landmarks obtenida de varios frames consecutivos. En este caso, se utilizan secuencias de 30 frames, donde cada frame contiene 63 valores numéricos. La forma general del arreglo es:

(Total_de_muestras, 30, 63)

Esta estructura permite conservar la evolución temporal del movimiento y resulta adecuada para modelos recurrentes, como LSTM, debido a que este tipo de red analiza la variación de los datos a lo largo del tiempo.

Para facilitar la carga de datos durante el entrenamiento y optimizar el almacenamiento local, las muestras procesadas pueden organizarse como arreglos numéricos y almacenarse en formatos serializados, como .pickle o .npy, de acuerdo con la estructura usada en la implementación.

3.3.3.4 Etiquetado de las muestras

El etiquetado permite asociar cada muestra del dataset con la seña que representa. Para letras y números, cada etiqueta corresponde a una clase individual; mientras que, para palabras dinámicas, la etiqueta representa el gesto completo que se desea reconocer.

Junto con los arreglos numéricos, el sistema conserva archivos de etiquetas que permiten relacionar cada muestra con su clase correspondiente. Estas etiquetas convierten los nombres de las carpetas en valores numéricos, debido a que las redes neuronales trabajan con datos numéricos durante el entrenamiento.

Por ejemplo, una etiqueta puede asignar la clase ADIOS al valor 0, la clase HOLA al valor 1 y así sucesivamente. Estos mapeos pueden almacenarse en archivos como etiquetas_palabras.pickle, permitiendo que el sistema identifique la correspondencia entre la salida numérica del modelo y la seña reconocida.

Para que estas clases puedan ser procesadas por los modelos de inteligencia artificial, las etiquetas textuales se transforman a una representación numérica. En este proceso se puede emplear una codificación vectorial, como One-Hot Encoding, lo que permite que cada clase sea interpretada como una salida diferenciada durante el entrenamiento.

Una organización clara de etiquetas evita errores de clasificación, duplicidad de nombres y confusiones entre clases. Además, facilita la incorporación de nuevas señas en futuras versiones del sistema.

3.3.3.1 División del dataset

Para el entrenamiento y evaluación de los modelos, el dataset se divide en dos subconjuntos principales. El conjunto de entrenamiento se utiliza para ajustar los modelos de

reconocimiento, mientras que el conjunto de evaluación permite revisar su comportamiento con muestras no utilizadas directamente durante el entrenamiento.

La división aplicada corresponde a 80 % para entrenamiento y 20 % para evaluación, como se puede observar en la **Tabla *45***.

Tabla 45
División del dataset

Subconjunto	Porcentaje	Finalidad
Entrenamiento	80%	Ajustar los modelos de reconocimiento.
Evaluación	20%	Verificar el comportamiento del modelo con datos no usados en el entrenamiento.

Nota. Elaboración propia, 2026.

3.3.4 Diseño del modelo de reconocimiento

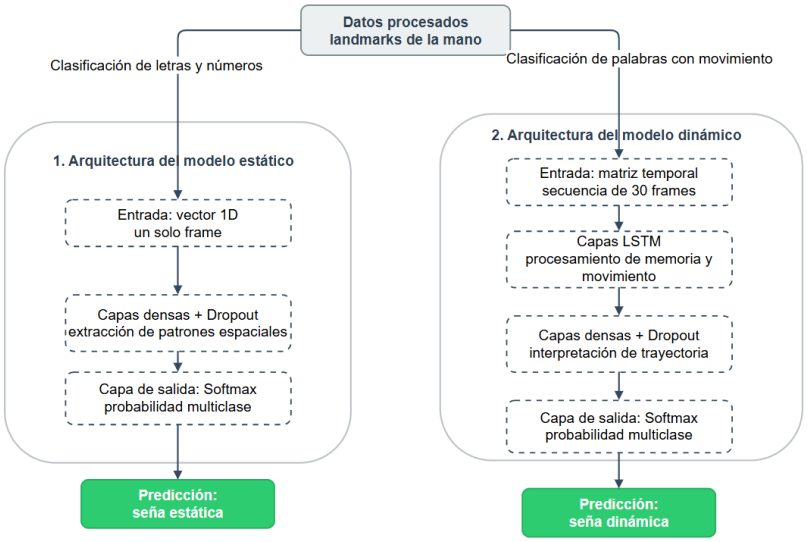
El diseño del modelo de reconocimiento define la forma en que los datos procesados del dataset son utilizados para clasificar las señas dactilológicas dentro del sistema. Debido a que el prototipo considera señas estáticas y dinámicas, se plantea una arquitectura de reconocimiento dividida en dos rutas: una orientada al reconocimiento de letras y números mediante landmarks obtenidos de un solo frame, y otra orientada al reconocimiento de palabras o gestos dinámicos mediante secuencias temporales de landmarks.

Esta separación permite que cada modelo reciba una entrada adecuada según el tipo de seña. En el caso de las señas estáticas, el reconocimiento depende principalmente de la configuración de la mano en una posición determinada. En cambio, para las señas dinámicas, el reconocimiento depende del movimiento realizado durante varios frames consecutivos.

La **Figura *14*** presenta el esquema general del diseño de los modelos de reconocimiento.

Figura 14

Diseño general de los modelos de reconocimiento de señas



Nota. Elaboración propia, 2026.

3.3.4.1 Modelo para señas estáticas

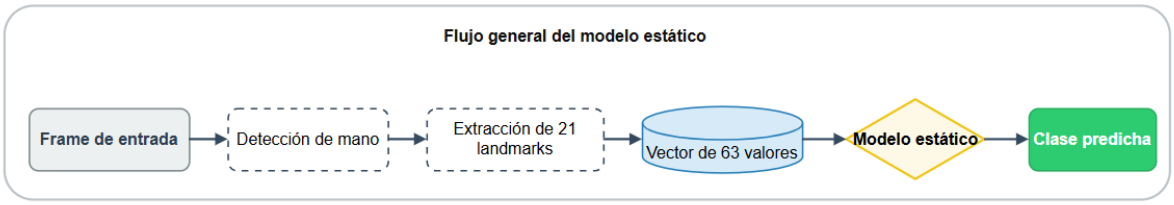
El modelo para señas estáticas está orientado al reconocimiento de letras y números. Para este tipo de señas, cada muestra se representa mediante los 21 landmarks de la mano detectada, considerando las coordenadas X, Y y Z de cada punto. De esta manera, cada frame se transforma en un vector de 63 valores numéricos, que funciona como entrada del modelo.

La finalidad de este modelo es clasificar la configuración de la mano en una de las clases definidas dentro del dataset. Por esta razón, las etiquetas textuales asociadas a cada letra o número se transforman previamente en una representación numérica, permitiendo que el modelo pueda procesarlas durante el entrenamiento.

El diseño del modelo estático se basa en una red neuronal orientada a datos estructurados, debido a que la entrada no corresponde a una imagen completa, sino a un conjunto de coordenadas obtenidas mediante landmarks. Esta decisión permite reducir la carga de procesamiento y facilita su integración en un sistema embebido con ejecución local.

El flujo general del modelo estático se puede observar en la **Figura *15***:

Figura 15
Flujo general del modelo estático



Nota. Elaboración propia, 2026.

3.3.4.2 Modelo para señas dinámicas

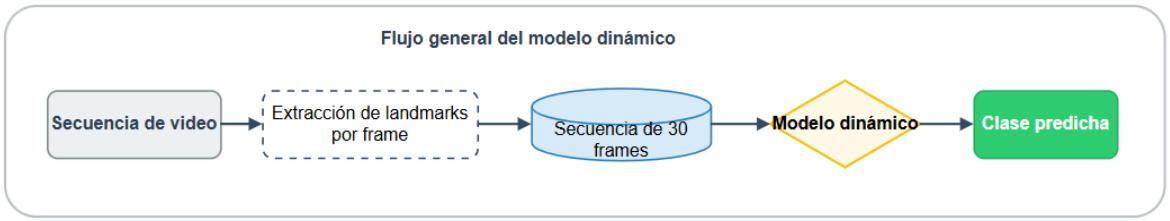
El modelo para señas dinámicas está orientado al reconocimiento de palabras o gestos que requieren movimiento. A diferencia del modelo estático, este no analiza un solo frame, sino una secuencia de landmarks obtenida a partir de varios frames consecutivos.

Para mantener una estructura uniforme, cada muestra dinámica se organiza en secuencias de 30 frames. En cada frame se extraen 21 landmarks con coordenadas X, Y y Z, por lo que cada secuencia conserva información espacial y temporal del movimiento realizado por la mano.

El diseño del modelo dinámico se basa en el análisis de secuencias temporales, debido a que el orden de los frames es importante para interpretar correctamente el gesto. Por esta razón, se considera una arquitectura capaz de procesar información secuencial, como una red recurrente de tipo LSTM, la cual permite analizar la evolución del movimiento a lo largo del tiempo.

En la **Figura *16*** se puede observar el flujo general del modelo dinámico.

Figura 16
Flujo general del modelo dinámico



Nota. Elaboración propia, 2026.

3.3.4.3 Consideraciones del diseño del modelo

El diseño de los modelos mantiene coherencia con el uso de landmarks como representación principal de entrada. Esta decisión permite reducir la cantidad de información procesada en comparación con el uso directo de imágenes completas, favoreciendo la ejecución local en la Jetson Nano.

En esta sección se describe el diseño conceptual de los modelos y su relación con los datos de entrada. Los detalles relacionados con la implementación del entrenamiento, parámetros utilizados, número de épocas, métricas de precisión, pérdidas o resultados obtenidos corresponden a las secciones posteriores de implementación y pruebas.

3.3.5 Diseño de la base de datos

El diseño de la base de datos define la estructura utilizada para almacenar la información generada durante el uso de la herramienta didáctica. Esta información permite registrar estudiantes, categorías de práctica, señas, sesiones, intentos y resultados obtenidos durante el reconocimiento. Además, sirve como base para la consulta de información desde el dashboard docente y para la generación de reportes de seguimiento.

Para el prototipo se considera una base de datos local en SQLite, debido a que el sistema funciona dentro de la Jetson Nano y no depende de servicios externos o plataformas en la nube. Esta característica mantiene coherencia con la arquitectura local del sistema, ya que los datos se almacenan en el mismo entorno donde se ejecutan los procesos de reconocimiento y supervisión.

3.3.5.1 Entidades principales

Las entidades principales consideradas para el diseño de la base de datos son: docente, estudiante, categoría, seña, sesión de práctica e intento, tal como se ve en la Tabla *46*. Cada una

cumple una función específica dentro del sistema y permite mantener organizada la información registrada.

Tabla 46
Entidades principales de la base de datos

Entidad	Descripción	Información principal almacenada
Docente	Usuario encargado de supervisar las prácticas.	Identificación del docente y credenciales de acceso al dashboard.
Estudiante	Niño que realiza las actividades de práctica.	Nombre, edad, e identificador del estudiante.
Categoría	Agrupación del tipo de práctica.	Letras, números o palabras seleccionadas.
Seña	Clase específica que debe ser reconocida.	Nombre de la seña, tipo de seña y categoría asociada.
Sesión de práctica	Registro de una actividad realizada por un estudiante.	Fecha, estudiante, categoría y estado de la sesión.
Intento	Registro individual de cada respuesta del estudiante.	Seña esperada, seña reconocida, acierto, error y número de intento.

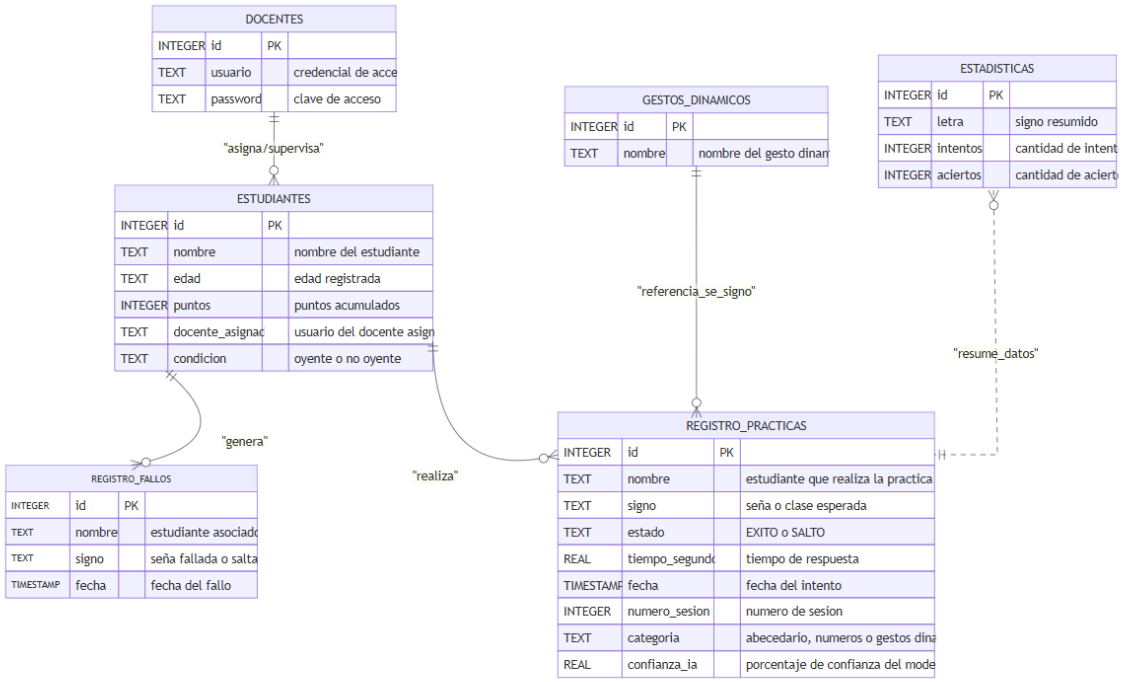
Nota. Elaboración propia, 2026

3.3.5.2 Modelo entidad-relación

La base de datos se diseña mediante relaciones que permiten vincular la información registrada durante el uso del sistema. Un docente puede supervisar a varios estudiantes, mientras que cada estudiante puede tener varias sesiones de práctica. A su vez, cada sesión puede contener varios intentos, dependiendo del número de señas trabajadas o de repeticiones realizadas durante la actividad.

Cada seña pertenece a una categoría determinada, lo que permite organizar el contenido en letras, números o palabras seleccionadas. Además, cada intento se relaciona con una seña esperada y con la predicción generada por el sistema, permitiendo identificar si la respuesta fue correcta o incorrecta. La **Figura *17*** presenta el diagrama entidad-relación propuesto para la base de datos del sistema.

Figura 17
Diagrama entidad-relación de la base de datos



Nota. Elaboración propia, 2026.

Para la correcta interpretación del diagrama entidad-relación, se emplea la notación estándar de bases de datos relacionales. Las siglas PK (*Primary Key* o Llave Primaria) indican el atributo principal que identifica de manera única e irrepetible a cada registro dentro de una tabla. Por otro lado, las siglas FK (*Foreign Key* o Llave Foránea) señalan los atributos que actúan como puntos de conexión para relacionar la información entre diferentes tablas. Finalmente, las conexiones etiquetadas como 1..N representan una cardinalidad de "uno a muchos", estableciendo que un único registro de una entidad (por ejemplo, un estudiante) puede estar vinculado a múltiples registros en otra entidad (por ejemplo, varias sesiones de práctica).

3.3.5.3 Datos registrados durante una sesión

Durante una sesión de práctica, el sistema registra los datos necesarios para realizar el seguimiento del desempeño del estudiante, que se detallan en la **Tabla *47***. Esta información

permite conocer qué seña fue solicitada, cuál fue la respuesta reconocida por el modelo y si el intento fue correcto o incorrecto.

Tabla 47
Datos registrados durante una sesión de práctica

Dato registrado	Finalidad
Identificador del estudiante	Relacionar la práctica con el niño que realizó la actividad.
Categoría seleccionada	Identificar si la práctica corresponde a letras, números o palabras.
Seña solicitada	Registrar la seña que el sistema pidió realizar.
Seña reconocida	Guardar la predicción generada por el modelo.
Resultado del intento	Determinar si la seña fue correcta o incorrecta.
Número de intento	Controlar la cantidad de repeticiones realizadas.
Numero de Sesión	Mantener un historial cronológico de las prácticas.

Nota. Elaboración propia, 2026.

3.3.5.4 Relación con el dashboard y reportes

La información almacenada en SQLite es utilizada por el dashboard docente para presentar los resultados de práctica. Desde este panel, la docente puede consultar el desempeño del estudiante, revisar intentos, aciertos, errores y observar el avance por categoría o por seña.

Los reportes se generan a partir de los datos registrados en la base de datos local. Para ello, el sistema organiza la información almacenada y permite exportarla en formato CSV, facilitando su revisión posterior. Esta relación entre base de datos, dashboard y reportes permite que el sistema no solo reconozca señas, sino que también proporcione una herramienta de seguimiento pedagógico.

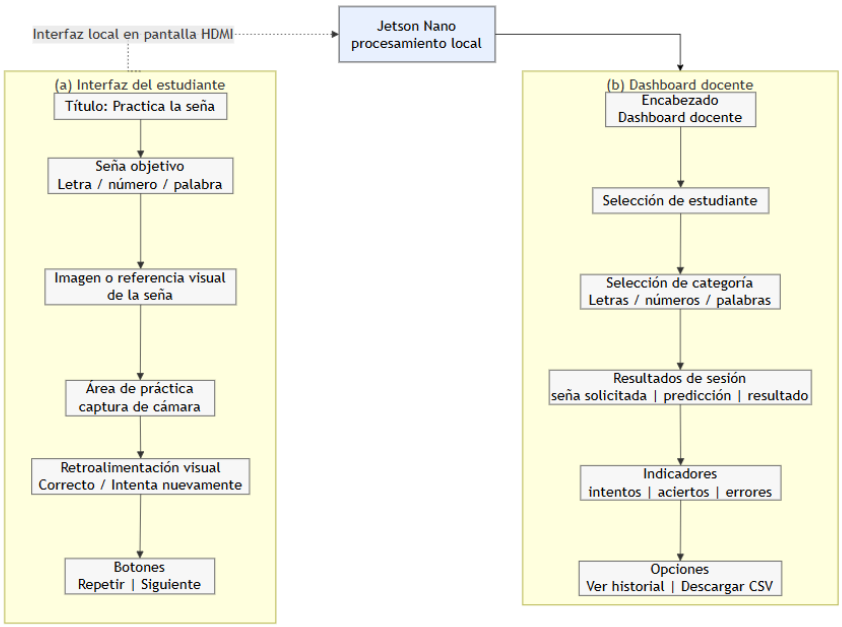
3.3.6 Diseño de interfaces

El diseño de interfaces define la forma en que los usuarios interactúan con la herramienta didáctica. En el sistema propuesto se consideran dos interfaces principales: la interfaz visual del estudiante y el dashboard docente. La primera está orientada a niños de 6 a 8 años y permite presentar las actividades de práctica, la seña objetivo y la retroalimentación visual. La segunda está orientada a la docente y permite configurar sesiones, consultar resultados y realizar seguimiento del desempeño.

El diseño de ambas interfaces mantiene relación directa con la arquitectura del sistema, debido a que la interfaz del estudiante se ejecuta localmente en la pantalla conectada a la Jetson Nano, mientras que el dashboard docente se accede mediante red local desde un teléfono, tableta o computadora. Esta separación permite que el estudiante interactúe con una interfaz sencilla y visual, mientras la docente supervisa el proceso desde un entorno independiente.

La **Figura *18*** presenta el boceto general de las interfaces consideradas para el sistema. En la parte (a) se muestra la distribución propuesta para la interfaz del estudiante, orientada a la presentación de la seña objetivo, el área de práctica y la retroalimentación visual. En la parte (b) se muestra la distribución propuesta para el dashboard docente, orientado a la selección de estudiantes, consulta de resultados y generación de reportes.

Figura 18
Boceto de interfaces del sistema: (a) interfaz del estudiante y (b) dashboard docente



Como se observa en la **Figura *18***, la interfaz del estudiante se plantea con una organización visual simple, priorizando elementos comprensibles para niños de 6 a 8 años. Por otra parte, el dashboard docente se estructura con componentes de consulta y seguimiento, permitiendo separar las funciones de práctica del estudiante y las funciones de supervisión docente

3.3.6.1 Interfaz del estudiante

La interfaz del estudiante está diseñada para facilitar la práctica de señas dactilológicas mediante recursos visuales claros y una interacción sencilla. Debido a que el público objetivo corresponde a niños de 6 a 8 años, la interfaz debe evitar la sobrecarga de información y priorizar elementos gráficos comprensibles, instrucciones breves y retroalimentación inmediata.

Durante una sesión de práctica, la interfaz presenta la seña objetivo que el estudiante debe realizar. Posteriormente, el sistema activa la cámara para capturar el gesto y procesarlo

mediante el modelo de reconocimiento. Una vez obtenida la predicción, la interfaz muestra una respuesta visual que indica si la seña fue reconocida correctamente o si es necesario repetir el intento

La **Tabla *48*** resume los elementos principales de la interfaz del estudiante.

Tabla 48
Elementos considerados en la interfaz del estudiante

Elemento	Función dentro de la interfaz
Seña objetivo	Presenta la letra, número o palabra que el estudiante debe practicar.
Indicaciones visuales	Orientan al estudiante antes y durante la práctica.
Área de práctica	Permite relacionar la actividad con la captura realizada por la cámara.
Retroalimentación visual	Informa si la seña fue reconocida correctamente o si debe repetirse.
Avance de actividad	Permite continuar con la siguiente seña o repetir el intento.

Nota. Elaboración propia, 2026.

3.3.6.2 Dashboard docente

El dashboard docente está diseñado como una interfaz web local que permite supervisar el uso de la herramienta didáctica. Esta interfaz se accede desde un dispositivo conectado a la misma red local del sistema, sin depender de servicios externos o plataformas en la nube.

Desde el dashboard, la docente puede seleccionar al estudiante, definir la categoría de práctica y revisar los resultados registrados durante las sesiones. Además, el panel permite consultar información almacenada en la base de datos SQLite, como intentos, aciertos, errores, señas practicadas y desempeño por actividad.

La **Tabla *49*** presenta los elementos considerados para el dashboard docente.

Tabla 49
Elementos considerados en el dashboard docente

Elemento	Función dentro del dashboard

Selección de estudiante	Permite asociar la práctica con un niño específico.
Selección de categoría	Permite definir si la práctica corresponde a letras, números o palabras.
Resultados por sesión	Presenta la información registrada durante una práctica.
Indicadores de desempeño	Muestra intentos, aciertos y errores del estudiante.
Reportes	Permite exportar información para seguimiento docente.

Nota. Elaboración propia, 2026.

3.3.6.3 Diseño de reportes

El diseño de reportes se relaciona directamente con la base de datos local y el dashboard docente. Los reportes se generan a partir de la información almacenada durante las sesiones de práctica, permitiendo conservar evidencia del desempeño del estudiante y facilitar el seguimiento pedagógico.

El formato considerado para la exportación de reportes es CSV, debido a que permite almacenar los datos de forma estructurada y abrirlos posteriormente en herramientas de análisis o procesamiento de hojas de cálculo. Esta característica facilita que la docente revise los resultados fuera del sistema, filtre información por estudiante o categoría, y conserve un registro del avance.

La **Tabla 50** resume los datos considerados para los reportes.

Tabla 50
Datos considerados para los reportes de seguimiento

Dato	Finalidad
Estudiante	Identificar a quién pertenece el registro.
Fecha de sesión	Ordenar cronológicamente las prácticas realizadas.
Categoría	Diferenciar letras, números o palabras.
Seña solicitada	Conocer la actividad propuesta por el sistema.
Seña reconocida	Registrar la predicción generada por el modelo.

Número de intentos	Identificar repeticiones realizadas durante la práctica.
Resultado	Determinar si la respuesta fue correcta o incorrecta.

Nota. Elaboración propia, 2026.

3.4 Fase de Implementación

La fase de implementación comprende la construcción e integración de los componentes de hardware y software definidos durante la etapa de diseño. En esta fase se realiza la configuración del entorno de trabajo, la conexión de los dispositivos físicos, la preparación de los datos, el entrenamiento de los modelos de reconocimiento, la creación de interfaces y la integración del sistema con la base de datos y el dashboard docente.

A diferencia de la fase de diseño, donde se describe la organización conceptual del sistema, en la implementación se detallan las actividades realizadas para poner en funcionamiento el prototipo. Además, durante esta fase se ejecutan verificaciones técnicas preliminares por módulo, con el propósito de comprobar que cada componente funcione correctamente antes de realizar las pruebas finales del sistema integrado.

3.4.1 Implementación del entorno de hardware

El ensamblaje del prototipo se realizó siguiendo la arquitectura física previamente diseñada, centralizando todas las conexiones en la placa NVIDIA Jetson Nano. El proceso de integración se dividió en la conexión de periféricos de entrada/salida y el acoplamiento en el envolvente mecánico lúdico.

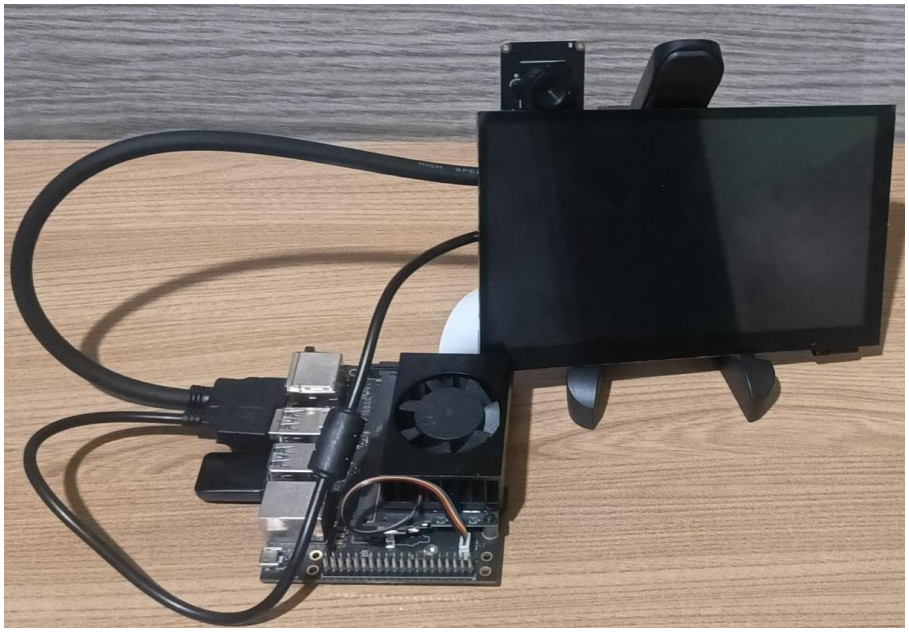
3.4.1.1 Conexión de componentes electrónicos

En primer lugar, se estableció la interfaz de captura visual conectando la cámara IMX219 directamente al puerto CSI (*Camera Serial Interface*) de la placa base mediante un cable plano

flexible. Esta conexión directa es vital para habilitar el flujo de video sin ocupar los puertos USB del sistema.

A continuación, se integró la pantalla táctil de 7 pulgadas. La transmisión de video se estableció mediante el puerto HDMI, mientras que la interfaz táctil se habilitó conectando la pantalla a uno de los puertos USB de la Jetson Nano, como se puede observar en la **Figura 19**. Finalmente, se conectó el módulo Wi-Fi para la conexion y se acopló la fuente de alimentación externa (5V - 4A) para asegurar la energía del sistema.

Figura 19
Conexión física de los componentes electrónicos principales



Nota. Elaboración propia, 2026.

3.4.1.2 Integración en el envolvente mecánico

Para proteger los circuitos y adaptar el prototipo al entorno infantil, todos los componentes electrónicos fueron instalados dentro de una carcasa de madera diseñada y fabricada con forma de nube blanca, esto debido a que al ser muy colorido distrae la atención del niño.

La Jetson Nano se fijó en la base interna de la estructura para garantizar estabilidad y flujo de aire hacia su disipador térmico. Como se puede ver en la **Figura *20*** la pantalla táctil se empotró en la cara frontal de la "nube" para facilitar la interacción visual del niño, y el módulo de la

cámara IMX219 se posicionó en la parte superior central para asegurar un encuadre óptimo del torso y las manos durante la ejecución de las señas.

Figura 20
Ensamblaje final del sistema en la carcasa lúdica de madera.



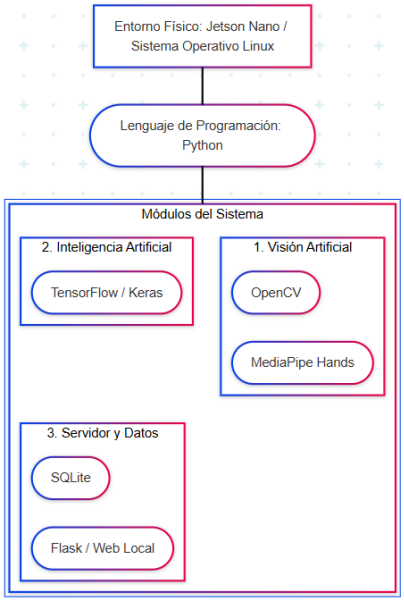
Nota. Elaboración propia, 2026.

3.4.2 Implementación del entorno de software

El sistema fue desarrollado principalmente en Python, debido a su compatibilidad con las herramientas seleccionadas durante la fase de diseño. Para la captura y el procesamiento del video se utilizó OpenCV, mientras que MediaPipe Hands permitió detectar la mano y extraer los landmarks empleados por los modelos de reconocimiento. También se integraron TensorFlow, Keras y TensorFlow Lite para la ejecución de los modelos, Flask para el funcionamiento del servidor web local y SQLite para el almacenamiento de la información generada durante las actividades.

El entorno de software fue configurado para ejecutar todos los procesos de forma local en la Jetson Nano. Esta configuración permite capturar el video, reconocer las señas, presentar la interfaz del estudiante, almacenar los resultados y habilitar el dashboard dentro de la red local. La Figura 21 muestra la configuración general del entorno de software utilizado para la implementación del sistema.

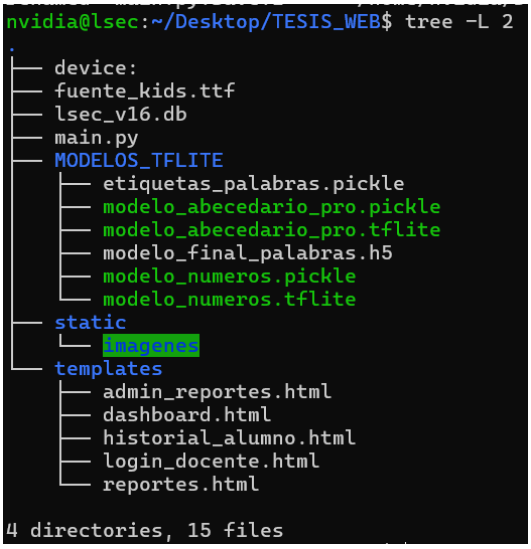
Figura 21
Entorno de software configurado para el desarrollo del sistema



Nota. Elaboración propia, 2026.

La organización del proyecto se estableció mediante carpetas y archivos destinados a los diferentes recursos del sistema. En el directorio principal se encuentra main.py, la base de datos lsec_v16.db, los modelos de reconocimiento, los archivos de etiquetas, las imágenes utilizadas durante las actividades y las plantillas correspondientes al dashboard. Esta distribución facilita la ubicación de los recursos y el mantenimiento del sistema. La Figura *22* presenta la estructura general de carpetas del proyecto implementado.

Figura 22
Estructura general del proyecto implementado



Nota. Elaboración propia, 2026

Dentro de esta estructura, main.py funciona como el archivo principal del sistema. Su función consiste en coordinar la captura de video, la carga de los modelos, el reconocimiento de las señas, el control de las actividades, la retroalimentación visual, el almacenamiento de los resultados y la ejecución del servidor Flask. El funcionamiento de estos procesos se explica posteriormente en los apartados correspondientes.

3.4.3 Captura y preparación del dataset

Para el entrenamiento de los modelos, el dataset se organizó en tres categorías: abecedario, números y palabras dinámicas. Las dos primeras se procesaron como señas estáticas, mientras que las palabras requirieron secuencias de movimiento. La preparación incluyó la organización de las muestras, la extracción de landmarks, la formación de secuencias y la división de los datos para entrenamiento y validación.

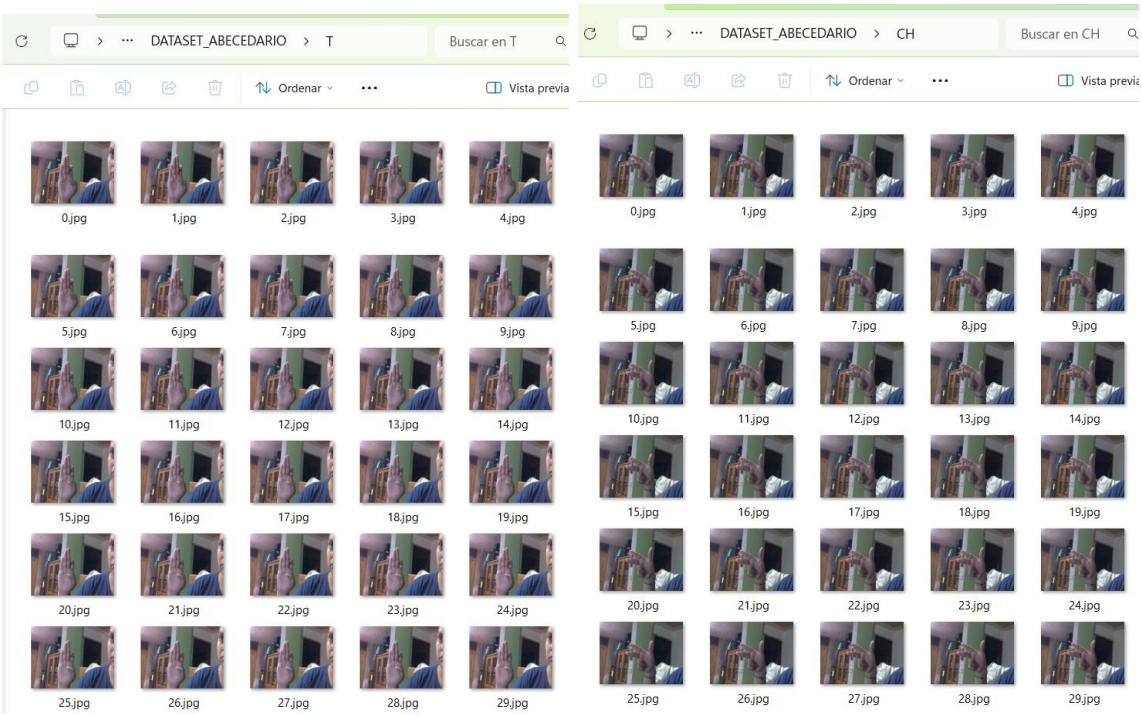
3.4.3.1 Recolección y organización de las muestras

El proceso de recolección se realizó bajo un enfoque híbrido. Para la categoría de números se utilizó el dataset externo *Synthetic ASL Numbers*, disponible en Kaggle, mientras que para el abecedario se utilizó como base el dataset *ASL Alphabet*, también disponible en Kaggle. Estos

conjuntos de datos permitieron contar con una cantidad inicial de imágenes para el entrenamiento de los modelos de señas estáticas.

Sin embargo, debido a que el sistema fue orientado al apoyo del aprendizaje de la Lengua de Señas Ecuatoriana (LSEC), fue necesario complementar el dataset con capturas propias, como se puede observar en la Figura 23. Esto se realizó especialmente en el caso de la consonante compuesta “CH” y la variación de la letra “T”, debido a que estas clases presentaban diferencias con respecto al dataset externo utilizado como base.

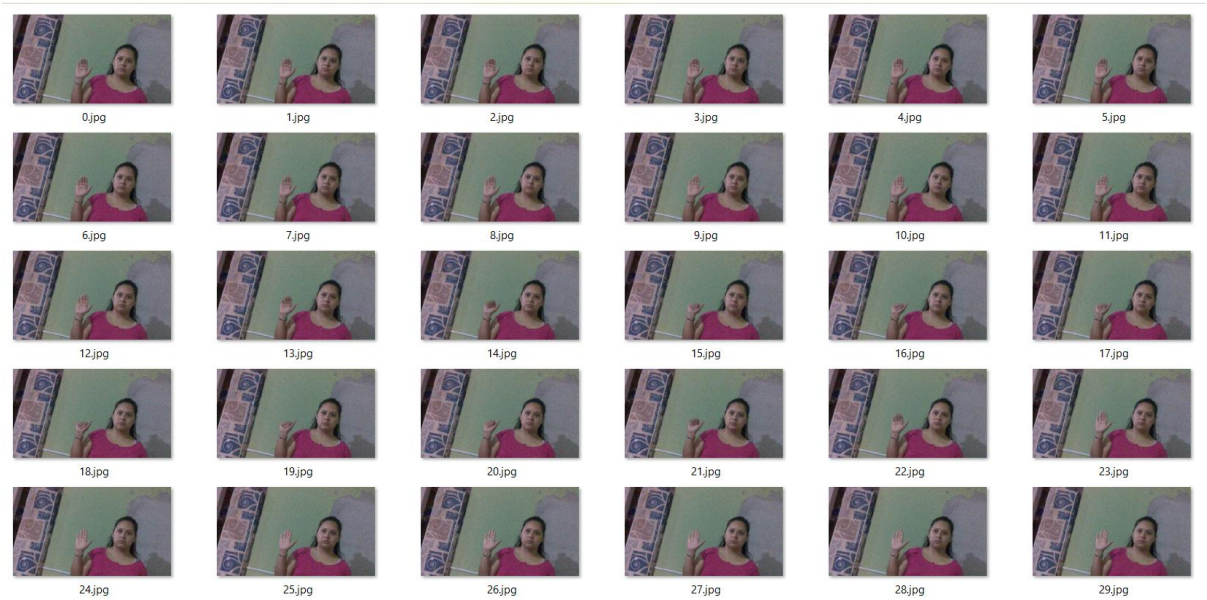
Figura 23
Muestras para la “CH” y la “T”



Nota. Las imágenes corresponden a capturas propias realizadas con la cámara del prototipo para complementar el dataset del abecedario. Elaboración propia, 2026

Para representar los gestos dinámicos, se realizaron capturas propias en ráfagas de 30 frames consecutivos por cada ejecución de la seña. Esta cantidad permitió conservar información del inicio, desarrollo y finalización del movimiento. La Figura 24 muestra la captura del flujo de video utilizado durante la implementación.

Figura 24
Captura del flujo de video desde la cámara del prototipo

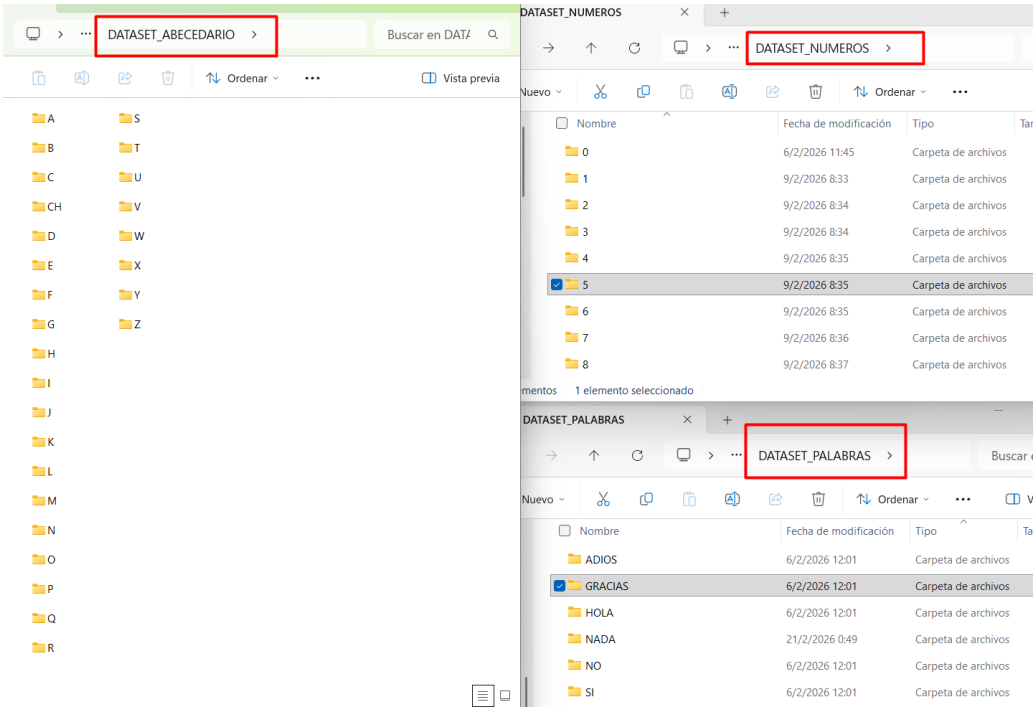


Nota. Elaboración propia, 2026.

Como se observa en la Figura 24, la seña dinámica no puede representarse adecuadamente mediante una sola imagen, ya que el movimiento forma parte de su significado. Por esta razón, esta categoría fue tratada posteriormente como una secuencia temporal de datos para su entrenamiento.

Después de la captura, las imágenes fueron organizadas en carpetas según su categoría, tal como se aprecia en la **Figura *25***. Esta organización permitió que los scripts de procesamiento recorrieran automáticamente cada directorio, identificaran la etiqueta correspondiente y generaran los archivos procesados para el entrenamiento.

Figura 25
Estructura de directorios del dataset



Nota. Elaboración propia, 2026.

Posteriormente, se implementó la detección de mano mediante MediaPipe Hands. Esta herramienta permitió identificar la mano dentro del frame y extraer los puntos característicos o landmarks, los cuales representan la posición de la palma y las articulaciones de los dedos. Estos puntos son utilizados como entrada numérica para los modelos de reconocimiento, evitando trabajar directamente con la imagen completa.

3.4.3.2 Procesamiento espacial de las señas estáticas

Para mejorar la calidad visual de las imágenes antes de la detección de la mano, se implementó una etapa de preprocesamiento mediante CLAHE (Contrast Limited Adaptive Histogram Equalization). Esta técnica fue aplicada con OpenCV y permitió mejorar el contraste de las imágenes cuando existían variaciones de iluminación.

En la **Figura *26*** se muestra el fragmento de código correspondiente a la función aplicar_clahe, utilizada para mejorar la iluminación de las imágenes antes de enviarlas a MediaPipe Hands.

Figura 26
Algoritmo de preprocesamiento de imagen mediante CLAHE

```python
# CLAHE ayuda a mejorar el contraste de la imagen.
# Esto sirve cuando algunas fotos están más oscuras o con diferente iluminación.
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))


def aplicar_clahe(imagen_bgr):
    """
    Mejora la iluminación de la imagen usando CLAHE.
    OpenCV lee las imágenes en formato BGR, por eso se trabaja con ese formato.
    """

    # Convertimos la imagen de BGR a LAB para mejorar solo la parte de iluminación
    lab = cv2.cvtColor(imagen_bgr, cv2.COLOR_BGR2LAB)

    # Separamos los canales de la imagen
    l, a, b = cv2.split(lab)

    # Aplicamos CLAHE solo al canal de luz
    l2 = clahe.apply(l)

    # Volvemos a unir los canales
    lab = cv2.merge((l2, a, b))

    # Regresamos la imagen al formato BGR
    imagen_mejorada = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)

    return imagen_mejorada
```

Nota. Elaboración propia, 2026.

Después de la mejora visual, cada imagen fue procesada mediante MediaPipe Hands, la cual permite detectar 21 puntos de referencia de la mano. Para cada punto se extrajeron las coordenadas X, Y y Z, generando un total de 63 características por imagen.

Para evitar que el modelo aprendiera la ubicación absoluta de la mano, se utilizó la muñeca como punto de referencia, correspondiente a landmark[0]. A partir de ella, las coordenadas de los demás landmarks fueron calculadas de forma relativa.

En la Figura 27 se muestra el código empleado para extraer los landmarks y normalizar las coordenadas respecto a la muñeca. El sistema obtiene las coordenadas base_x, base_y y base_z, y luego recalcula cada punto restando estos valores. Así, el modelo aprende principalmente la forma de la seña y mantiene una representación más estable aunque la mano se mueva ligeramente dentro del encuadre.

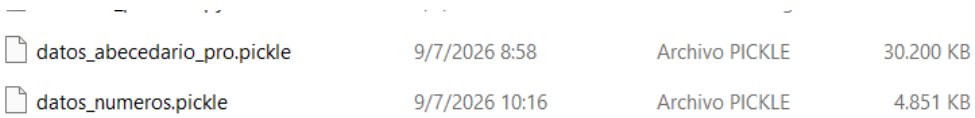
Figura 27
Extracción de landmarks y normalización de coordenadas relativas a la muñeca

```python
# Revisamos si MediaPipe detectó una mano
if results.multi_hand_landmarks:
    # Tomamos solo la primera mano detectada
    hand_landmarks = results.multi_hand_landmarks[0]

    # Usamos la muñeca como punto de referencia.
    # En MediaPipe, el punto 0 corresponde a la muñeca.
    base_x = hand_landmarks.landmark[0].x
    base_y = hand_landmarks.landmark[0].y
    base_z = hand_landmarks.landmark[0].z

    # Aquí se guardarán los puntos de una sola imagen
    data_aux = []

    # MediaPipe detecta 21 puntos de la mano
    for punto in hand_landmarks.landmark:

        # Coordenadas originales del punto
        x = punto.x
        y = punto.y
        z = punto.z

        # Guardamos coordenadas relativas a la muñeca.
        # Esto ayuda a que el modelo no dependa tanto de la posición exacta de la mano.
        data_aux.append(x - base_x)
        data_aux.append(y - base_y)
        data_aux.append(z - base_z)

    # Se valida que existan 63 valores:
    # 21 puntos de la mano * 3 coordenadas = 63
    if len(data_aux) == 63:
        data.append(data_aux)
        labels.append(dir_)
```

Nota. Elaboración propia, 2026.

En el caso del abecedario y los números, los vectores obtenidos fueron almacenados junto con sus respectivas etiquetas en archivos .pickle. Para el abecedario se generó el archivo datos_abecedario_pro.pickle, mientras que para los números se generó datos_numeros.pickle, como se puede ver en la **Figura *28***, estos archivos contienen tanto los datos procesados como las clases asociadas a cada muestra, por lo que fueron utilizados posteriormente en los scripts de entrenamiento.

Figura 28
Archivos de datos serializados (.pickle) listos para el entrenamiento de los modelos



Nota. Elaboración propia, 2026.

3.4.3.3 Procesamiento temporal de las señas dinámicas

A diferencia de las señas estáticas, las palabras o gestos dinámicos no pueden representarse correctamente mediante una sola imagen, ya que su significado depende del movimiento. Por esta razón, el dataset de palabras fue procesado como secuencias temporales.

Para cada palabra dinámica, las imágenes capturadas fueron ordenadas y agrupadas en bloques de 30 frames consecutivos. Cada frame fue procesado con el mismo procedimiento de mejora visual y extracción de landmarks, generando un vector de 63 características por imagen.

En la **Figura *29*** se muestra el fragmento de código donde se agrupan los fotogramas en bloques de 30 imágenes y se guardan como archivos .npy.

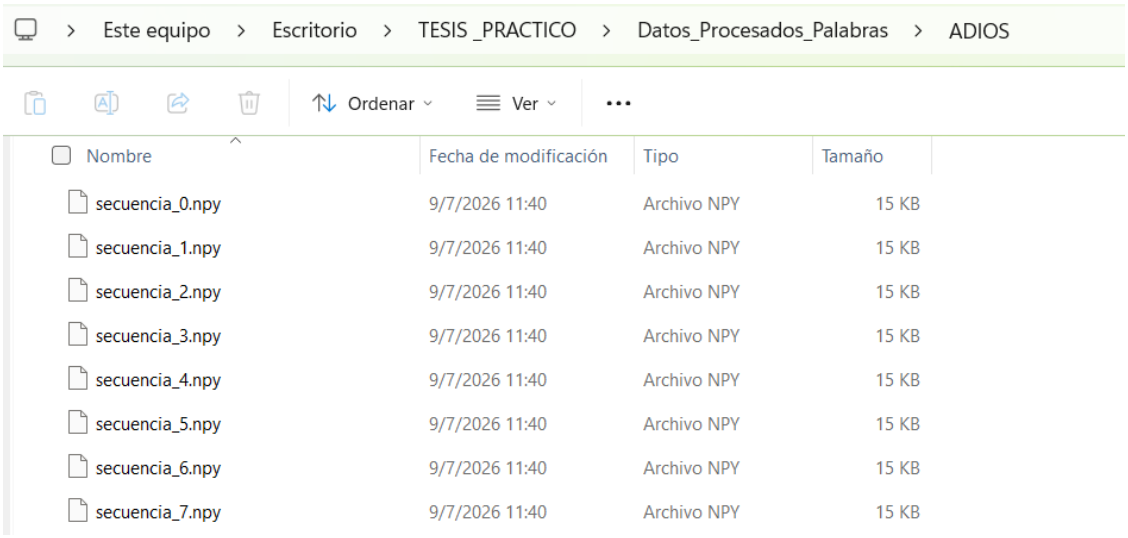
Figura 29
Algoritmo de segmentación temporal y almacenamiento de secuencias de 30 frames

```
# Cada secuencia estará formada por 30 imágenes
FOTOS_POR_RAFAGA = 30
# Calculamos cuántas secuencias completas se pueden formar
total_secuencias = len(fotos) // FOTOS_POR_RAFAGA
secuencias_validas = 0
for i in range(total_secuencias):
    # Aquí se guardarán los 30 fotogramas procesados
    window = []
    # Posición inicial del grupo de 30 imágenes
    idx_inicio = i * FOTOS_POR_RAFAGA
    # Tomamos un lote de 30 fotos
    lote_fotos = fotos[idx_inicio: idx_inicio + FOTOS_POR_RAFAGA]
    for nombre_foto in lote_fotos:
        img_path = os.path.join(ruta_clase, nombre_foto)
        try:
            # Leemos la imagen
            frame = cv2.imread(img_path)
            # Si la imagen no se puede leer, se coloca un vector de ceros
            if frame is None:
                keypoints = np.zeros(63)
                window.append(keypoints)
                continue
            # Mejoramos un poco la iluminación
            frame = aplicar_clahe(frame)
            # MediaPipe trabaja con RGB, por eso convertimos de BGR a RGB
            image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            # Procesamos la imagen para detectar la mano
            results = hands.process(image_rgb)
            # Extraemos los 63 valores de la mano
            keypoints = extraer_puntos_relativos(results)
            # Agregamos el fotograma procesado a la secuencia
            window.append(keypoints)
        except Exception as e:
            # Si una imagen da error, se coloca un vector de ceros para no romper la secuencia
            print(f"Error al procesar {img_path}: {e}")
```

Nota. Elaboración propia, 2026.

Al finalizar cada bloque, el código verifica que la secuencia tenga los 30 fotogramas completos antes de guardarla como archivo .npy, como se puede ver en la **Figura *30***.

Figura 30
Archivos de datos procesados en .npy listo para el entrenamiento



Nota. Elaboración propia, 2026.

Este procedimiento permitió evitar que el modelo recibiera secuencias incompletas. Además, facilitó que las palabras dinámicas fueran preparadas en una estructura adecuada para el entrenamiento de una red LSTM, la cual necesita recibir información organizada.

3.4.3.4 División del dataset

Después de la extracción de características, los datos fueron divididos en dos subconjuntos: 80 % para entrenamiento y 20 % para validación. Se eligió esta proporción para conservar la mayor parte de las muestras durante el aprendizaje y, al mismo tiempo, disponer de un grupo separado para revisar el comportamiento de los modelos con datos no utilizados directamente en el ajuste.

La división se realizó mediante train_test_split con el parámetro test_size=0.2. También se utilizó shuffle=True para mezclar las muestras, stratify=labels_encoded para mantener la distribución de las clases y random_state=42 para repetir la misma separación durante las evaluaciones. La Figura 29 presenta el fragmento correspondiente.

Figura 31
División del dataset

```
# Se divide el dataset en 80 % entrenamiento y 20 % validación/prueba
# stratify ayuda a mantener la misma cantidad proporcional de clases
x_train, x_test, y_train, y_test = train_test_split(
    data,
    labels_one_hot,
    test_size=0.2,
    shuffle=True,
    stratify=labels_encoded,
    random_state=42
)
```

Nota. Elaboración propia, 2026.

3.4.4 Entrenamiento de los modelos de reconocimiento

Una vez preparados los datos, se ejecutaron tres scripts de entrenamiento: uno para el abecedario, otro para los números y uno para las palabras dinámicas. Cada script carga los archivos procesados anteriormente, prepara las etiquetas y construye el modelo correspondiente. De esta manera, las tres categorías se entrenaron por separado y generaron sus propios modelos y archivos de etiquetas.

Las arquitecturas definidas en la fase de diseño fueron implementadas mediante TensorFlow y Keras. Para el abecedario y los números se utilizó una red con dos capas densas de 128 y 64 neuronas. Entre estas capas se aplicó un dropout de 0,2 para reducir el riesgo de que el modelo memorice los datos de entrenamiento. Como se observa en la **Figura *32***, el modelo termina con una capa softmax, encargada de generar la probabilidad correspondiente a cada clase.

Figura 32

Función utilizada para crear los modelos del abecedario y los números

```python
def crear_modelo(num_clases):
    """
    Esta función crea el modelo desde cero.
    Se usa una función para poder entrenar el mismo modelo con diferentes épocas.
    """

    model = Sequential()

    # Entrada del modelo: 63 valores por imagen
    model.add(Input(shape=(63,)))

    # Primera capa oculta
    model.add(Dense(128, activation="relu"))

    # Dropout ayuda a reducir el sobreentrenamiento
    model.add(Dropout(0.2))

    # Segunda capa oculta
    model.add(Dense(64, activation="relu"))

    # Otro Dropout para que el modelo no memorice demasiado
    model.add(Dropout(0.2))

    # Capa de salida, una neurona por cada letra
    model.add(Dense(num_clases, activation="softmax"))

    # Compilación del modelo
    model.compile(
        loss="categorical_crossentropy",
        optimizer="adam",
        metrics=["accuracy"]
    )

    return model
```

Nota. Elaboración propia, 2026.

En el caso de las palabras se implementaron dos capas LSTM de 128 y 64 unidades, seguidas por dos capas densas, tal como se puede ver en la **Figura *33***.

Figura 33
Función utilizada para crear el modelo LSTM de palabras

```python
def crear_modelo_lstm(num_clases):
    """
    Esta función crea el modelo LSTM desde cero.
    Se usa una función para poder entrenar varias veces con diferente número de épocas.
    """
    model = Sequential()
    # Entrada del modelo:
    # 30 fotogramas por secuencia y 63 valores por cada fotograma
    model.add(Input(shape=(FRAMES, FEATURES)))
    # Primera capa LSTM.
    # return_sequences=True porque después viene otra capa LSTM
    model.add(LSTM(128, return_sequences=True, activation="relu"))
    model.add(Dropout(0.2))
    # Segunda capa LSTM.
    # return_sequences=False porque ya pasamos a las capas densas
    model.add(LSTM(64, return_sequences=False, activation="relu"))
    model.add(Dropout(0.2))
    # Capas densas para terminar la clasificación
    model.add(Dense(64, activation="relu"))
    model.add(Dense(32, activation="relu"))
    # Capa de salida.
    # Tiene una neurona por cada palabra
    model.add(Dense(num_clases, activation="softmax"))
    # Compilación del modelo
    model.compile(
        optimizer="adam",
        loss="categorical_crossentropy",
        metrics=["accuracy"]
    )
    return model
```

Nota. Elaboración propia, 2026.

Los tres modelos fueron compilados con el optimizador Adam y la función de pérdida categorical_crossentropy. Durante el entrenamiento se utilizó un tamaño de lote de 32 muestras. Los parámetros principales se resumen en la **Tabla *51***.

Tabla 51
Parámetros utilizados para el entrenamiento de los modelos

Modelo	Arquitectura implementada	Tamaño de lote	Épocas evaluadas	Épocas seleccionadas
Abecedario	Dense 128, Dropout, Dense 64, Dropout y Softmax	32	5, 25, 50, 100 y 200	200
Números	Dense 128, Dropout, Dense 64, Dropout y Softmax	32	5, 25, 50, 100 y 200	100

Modelo	Arquitectura implementada	Tamaño de lote	Épocas evaluadas	Épocas seleccionadas
Palabras	LSTM 128, Dropout, LSTM 64, Dropout, Dense 64, Dense 32 y Softmax	32	5, 25, 50, 100 y 200	200

Nota. Elaboración propia, 2026.

Para determinar la cantidad de épocas, cada modelo se entrenó desde cero con 5, 25, 50, 100 y 200 épocas, como se puede ver en la **Figura *34***.

Figura 34
Cantidades de épocas evaluadas y ejecución del entrenamiento

```
# ==================================================
# ÉPOCAS A PROBAR
# ==================================================
# Se prueban varias épocas para elegir el mejor comportamiento del modelo
epocas_a_probar = [5, 25, 50, 100, 200]
# Aquí se guardarán los resultados de cada entrenamiento
resultados = []
# Variables para guardar el mejor modelo
mejor_modelo = None
mejor_epoca = None
mejor_f1 = -1
mejor_val_accuracy = -1

# ==================================================
# ENTRENAMIENTO CON DIFERENTES ÉPOCAS
# ==================================================
for epocas in epocas_a_probar:
    print("\n==================================================")
    print(f"Entrenando modelo con {epocas} épocas")
    print("==================================================")
    # Limpiamos sesión para no mezclar entrenamientos anteriores
    K.clear_session()
    # Se crea un modelo nuevo para cada número de épocas
    model = crear_modelo(num_clases=len(le.classes_))

    # Entrenamiento del modelo
    # history guarda los valores para hacer las curvas de aprendizaje
    history = model.fit(
        x_train,
        y_train,
        epochs=epocas,
        batch_size=32,
        validation_data=(x_test, y_test),
        verbose=1
    )
```

Nota. Elaboración propia, 2026.

Después de cada ejecución, el código calculó la exactitud, precisión, *recall* y F1-score ponderados., teniendo de esta manera el cálculo de las métricas como se puede ver en la **Figura *35***.

Figura 35
Cálculo de las métricas

```
# ======================================================
# CÁLCULO DE MÉTRICAS
# ======================================================

accuracy = accuracy_score(y_true_classes, y_pred_classes)

precision = precision_score(
    y_true_classes,
    y_pred_classes,
    average="weighted",
    zero_division=0
)

recall = recall_score(
    y_true_classes,
    y_pred_classes,
    average="weighted",
    zero_division=0
)

f1 = f1_score(
    y_true_classes,
    y_pred_classes,
    average="weighted",
    zero_division=0
)
```

Nota. Elaboración propia, 2026.

Como se muestra en la **Figura 36**, el modelo con mayor F1-score fue seleccionado como resultado final; en caso de empate, se consideró la exactitud obtenida durante la validación.

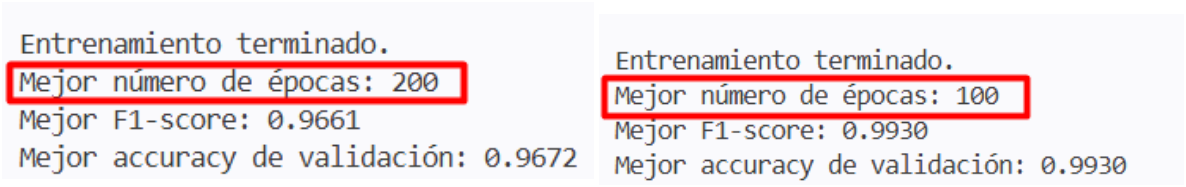
Figura 36
Condición utilizada para seleccionar el mejor modelo.

```
# ======================================================
# SELECCIÓN DEL MEJOR MODELO
# ======================================================

# Se escoge el mejor modelo usando F1-score.
# Si hay empate, se toma el que tenga mejor accuracy de validación.
if (f1 > mejor_f1) or (f1 == mejor_f1 and val_accuracy_final > mejor_val_accuracy):
    mejor_f1 = f1
    mejor_val_accuracy = val_accuracy_final
    mejor_epoca = epocas
    mejor_modelo = model
```

Nota. Elaboración propia, 2026.

Al finalizar las ejecuciones, la consola indicó que el mejor resultado se obtuvo con 200 épocas para el abecedario, 100 para los números y 200 para las palabras, como se evidencia en la Figura 25. Estas cantidades representan el mejor comportamiento entre los valores evaluados para cada modelo. Las curvas de aprendizaje, matrices de confusión y resultados numéricos de las métricas se presentan en el capítulo IV.

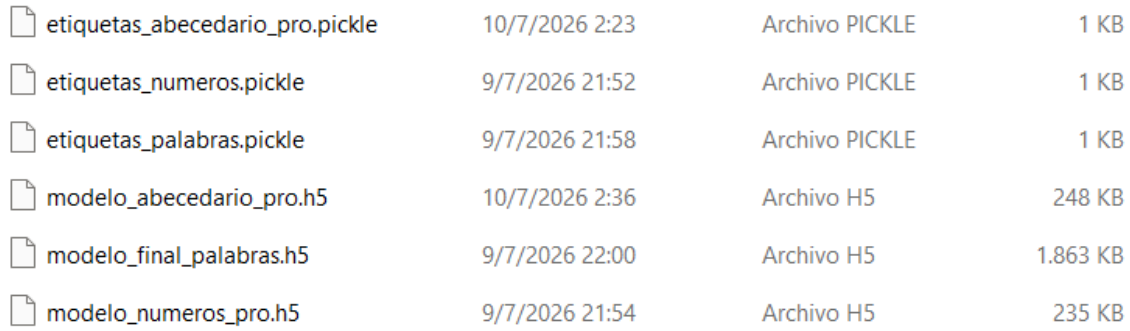
Figura 37
Resultados de la selección del número de épocas



Nota. Elaboración propia, 2026.

Una vez finalizado el entrenamiento, cada script guardó el modelo seleccionado en formato .h5, junto con el archivo .pickle que contiene las etiquetas de cada categoría. Como se observa en la Figura 25, este proceso generó tres modelos independientes para el reconocimiento del abecedario, los números y las palabras dinámicas. Estos archivos fueron utilizados posteriormente para la integración de los modelos en el sistema.

Figura 38
Archivos generados al finalizar el entrenamiento de los modelos



Nota. Elaboración propia, 2026.

3.4.5 Conversión e integración de los modelos entrenados

Una vez seleccionado el mejor modelo de cada categoría, los modelos del abecedario y los números fueron convertidos del formato .h5 a .tflite. Esta conversión se realizó para obtener archivos adecuados para su ejecución local mediante TensorFlow Lite. El modelo de palabras no fue incluido en este proceso, debido a que la aplicación lo carga directamente en formato .h5 mediante Keras.

Para realizar la conversión, el script carga cada modelo mediante load_model() y utiliza TFLiteConverter.from_keras_model() para generar el archivo compatible con TensorFlow Lite. También se aplica Optimize.DEFAULT, con el propósito de optimizar el modelo para su posterior ejecución en el dispositivo. Como se observa en la **Figura *39***, el código comprueba la existencia de cada archivo, realiza la conversión y almacena el resultado en la carpeta de destino.

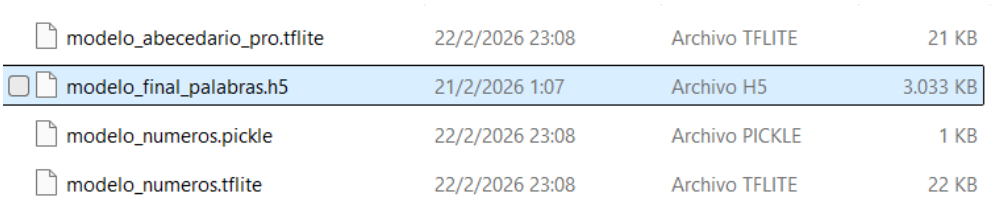
Figura 39
Conversión de los modelos estáticos al formato TensorFlow Lite

```
# Cargamos el modelo entrenado en formato .h5
model = tf.keras.models.load_model(ruta_completa_h5)

# Creamos el convertidor de TensorFlow Lite
converter = tf.lite.TFLiteConverter.from_keras_model(model)

# Esta optimización ayuda a reducir el tamaño del modelo.
# Es útil para usarlo en la Jetson Nano o en sistemas embebidos.
converter.optimizations = [tf.lite.Optimize.DEFAULT]

# Convertimos el modelo a formato .tflite
tflite_model = converter.convert()

# Cambiamos la extensión del archivo de .h5 a .tflite
nombre_tflite = nombre_h5.replace(".h5", ".tflite")

# Ruta donde se guardará el modelo convertido
ruta_salida = os.path.join(carpeta_destino, nombre_tflite)

# Guardamos el archivo .tflite
with open(ruta_salida, "wb") as f:
    f.write(tflite_model)
```

Nota. Elaboración propia, 2026.

Como resultado de este proceso, se generaron los archivos .tflite utilizados posteriormente para la inferencia local en la Jetson Nano. La Figura *40* presenta los dos modelos convertidos dentro de la carpeta de destino.

Figura 40
Modelos estáticos convertidos al formato TensorFlow Lite



Nota. Elaboración propia, 2026.

3.4.6 Integración y ejecución de los modelos en main.py

El archivo main.py integra los modelos de reconocimiento con la captura de video, el procesamiento de los landmarks y la interfaz del estudiante. Durante la ejecución, este archivo selecciona el modelo correspondiente a la categoría de práctica, realiza la inferencia y compara el resultado con la seña solicitada. También controla la retroalimentación visual y el almacenamiento de los resultados en la base de datos.

3.4.6.1 Carga de los modelos y las etiquetas

Al iniciar el sistema, se cargan los modelos del abecedario y los números mediante el intérprete de TensorFlow Lite. La función cargar_tflite() localiza cada archivo, reserva los tensores necesarios para la inferencia y carga el archivo .pickle que contiene la relación entre las salidas numéricas y las clases reconocidas.

El modelo de palabras se carga por separado en formato .h5 mediante Keras. En este caso, también se recupera el archivo de etiquetas correspondiente a las palabras dinámicas. Este procedimiento permite mantener disponibles los tres modelos antes de iniciar las actividades de práctica, como se observa en la **Figura *41***.

Figura 41
Carga de los modelos y archivos de etiquetas en main.py

```
# Diccionarios donde se guardan los modelos TFLite y sus etiquetas.
# Se usan para abecedario y números.
interpreters = {}
label_encoders = {}

def cargar_tflite(nombre, archivo_tflite, archivo_pickle):
    # Carga un modelo TensorFlow Lite y su archivo de etiquetas.
    # El parámetro 'nombre' sirve para identificar si es alfabeto o números.
    try:
        path_tf = os.path.join(MODEL_FOLDER, archivo_tflite)
        path_pk = os.path.join(MODEL_FOLDER, archivo_pickle)
        interp = tflite.Interpreter(model_path=path_tf)
        interp.allocate_tensors()
        interpreters[nombre] = interp
        with open(path_pk, 'rb') as f: label_encoders[nombre] = pickle.load(f)
    except: pass
```

Nota. Elaboración propia, 2026.

3.4.6.2 Configuración de la cámara y MediaPipe Hands

La captura de video se realiza mediante una tubería de GStreamer configurada para la cámara CSI IMX219. Esta tubería permite enviar el flujo de video hacia OpenCV y mantener una captura continua durante la actividad.

Dentro de la función bucle_visual() se inicializa MediaPipe Hands para detectar una mano con una confianza mínima de 0,70. También se abre la cámara y se configura la ventana que presenta la interfaz del estudiante en la pantalla HDMI, como se muestra en la **Figura *42***.

Figura 42
Configuración de la cámara y detección de la mano

```
def gstreamer_pipeline(capture_width=1280, capture_height=720, display_width=640, display_height=480, framerate=30, flip_method=0):
    # Pipeline de GStreamer para usar la cámara CSI de la Jetson Nano.
    # Convierte la imagen de la cámara al formato BGR que usa OpenCV.
    return ( "nvarguscamerasrc ! video/x-raw(memory:NVMM), "
             f"width=(int){capture_width}, height=(int){capture_height}, format=(string)NV12, framerate=(fraction){framerate}/1 ! "
             f"nvvidconv flip-method={flip_method} ! video/x-raw, width=(int){display_width}, height=(int){display_height}, format=(string)BGRx ! "
             "videoconvert ! video/x-raw, format=(string)BGR ! appsink drop=true max-buffers=1" )
```

```python
def bucle_visual():
    # Bucle principal de la interfaz local.
    # Aquí se captura la cámara, se muestra la pantalla HDMI y se realiza la predicción.
    global estado, sequence_lstm
    mp_hands = mp.solutions.hands
    hands = mp_hands.Hands(max_num_hands=1, min_detection_confidence=0.7)

    contador_frames = 0
    fps_start_time = time.time()
    fps_frames = 0

    cap = cv2.VideoCapture(gstreamer_pipeline(flip_method=0), cv2.CAP_GSTREAMER)

    cv2.namedWindow("HDMI", cv2.WINDOW_NORMAL)
    cv2.setWindowProperty("HDMI", cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)
    cv2.setMouseCallback("HDMI", click_pantalla)
```

Nota. Configuración utilizada para capturar el video e inicializar MediaPipe Hands. Elaboración propia.

Durante la implementación se habilitó temporalmente la visualización de los 21 landmarks para comprobar la detección de la mano. Estos puntos se procesan internamente y se utilizan como entrada para los modelos, pero no se muestran en la interfaz final porque podían distraer al niño durante la actividad. La **Figura 43** presenta la configuración utilizada y el resultado de esta verificación.

Figura 43
Detección de mano y landmarks mediante MediaPipe Hands



Nota. Elaboración propia, 2026.

3.4.6.3 Inferencia de las señas estáticas y dinámicas

Durante la práctica, el sistema procesa la imagen capturada mediante MediaPipe Hands y selecciona la ruta de inferencia según la categoría activa. Para las palabras dinámicas se acumula una secuencia de 30 vectores y se ejecuta el modelo LSTM cada tres fotogramas. La predicción se acepta cuando alcanza una confianza superior al 65 %. Además, se utiliza un historial de resultados para evitar que una predicción aislada sea tomada como respuesta definitiva.

Para el abecedario y los números se extraen las coordenadas relativas de los 21 landmarks, formando un vector de 63 valores. Este vector se introduce en el intérprete de TensorFlow Lite y la predicción se acepta cuando supera el 70 % de confianza. Finalmente, el archivo de etiquetas permite transformar el índice obtenido en la letra o número correspondiente. La **Figura *44*** presenta los dos procesos de inferencia implementados.

Figura 44
Inferencia de los modelos de reconocimiento

```
# Si el modelo activo es palabras, se juntan 30 fotogramas y se usa el modelo LSTM.
if estado["modelo_activo"] == "palabras" and model_lstm:
    sequence_lstm.append(extraer_puntos_lstm(results))
    sequence_lstm = sequence_lstm[-30:]
    if len(sequence_lstm) == 30 and contador_frames % 3 == 0:
        with tf.device('/cpu:0'): res = model_lstm(np.expand_dims(sequence_lstm, axis=0).astype(np.float32), training=False)[0].numpy()
        idx = np.argmax(res)
        if res[idx] > 0.65:
            historial_pred.append(etiquetas_lstm[idx])
            estado["ultima_confianza"] = float(res[idx]) * 100
        if len(historial_pred) > 4:
            comun = Counter(historial_pred).most_common(1)[0]
            if comun[1] >= 2: estado["ultimo_detectado"] = comun[0]
# Para abecedario y números se usa TFLite.
# Solo se predice cada 2 frames para reducir carga de procesamiento.
elif estado["modelo_activo"] in interpreters and results.multi_hand_landmarks and contador_frames % 2 == 0:
    hl = results.multi_hand_landmarks[0]; coords = []; base = hl.landmark[0]
    for p in hl.landmark: coords.extend([p.x - base.x, p.y - base.y, p.z - base.z])
    interp, enc = interpreters[estado["modelo_activo"]], label_encoders[estado["modelo_activo"]]
    interp.set_tensor(interp.get_input_details()[0]['index'], np.array([coords], dtype=np.float32).reshape(1, 63))
    interp.invoke()
    res = interp.get_tensor(interp.get_output_details()[0]['index'])
    idx = np.argmax(res[0])
    if res[0][idx] > 0.70:
        dt = enc.inverse_transform([idx])[0]
        if dt != "NADA": estado["ultimo_detectado"], estado["ultima_confianza"] = dt, float(res[0][idx]) * 100
```

Nota. Procesos utilizados para ejecutar los modelos según la categoría seleccionada. Elaboración propia.

3.4.6.4 Comparación, retroalimentación y registro

Después de obtener la predicción, el sistema convierte a mayúsculas la clase detectada y la seña solicitada para realizar la comparación. Cuando ambas coinciden, se muestra el mensaje “MUY BIEN” y se incrementa el progreso de reconocimiento. Una vez confirmada la respuesta durante varios fotogramas, se presenta la pantalla de felicitación.

Si la predicción no coincide, la interfaz solicita repetir la seña y controla los intentos realizados. Después de varios intentos sin coincidencia, el sistema vuelve a mostrar la referencia visual para ayudar al estudiante.

Cuando la seña es reconocida correctamente, se almacena en SQLite el estudiante, número de sesión, categoría, seña solicitada, estado, confianza y tiempo utilizado. Las señas omitidas mediante la opción de salto se registran con el estado SALTO. La **Figura *45*** muestra la comparación y el almacenamiento del resultado.

Figura 45
Comparación de la predicción y registro del resultado

```
# Si la predicción coincide con el objetivo, aumenta el progreso.
if act == obj:
    estado["color_marco"], estado["mensaje"], estado["progreso_exito"], estado["frames_malos"] = VERDE_MENTA, "MUY BIEN!", estado["progreso_exito"] + 2, 0
    cv2.rectangle(frame_show, (0, 460), (int((estado["progreso_exito"] / 20) * 640), 480), (0, 255, 0), -1)
    if estado["progreso_exito"] > 20:
        estado["pantalla"], estado["intentos_fallidos"], estado["mostrar_ayuda"] = "FELICITACION", 0, False
        t_inv, estado["timer_estado"] = round(time.time() - estado["timer_estado"], 1), time.time()
        try:
            conn = get_db()
            conn.execute("INSERT INTO registro_practicas (nombre, numero_sesion, categoria, signo, estado, confianza_ia, tiempo_segundos) VALUES (?, ?, ?, ?, 'EXITO',
                         (estado["alumno"]["nombre"], estado["numero_sesion"], obtener_categoria(estado["modelo_activo"]), obj, estado["ultima_confianza"], t_inv))
            conn.commit()
        except: pass
else:
    # Si no coincide, se muestra retroalimentación para que el estudiante vuelva a intentar.
    estado["color_marco"], estado["mensaje"], estado["progreso_exito"] = ROJO_SUAVE, f"Intenta: {obj}", max(0, estado["progreso_exito"] - 1)
    cv2.rectangle(frame_show, (0, 460), (640, 480), (200, 200, 200), -1)
    if act != "?":
        estado["frames_malos"] += 1
        if estado["frames_malos"] > 30:
            estado["intentos_fallidos"] += 1; estado["frames_malos"] = 0
            if estado["intentos_fallidos"] >= 3:
                estado["mostrar_ayuda"], estado["pantalla"], estado["timer_estado"], estado["intentos_fallidos"], estado["progreso_exito"] = True, "TEORIA", time.time
                sequence_lstm.clear(); historial_pred.clear(); continue
```

Nota. Fragmento utilizado para comparar la seña reconocida y almacenar el resultado de la actividad. Elaboración propia.

3.4.6.5 Ejecución simultánea de la interfaz y el dashboard

Al ejecutar main.py, el bucle visual se inicia en un hilo independiente, mientras que Flask mantiene activo el servidor web local. Esta organización permite que la interfaz del estudiante funcione en la pantalla de la Jetson Nano y que, al mismo tiempo, la docente acceda al

dashboard desde otro dispositivo conectado a la red local, el código usado se puede ver en la **Figura *46*** .

Figura 46
Ejecución simultánea de la interfaz y el dashboard

```
# Punto de inicio del programa.
# Primero se lanza la interfaz visual en un hilo aparte.
# Luego se inicia el servidor Flask para el dashboard.
if __name__ == '__main__':
    threading.Thread(target=bucle_visual, daemon=True).start()
    app.run(host='0.0.0.0', port=PUERTO_WEB, debug=False)
```

Nota. Elaboración propia

3.4.7 Implementación de la interfaz del estudiante

La implementación de la interfaz del estudiante permitió integrar la parte visual del sistema con el proceso de reconocimiento de señas. Esta interfaz fue diseñada para presentar actividades de práctica de forma sencilla, clara y adecuada para niños de 6 a 8 años, priorizando elementos visuales, instrucciones breves y retroalimentación inmediata.

La interfaz se ejecuta localmente en la pantalla conectada a la Jetson Nano. Desde esta pantalla, el estudiante puede observar la seña objetivo que debe realizar, ubicarse frente a la cámara y recibir una respuesta visual del sistema después del reconocimiento. Esta organización permite que el niño interactúe directamente con la herramienta sin necesidad de manipular configuraciones técnicas.

La **Figura 47**muestra la pantalla principal de la interfaz del estudiante.

Figura 47
Pantalla principal de la interfaz del estudiante



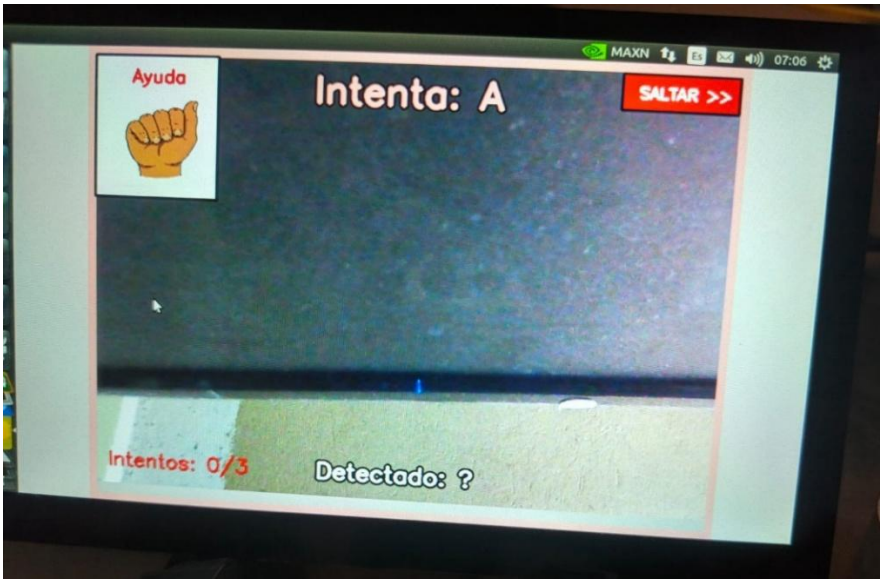
Nota. Elaboración propia, 2026.

Como se observa en la **Figura 47**, la interfaz presenta elementos visuales orientados a guiar la actividad de práctica. La seña objetivo se muestra de forma destacada, junto con indicaciones que permiten al estudiante comprender qué acción debe realizar frente a la cámara.

Durante la ejecución de la actividad, el sistema captura la seña realizada, procesa los landmarks y genera una predicción mediante el modelo correspondiente. Posteriormente, la interfaz muestra una retroalimentación visual que indica si la seña fue reconocida correctamente o si el estudiante debe intentarlo nuevamente.

La **Figura 48** presenta un ejemplo de retroalimentación visual mostrada al estudiante.

Figura 48
Retroalimentación visual durante la práctica de señas



Nota. Elaboración propia, 2026.

Como se observa en la **Figura 48**, la retroalimentación permite informar al estudiante el resultado de su intento de manera inmediata. Este recurso es importante dentro de la herramienta didáctica, ya que refuerza el aprendizaje y permite que el niño corrija la seña durante la práctica.

La implementación de esta interfaz permitió conectar el módulo de reconocimiento con la experiencia visual del estudiante. De esta manera, el sistema no solo ejecuta la predicción de señas, sino que también presenta la información de forma comprensible para el usuario final.

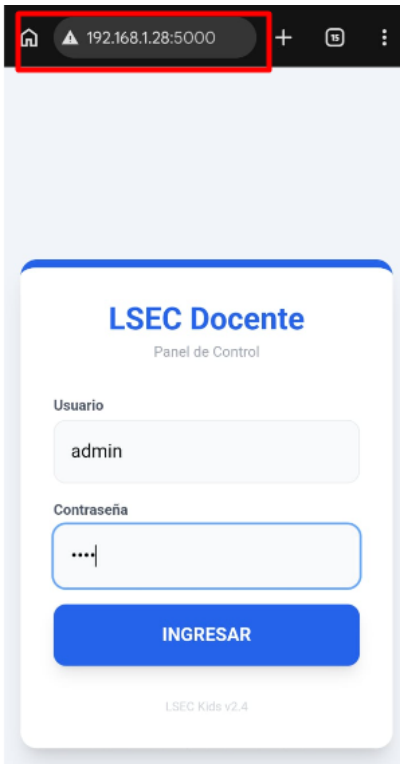
3.4.8 Implementación del dashboard docente

La implementación del dashboard docente permitió incorporar una interfaz web local para la supervisión de las actividades realizadas por los estudiantes. Este panel fue desarrollado con el propósito de que la docente pueda acceder a la información generada por el sistema, consultar los resultados de práctica y realizar el seguimiento del desempeño sin intervenir directamente en la interfaz utilizada por el niño.

El dashboard se ejecuta desde la Jetson Nano mediante un servidor local desarrollado con Flask. El acceso se realiza desde un dispositivo externo, como teléfono, tableta o computadora, siempre que se encuentre conectado a la misma red local del prototipo. Esta característica permite mantener el funcionamiento del sistema dentro de un entorno local, sin depender de servicios en la nube o conexión permanente a internet.

La **Figura 49** muestra el acceso al dashboard docente.

Figura 49
Acceso al dashboard docente mediante red local

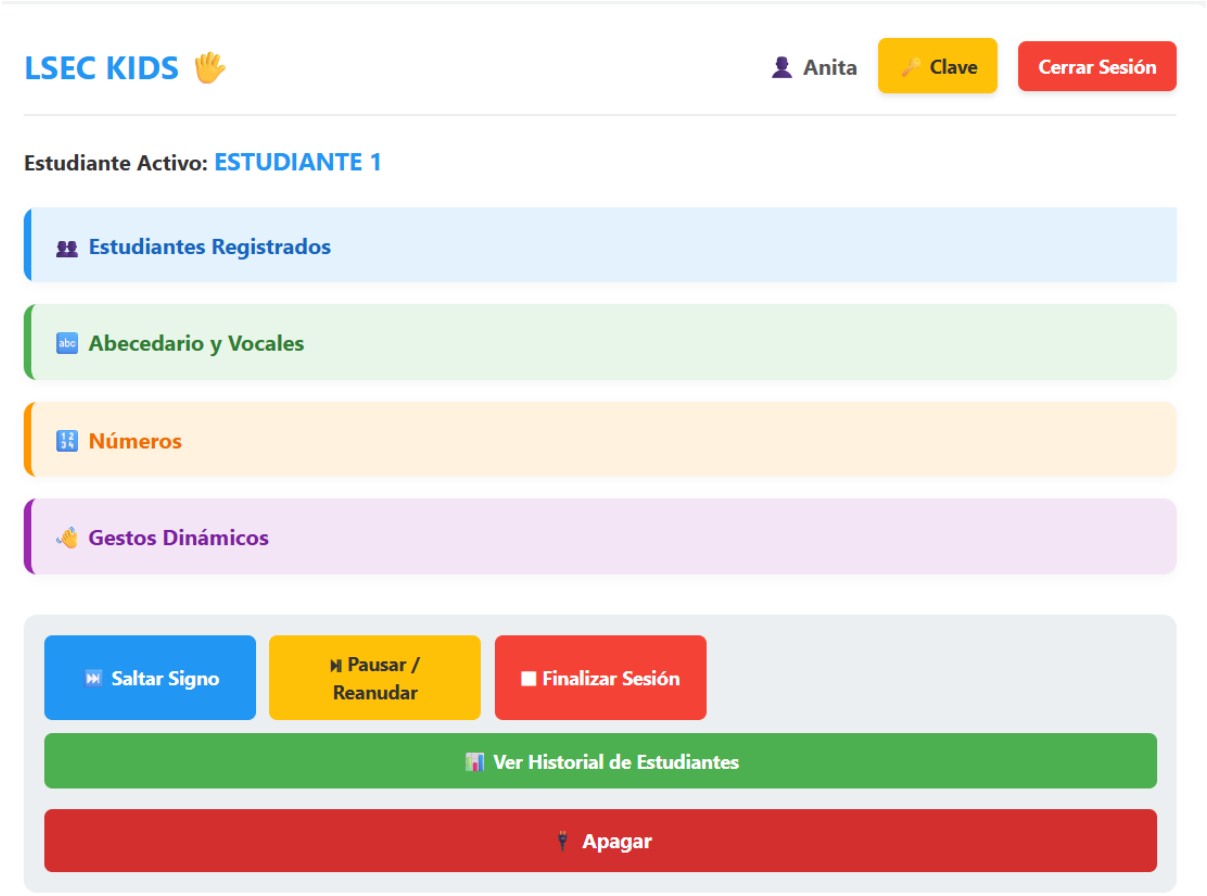


Nota. Elaboración propia, 2026.

Dentro del dashboard se implementaron opciones para seleccionar o consultar estudiantes, revisar categorías de práctica y visualizar los resultados almacenados en la base de datos local. Esta organización permite que la docente identifique las señas practicadas, los intentos realizados y los resultados obtenidos durante las sesiones.

La **Figura 50** presenta la pantalla principal del dashboard docente implementado.

Figura 50
Pantalla principal del dashboard docente



Nota. Elaboración propia, 2026.

Como se observa en la **Figura 50**, el dashboard organiza la información de forma clara para facilitar la consulta docente. Los registros obtenidos durante las prácticas se presentan a través de tablas o secciones de consulta, permitiendo revisar el desempeño por estudiante, categoría o sesión. Esta visualización facilita el seguimiento del proceso de aprendizaje y permite identificar las señas que requieren mayor refuerzo.

La implementación del dashboard también permitió integrar la base de datos SQLite con la interfaz web. De esta manera, los resultados generados por el sistema de reconocimiento no permanecen únicamente como datos internos, sino que pueden ser consultados desde un panel accesible para la docente.

3.4.9 Implementación de la base de datos y reportes

La base de datos del sistema fue implementada mediante SQLite y se almacena localmente en la Jetson Nano con el nombre lsec_v16.db. Al iniciar la aplicación, la función init_db() verifica la existencia de las tablas y agrega los campos requeridos cuando todavía no se encuentran disponibles. Esto permite conservar la información de estudiantes, docentes, gestos dinámicos y actividades de práctica dentro del mismo entorno de ejecución.

Para registrar los resultados se utiliza principalmente la tabla registro_practicas. En ella se almacena la información generada cuando una seña es reconocida correctamente o cuando el estudiante decide omitirla mediante la opción de salto. Los campos principales se presentan en la **Tabla *52***.

Tabla 52
Datos almacenados en la tabla registro_practicas

Campo	Información almacenada
id	Identificador interno del registro.
nombre	Nombre del estudiante que realiza la actividad.
numero_sesion	Número de la sesión de práctica.
categoria	Categoría seleccionada: abecedario, números o gestos dinámicos.
signo	Seña solicitada durante la actividad.
estado	Resultado final registrado como EXITO o SALTO.
confianza_ia	Confianza obtenida por el modelo cuando la seña es aceptada.
tiempo_segundos	Tiempo empleado para completar la actividad.

Nota. Elaboración propia, 2026.

La base de datos conserva el resultado final de cada actividad. El estado EXITO indica que la seña fue reconocida y confirmada por el sistema, mientras que SALTO señala que la actividad fue omitida. Los intentos intermedios y las clases detectadas durante cada fotograma no se almacenan como registros independientes. La **Figura *51*** muestra la base de datos local implementada en SQLite.

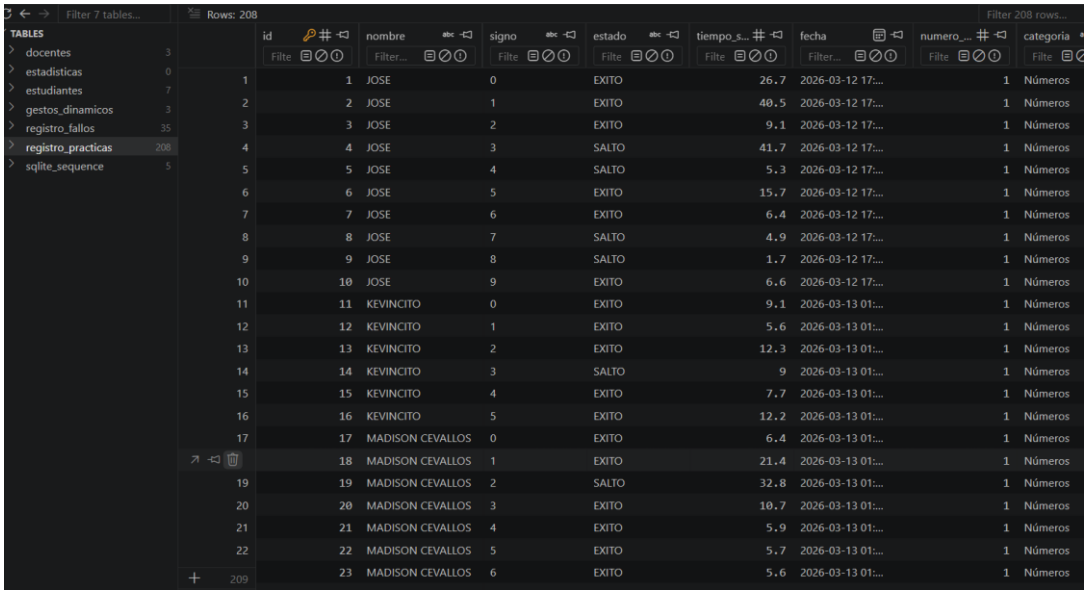
Figura 51
Base de datos local implementada en SQLite



Nota. Elaboración propia, 2026.

Durante la implementación, se verificó que los resultados generados por el reconocimiento sean almacenados correctamente en la base de datos. Para ello, el sistema registra la información después de cada intento realizado por el estudiante. Estos registros permiten conocer si la seña fue reconocida correctamente o si se produjo un error durante la práctica. La **Figura *52*** presenta un ejemplo de registros almacenados después de una sesión de práctica.

Figura 52
Registros almacenados durante una sesión de práctica



Filter 7 tables... | Rows: 208 | Filter 208 rows...

TABLES
- docentes 3
- estadisticas 0
- estudiantes 7
- gestos_dinamicos 3
- registro_fallos 35
- registro_practicas 208
- sqlite_sequence 5

	id	nombre	signo	estado	tiempo_s...	fecha	numero_...	categoria
1	1	JOSE	0	EXITO	26.7	2026-03-12 17:...	1	Números
2	2	JOSE	1	EXITO	40.5	2026-03-12 17:...	1	Números
3	3	JOSE	2	EXITO	9.1	2026-03-12 17:...	1	Números
4	4	JOSE	3	SALTO	41.7	2026-03-12 17:...	1	Números
5	5	JOSE	4	SALTO	5.3	2026-03-12 17:...	1	Números
6	6	JOSE	5	EXITO	15.7	2026-03-12 17:...	1	Números
7	7	JOSE	6	EXITO	6.4	2026-03-12 17:...	1	Números
8	8	JOSE	7	SALTO	4.9	2026-03-12 17:...	1	Números
9	9	JOSE	8	SALTO	1.7	2026-03-12 17:...	1	Números
10	10	JOSE	9	EXITO	6.6	2026-03-12 17:...	1	Números
11	11	KEVINCITO	0	EXITO	9.1	2026-03-13 01:...	1	Números
12	12	KEVINCITO	1	EXITO	5.6	2026-03-13 01:...	1	Números
13	13	KEVINCITO	2	EXITO	12.3	2026-03-13 01:...	1	Números
14	14	KEVINCITO	3	SALTO	9	2026-03-13 01:...	1	Números
15	15	KEVINCITO	4	EXITO	7.7	2026-03-13 01:...	1	Números
16	16	KEVINCITO	5	EXITO	12.2	2026-03-13 01:...	1	Números
17	17	MADISON CEVALLOS	0	EXITO	6.4	2026-03-13 01:...	1	Números
18	18	MADISON CEVALLOS	1	EXITO	21.4	2026-03-13 01:...	1	Números
19	19	MADISON CEVALLOS	2	SALTO	32.8	2026-03-13 01:...	1	Números
20	20	MADISON CEVALLOS	3	EXITO	10.7	2026-03-13 01:...	1	Números
21	21	MADISON CEVALLOS	4	EXITO	5.9	2026-03-13 01:...	1	Números
22	22	MADISON CEVALLOS	5	EXITO	5.7	2026-03-13 01:...	1	Números
23	23	MADISON CEVALLOS	6	EXITO	5.6	2026-03-13 01:...	1	Números

Nota. Elaboración propia, 2026.

3.4.9.1 Exportación de resultados para la validación

Como apoyo para el análisis realizado en el capítulo IV, se implementó una función que exporta los registros en formato CSV. Esta función agrupa la información de acuerdo con la seña esperada y calcula la cantidad de aciertos, los saltos registrados, el total de actividades y el porcentaje de aciertos obtenido.

El archivo CSV contiene los siguientes campos:

- Seña esperada.
- Aciertos.
- Saltos registrados.
- Total de actividades.
- Porcentaje de aciertos.

La Figura 53 presenta el código utilizado para generar el archivo.

Figura 53
Generación del archivo CSV para el análisis de validación

```
# Genera la matriz cuantitativa para la validación
query = """
    SELECT signo,
           SUM(CASE WHEN estado='EXITO' THEN 1 ELSE 0 END) as aciertos,
           SUM(CASE WHEN estado='SALTO' THEN 1 ELSE 0 END) as fallos,
           COUNT(*) as total
    FROM registro_practicas
    GROUP BY signo
    ORDER BY signo
"""
registros = conn.execute(query).fetchall()

output = io.StringIO()
writer = csv.writer(output)
# Cabeceras exactas para tu documento de tesis
writer.writerow(['Signo (Esperado)', 'Aciertos (Verdaderos Positivos)', 'Fallos / Saltos', 'Total de Intentos', 'Precision del Sistema (%)'])

for r in registros:
    precision = round((r['aciertos'] / r['total']) * 100, 2) if r['total'] > 0 else 0
    writer.writerow([r['signo'], r['aciertos'], r['fallos'], r['total'], precision])
```

Nota. Archivo generado a partir de los resultados almacenados en SQLite. Elaboración propia.

3.4.10 Manual de usuario del sistema

Como complemento a la implementación del prototipo, se elaboró el manual de usuario del sistema, presentado en el Anexo G. En este documento se describen los procedimientos para acceder al dashboard docente, gestionar estudiantes, seleccionar actividades, controlar sesiones, consultar reportes de progreso y apagar el sistema de forma segura. Además, se incluyen buenas prácticas de uso, criterios para interpretar los reportes, problemas frecuentes y recomendaciones para la operación adecuada del sistema

CAPÍTULO IV. PRUEBAS DE FUNCIONAMIENTO

En este capítulo se presentan los resultados de la evaluación de los modelos de reconocimiento y las pruebas realizadas con el prototipo integrado. Se analiza el funcionamiento de sus principales módulos, el rendimiento de la Jetson Nano, la aplicación del sistema con los usuarios participantes y la valoración realizada por la docente.

4.1 Evaluación de los modelos de reconocimiento

En este apartado se presentan los resultados obtenidos durante la evaluación de los modelos desarrollados para el reconocimiento del abecedario, los números y las palabras dinámicas. La evaluación se realizó mediante la comparación de diferentes cantidades de épocas, el análisis de las curvas de aprendizaje, las matrices de confusión y el cálculo de las métricas de exactitud, precisión, recall y F1-score.

Para determinar la cantidad de épocas adecuada, cada modelo fue entrenado desde cero con 5, 25, 50, 100 y 200 épocas. Después de cada entrenamiento se calcularon las métricas correspondientes sobre el conjunto de validación.

Debido a que el F1-score permite considerar de manera conjunta la precisión y el recall, esta métrica fue utilizada como criterio principal para seleccionar el modelo final. La **Tabla *53*** presenta la comparación del F1-score obtenido en cada entrenamiento.

Tabla 53
Comparación del F1-score ponderado según el número de épocas

Modelo	5 épocas	25 épocas	50 épocas	100 épocas	200 épocas	Épocas seleccionadas
Abecedario	97,33 %	98,52 %	98,35 %	99,00 %	**99,06 %**	200
Números	98,03 %	98,96 %	99,19 %	**99,30 %**	99,25 %	100
Palabras	53,75 %	61,82 %	52,19 %	88,91 %	**96,61 %**	200

Nota. Los valores corresponden al F1-score ponderado obtenido con el conjunto de validación. Elaboración propia, 2026.

Como se puede ver los modelos no alcanzaron su mejor resultado con la misma cantidad de épocas. El abecedario y las palabras obtuvieron el mayor F1-score con 200 épocas, mientras que el modelo de números alcanzó su mejor desempeño con 100 épocas. Las curvas completas y las matrices de confusión de todas las configuraciones se presentan en el anexo D

4.1.1 Evaluación del modelo estático para el abecedario

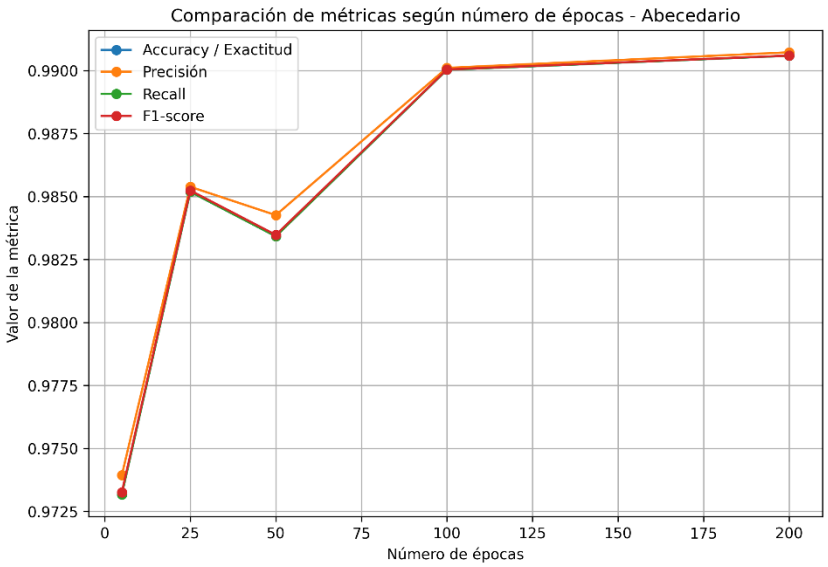
El modelo de abecedario presentó resultados superiores al 97 % desde la configuración de 5 épocas. Al aumentar el entrenamiento, las métricas mostraron una mejora gradual, aunque en 50 épocas se produjo una disminución pequeña respecto a las 25 épocas. Posteriormente, el desempeño volvió a incrementarse hasta alcanzar el mejor resultado con 200 épocas.

En esta configuración, el modelo obtuvo una exactitud del 99,06 %, una precisión del 99,07 %, un recall del 99,06 % y un F1-score del 99,06 %. La pérdida de validación final fue de 0,0408. Aunque la diferencia respecto a las 100 épocas fue reducida, la configuración de 200 épocas presentó el F1-score más alto y la menor cantidad de errores entre las alternativas evaluadas.

Como se observa en la **Figura *54***, las cuatro métricas mantienen valores cercanos entre sí. Esto indica que el modelo presentó un desempeño equilibrado en el reconocimiento de las diferentes letras. La matriz de confusión de 200 épocas mostró que los errores se concentraron principalmente entre las letras M y N, así como entre U y V. El análisis completo se encuentra en el Anexo B.

Figura 54

Comparación de las métricas del modelo de abecedario según el número de épocas



Nota. Elaboración propia a partir de los resultados del entrenamiento, 2026.

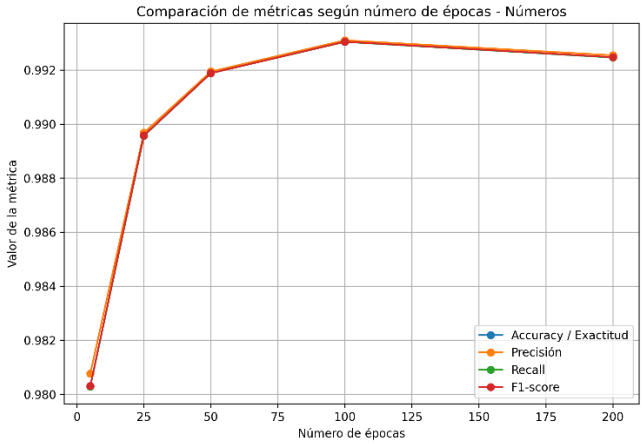
4.1.2 Evaluación del modelo estático para números

En el modelo de números se observó una mejora continua desde las 5 hasta las 100 épocas. La configuración de 100 épocas alcanzó una exactitud del 99,30 %, una precisión del 99,31 %, un recall del 99,30 % y un F1-score del 99,30 %. En la matriz de confusión se registraron 12 errores dentro de las 1725 muestras de validación.

Al aumentar el entrenamiento a 200 épocas, el F1-score disminuyó ligeramente hasta el 99,25 % y la pérdida de validación aumentó de 0,0667 a 0,1122. Esto indica que prolongar el entrenamiento no mejoró la respuesta del modelo sobre los datos de validación. Por esta razón, se seleccionó la configuración de 100 épocas.

La Figura 55 muestra que el mejor resultado se obtuvo con 100 épocas. Las métricas disminuyen ligeramente al llegar a 200 épocas, lo que respalda la configuración seleccionada. La matriz de confusión correspondiente a 100 épocas presenta una diagonal definida y errores aislados, sin que exista una confusión constante entre dos números específicos.

Figura 55
Comparación de las métricas del modelo de números según el número de épocas



Nota. Elaboración propia a partir de los resultados del entrenamiento, 2026.

4.1.3 Evaluación del modelo dinámico para palabras

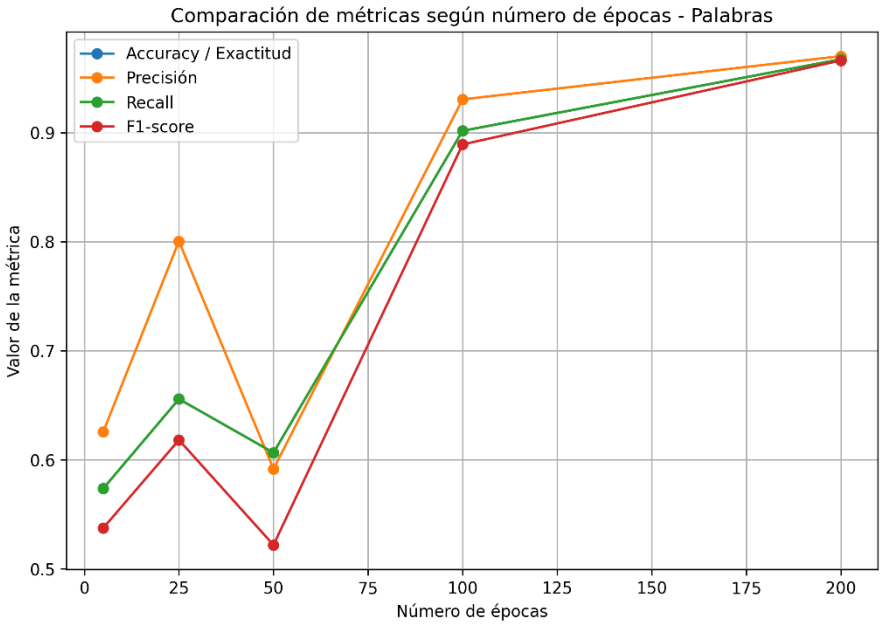
El modelo de palabras presentó mayores variaciones durante el entrenamiento debido a que trabaja con secuencias de movimiento. Con 5 épocas obtuvo un F1-score del 53,75 %, mientras que con 25 épocas alcanzó el 61,82 %. Sin embargo, al entrenar durante 50 épocas el F1-score disminuyó al 52,19%, lo que mostró que el proceso todavía no era estable.

A partir de las 100 épocas se observó una mejora importante, con una exactitud del 90,16 % y un F1-score del 88,91 %. El mejor resultado se alcanzó con 200 épocas, configuración que obtuvo una exactitud del 96,72 %, una precisión del 97,01 %, un recall del 96,72 % y un F1-score del 96,61 %. La pérdida final de validación también disminuyó hasta 0,1359.

Como se observa en la **Figura *56***, existe una diferencia marcada entre las primeras configuraciones y los entrenamientos de 100 y 200 épocas. La matriz de confusión de 200 épocas mostró 59 clasificaciones correctas de un total de 61 secuencias. Los dos errores correspondieron a la palabra NADA, que fue clasificada una vez como GRACIAS y otra como SÍ. Debido a estos resultados, se seleccionó el modelo entrenado durante 200 épocas. Las curvas y matrices completas se presentan en el Anexo D.

Figura 56

Comparación de las métricas del modelo de palabras según el número de épocas



Nota. Elaboración propia a partir de los resultados del entrenamiento, 2026.

4.2 Escenario de pruebas

Después de evaluar y seleccionar los modelos de reconocimiento, se realizaron pruebas con el prototipo completo para comprobar la interacción entre sus componentes. El sistema fue ejecutado localmente en la Jetson Nano, utilizando la cámara IMX219 para la captura, la pantalla HDMI para la interfaz del estudiante y un dispositivo externo para acceder al dashboard.

Las pruebas se desarrollaron en un espacio interior con iluminación estable y sin obstáculos frente a la cámara. El usuario se ubicó de manera que la mano permaneciera dentro del campo de visión durante la ejecución de las señas. El teléfono o computadora utilizado para el dashboard se conectó a la red local del prototipo, sin depender de acceso a Internet.

La **Figura 57** presenta el escenario utilizado para las pruebas del sistema.

Figura 57
Escenario de pruebas del prototipo integrado



Nota. Elaboración propia, 2026.

Como se observa en la **Figura 57**, el escenario de pruebas integra los elementos principales del sistema: prototipo físico, cámara, pantalla, red local y dispositivo de consulta docente. Esta disposición permitió evaluar el flujo completo de funcionamiento, desde la ejecución de la seña hasta el registro y visualización de resultados.

Para mantener condiciones similares durante las pruebas, se consideraron aspectos como la iluminación del entorno, la distancia aproximada entre la mano y el dispositivo, y la conexión del equipo docente a la misma red local. Estas condiciones permiten reducir variaciones externas que podrían afectar la captura de la imagen o el reconocimiento de la seña.

La **Tabla *54*** resume las condiciones consideradas durante el escenario de pruebas.

Tabla 54
Condiciones generales del escenario de pruebas

Elemento	Condición utilizada
Procesamiento	NVIDIA Jetson Nano con la aplicación LSEC KIDS
Cámara	IMX219 CSI
Captura	Configuración de 30 FPS y salida de 640 × 480 píxeles

Pantalla	Pantalla HDMI táctil de 7 pulgadas
Conectividad	Red local del prototipo
Acceso web	lsec.local o dirección IP 10.42.0.1
Dispositivo externo	Teléfono, tableta o computadora
Categorías	Abecedario, números y palabras dinámicas
Estados registrados	EXITO y SALTO
Datos almacenados	Estudiante, sesión, categoría, seña y estado

Nota. Elaboración propia, 2026.

4.3 Plan de pruebas

A partir del escenario definido, se estableció un plan para comprobar el funcionamiento de los modelos de reconocimiento, la integración de los módulos, el comportamiento del hardware y la interacción con los usuarios. Las pruebas se organizaron según los componentes y flujos implementados en el prototipo. Los criterios se establecieron mediante resultados observables.

La **Tabla *55*** presenta el plan general de pruebas aplicado al sistema.

Tabla 55
Plan general de pruebas del sistema

Tipo de prueba	Código	Prueba	Propósito
PRUEBAS FUNCIONALES Y DE INTEGRACION	PFI-01	Acceso y autenticación	Verificar el ingreso al dashboard según las credenciales registradas.
	PFI-02	Gestión y selección del estudiante	Comprobar el registro, modificación y selección de estudiantes.
	PFI-03	Configuración de la actividad	Verificar la selección de categorías y señas desde el dashboard.
	PFI-04	Captura y detección de la mano	Comprobar la captura continua de video y el procesamiento de los landmarks.
	PFI-05	Reconocimiento de señas estáticas	Verificar la inferencia de letras y números.
	PFI-06	Reconocimiento de gestos dinámicos	Comprobar el procesamiento de las

			secuencias de movimiento.
	PFI-07	Retroalimentación y controles	Verificar los mensajes visuales y las opciones Pausar, Saltar Signo y Finalizar Sesión.
	PFI-08	Registro final de la actividad	Comprobar el almacenamiento de los resultados finales.
	PFI-09	Consulta de resultados e historial	Verificar que la docente pueda consultar las actividades almacenadas.
	PFI-10	Exportación administrativa	Comprobar la generación del archivo CSV desde el modo administrador.
PRUEBA DE RENDIMIENTO	PR-01	Rendimiento durante la operación	Observar el comportamiento de la Jetson Nano mientras el sistema permanece activo.
PRUEBAS CON USUARIOS	PU-01	Validación con los niños	Comprobar el uso del prototipo con participantes de 6 a 8 años.

Nota. Elaboración propia, 2026.

4.3.1 Pruebas funcionales y de integración del sistema

Las pruebas funcionales y de integración se realizaron de manera conjunta para comprobar las funciones principales del sistema y la comunicación entre los módulos relacionados. Cada prueba permitió verificar una función específica y, al mismo tiempo, observar el intercambio de información entre el dashboard, la interfaz del estudiante, los modelos de reconocimiento, la base de datos y los reportes.

Las pruebas se identificaron mediante el prefijo PF y se desarrollaron siguiendo el flujo normal de uso del sistema: acceso, selección del estudiante, configuración de la actividad, reconocimiento de señas, retroalimentación, almacenamiento y consulta de resultados.

4.3.1.1 Acceso y autenticación

Para comprobar la seguridad y el control de usuarios del sistema, se realizó una prueba del módulo de inicio de sesión. El objetivo de esta evaluación fue verificar tres aspectos clave: el bloqueo ante credenciales incorrectas, el acceso de un docente estándar (con permisos limitados) y el acceso del administrador (con control total del sistema).

En la primera parte de la prueba, se evaluó la respuesta del sistema ante un intento de ingreso fallido. Se introdujo un usuario existente, pero con una contraseña incorrecta, para confirmar que el software protege la información. Como se observa en la Figura 58, el sistema detectó la anomalía, bloqueó el acceso y mostró una alerta clara en la pantalla.

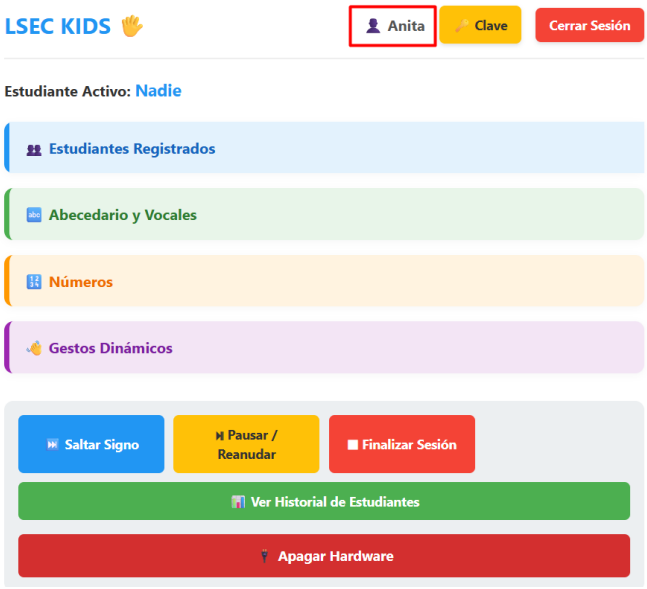
Figura 58
Bloqueo de acceso por credenciales inválidas



Nota. Elaboración propia, 2026.

Después se inició sesión mediante una cuenta de docente. El acceso fue aceptado y se mostraron las funciones necesarias para seleccionar estudiantes, configurar las categorías y controlar la sesión. La vista correspondiente al rol docente se presenta en la Figura 59.

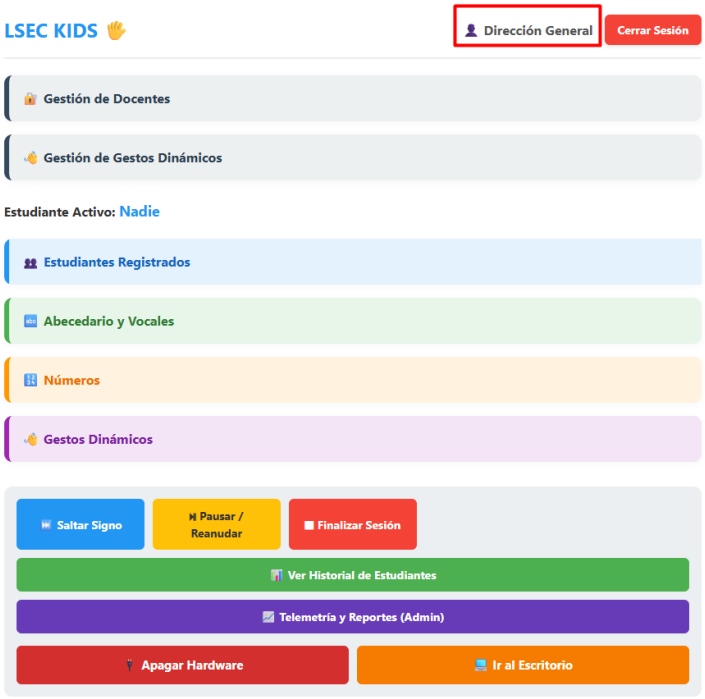
Figura 59
Interfaz del panel principal con el rol de Docente



Nota. Elaboración propia, 2026.

Finalmente, se ingresó con una cuenta de administrador. En este caso, el dashboard presentó opciones adicionales relacionadas con la gestión de usuarios, la configuración de gestos dinámicos, la consulta general de resultados y el acceso a la información del sistema, como se muestra en la Figura 60.

Figura 60
Interfaz del panel principal con el rol de Administrador



Nota. Elaboración propia, 2026.

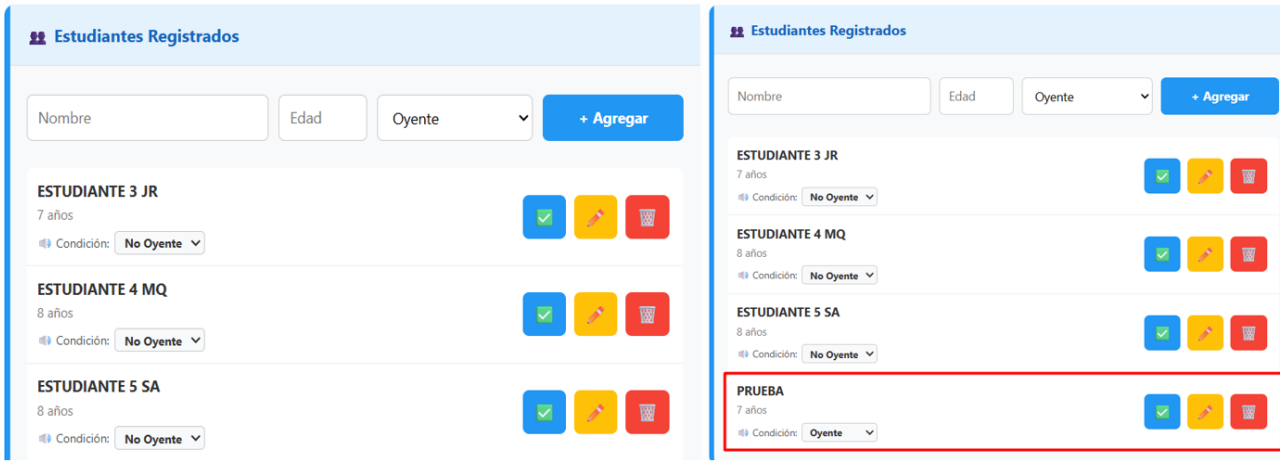
La prueba permitió comprobar que el sistema rechaza las credenciales incorrectas y presenta las funciones correspondientes al usuario autenticado. También se verificó la integración entre el formulario de acceso, la sesión administrada mediante Flask y las opciones del dashboard

4.3.1.2 Gestión y selección del estudiante

Para comprobar que el módulo del docente funciona correctamente, se realizó una prueba completa de la gestión de estudiantes. El objetivo fue verificar que el sistema permita registrar a un niño, editar su información, seleccionarlo para realizar la práctica y, sobre todo, que todos estos cambios se guarden de forma segura en la base de datos.

En primer lugar, se registró un estudiante de prueba utilizando el formulario del panel. Como se observa en la Figura 61, el sistema procesó los datos correctamente y agregó al nuevo niño a la lista de estudiantes registrados.

Figura 61
Registro inicial de usuario.



Nota. Elaboración propia.

A continuación, se probó la funcionalidad para modificar los datos. Al usar el botón de edición, la página muestra pequeños cuadros de diálogo que piden ingresar el nuevo nombre y la edad. Como se ve en la **Figura *62***, luego de ingresar los nuevos valores, la tarjeta del estudiante se actualizó de inmediato mostrando el nombre "PRUEBA_MODIFICADO" y la nueva edad.

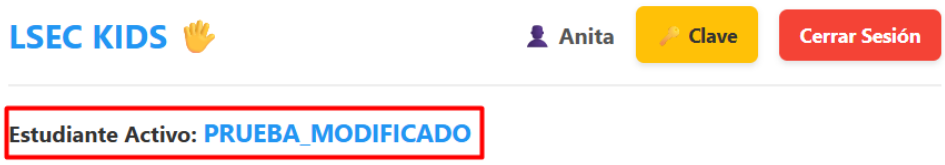
Figura 62
Edición de los datos del estudiante mediante cuadros de diálogo



Nota. Elaboración propia.

Una vez validados los datos, se procedió a seleccionar al estudiante. Este paso es fundamental porque el software necesita saber a qué niño le va a guardar las calificaciones cuando use la cámara. Como se muestra en la Figura 63, al seleccionarlo, el nombre del estudiante se fija en la parte superior de la pantalla confirmando que ahora es el usuario activo.

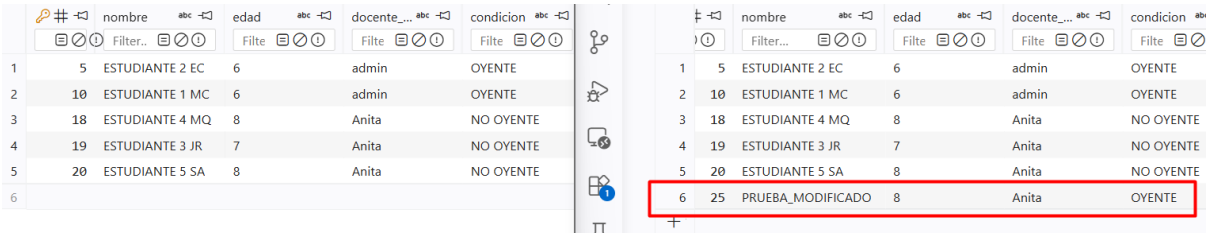
Figura 63
Confirmación del estudiante activo en el panel principal



Nota. Elaboración propia.

Para comprobar que la modificación no correspondiera únicamente a un cambio visual, se consultó la tabla de estudiantes en SQLite. La Figura 64 muestra que la información actualizada quedó almacenada en la base de datos.

Figura 64
Verificación de los datos guardados en la base de datos SQLite



	id	nombre	edad	docente_...	condicion
1	5	ESTUDIANTE 2 EC	6	admin	OYENTE
2	10	ESTUDIANTE 1 MC	6	admin	OYENTE
3	18	ESTUDIANTE 4 MQ	8	Anita	NO OYENTE
4	19	ESTUDIANTE 3 JR	7	Anita	NO OYENTE
5	20	ESTUDIANTE 5 SA	8	Anita	NO OYENTE
6					

	id	nombre	edad	docente_...	condicion
1	5	ESTUDIANTE 2 EC	6	admin	OYENTE
2	10	ESTUDIANTE 1 MC	6	admin	OYENTE
3	18	ESTUDIANTE 4 MQ	8	Anita	NO OYENTE
4	19	ESTUDIANTE 3 JR	7	Anita	NO OYENTE
5	20	ESTUDIANTE 5 SA	8	Anita	NO OYENTE
6	25	PRUEBA_MODIFICADO	8	Anita	OYENTE

Nota. Elaboración propia.

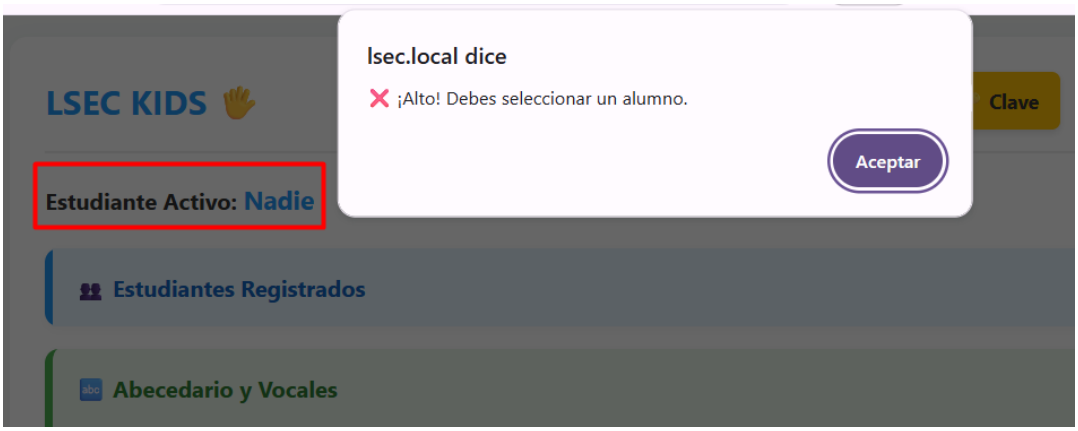
Los resultados permitieron comprobar el registro, modificación y selección del estudiante. Además, se verificó la integración entre las opciones de gestión del dashboard y la base de datos SQLite.

4.3.1.3 Configuración e inicio de la actividad

En esta prueba se evaluó la capacidad del sistema para preparar e iniciar una sesión de práctica. El objetivo principal fue verificar que el panel del docente permita configurar los parámetros de la clase (selección del alumno y elección de señas) y que estas instrucciones se reflejen correctamente en la pantalla física del hardware.

Primero se intentó iniciar una sesión sin seleccionar previamente a un estudiante. Como se observa en la Figura 65, el sistema presentó una advertencia y evitó el inicio de la actividad.

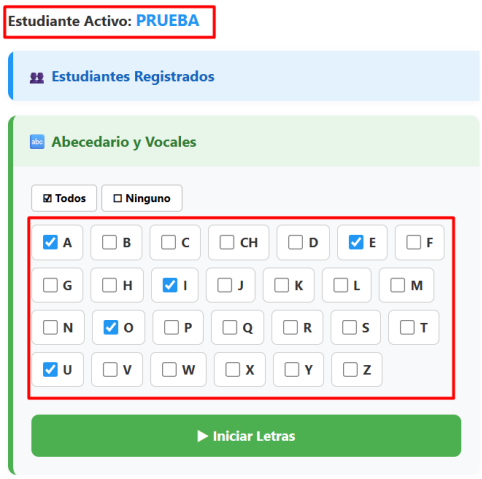
Figura 65
Bloqueo preventivo por falta de selección de estudiante



Nota. Elaboración propia.

Después se seleccionó al estudiante y se configuraron las señas que serían utilizadas durante la sesión. El dashboard permite escoger elementos de las categorías de abecedario, números y gestos dinámicos. La selección realizada se presenta en la Figura 66.

Figura 66
Configuración de las señas de la actividad



Nota. Elaboración propia.

Al iniciar la actividad, la información configurada desde el dashboard se reflejó en la pantalla del prototipo. Como se muestra en la Figura 67, la interfaz presentó el nombre del estudiante activo y quedó preparada para comenzar la práctica.

Figura 67
Sincronización con la interfaz del estudiante



Nota. Elaboración propia.

Posteriormente, la interfaz mostró los recursos visuales correspondientes a las categorías seleccionadas. Como se evidencia en laFigura 68, las señas objetivo aparecieron acompañadas de una imagen de apoyo, de acuerdo con la configuración realizada previamente por el docente.

Figura 68
Despliegue de los recursos pedagógicos visuales en la fase de aprendizaje



Nota. Elaboración propia.

Los resultados permitieron comprobar que el sistema evita iniciar una actividad sin un estudiante activo, recibe la configuración realizada desde el dashboard y carga los recursos

visuales correspondientes en la interfaz. De esta manera, se verificó la integración entre el panel docente y la pantalla del prototipo.

4.3.1.4 Captura y detección de la mano

En esta fase se evaluó el funcionamiento del módulo de visión artificial encargado de capturar el entorno y detectar la presencia de una mano en el encuadre. El sistema procesa el flujo de video en tiempo real mediante los algoritmos de detección espacial.

Al iniciar la actividad sin colocar la mano frente a la cámara, el sistema mantuvo la detección en espera. Como se observa en la Figura 69 la interfaz mostró el símbolo de interrogación debido a que no existía información suficiente para iniciar el reconocimiento.

Figura 69
Captura de video sin presencia de la mano



Nota. Elaboración propia.

Posteriormente, se colocó la mano dentro del campo de visión de la cámara. La Figura 70 muestra que el sistema inició el procesamiento al detectar su presencia. Los landmarks son utilizados internamente, pero no se muestran en la interfaz final para evitar elementos visuales que puedan distraer al estudiante.

Figura 70
Captura de video con presencia de la mano



Nota. Elaboración propia.

Los resultados permitieron verificar la comunicación entre la cámara IMX219, OpenCV, MediaPipe Hands y la interfaz del estudiante.

4.3.1.5 Reconocimiento de señas estáticas

Esta prueba se realizó para comprobar el funcionamiento de los modelos de reconocimiento del abecedario y los números dentro del prototipo. Para ello, se configuraron actividades de ambas categorías y se realizaron las señas solicitadas frente a la cámara.

En la actividad del abecedario, el sistema procesó las características de la mano mediante el modelo TensorFlow Lite y comparó el resultado con la letra solicitada. Cuando ambas coincidieron, la interfaz presentó el mensaje “MUY BIEN” y actualizó la barra de progreso. El resultado de esta prueba se muestra en la Figura 71.

Figura 71
Reconocimiento de una seña del abecedario



Nota. Retroalimentación presentada después del reconocimiento de una letra. Elaboración propia.

El mismo procedimiento se aplicó para la categoría de números. Como se observa en la Figura 72, al reconocer correctamente el número solicitado, la interfaz presentó la confirmación y completó el progreso de la actividad.

Figura 72
Reconocimiento de una seña numérica



Nota. Elaboración propia.

Esta prueba permitió comprobar la integración entre MediaPipe Hands, los modelos TensorFlow Lite, los archivos de etiquetas y la interfaz del estudiante.

4.3.1.6 Reconocimiento de palabras dinámicas

Esta prueba se realizó para comprobar el funcionamiento del modelo de reconocimiento de gestos dinámicos dentro del prototipo. En esta categoría, el sistema debe procesar una secuencia de movimientos antes de mostrar el resultado.

Durante la prueba, se seleccionó la palabra “SÍ” desde el dashboard y se realizó el gesto frente a la cámara. Luego, se verificó que el sistema procesara correctamente la secuencia de movimiento y comparara el resultado con la palabra solicitada.

Como se muestra en la Figura 73, al reconocer el gesto de forma correcta, la interfaz presentó el mensaje “MUY BIEN” y completó la barra de progreso.

Figura 73
Reconocimiento de gestos dinámicos en tiempo real



Nota. Elaboración propia.

La prueba permitió comprobar la integración entre la cámara, el procesamiento de la secuencia, el modelo Keras y la interfaz del estudiante. Su propósito fue verificar que el reconocimiento dinámico pudiera ejecutarse dentro del sistema

4.3.1.7 Retroalimentación y controles

Esta prueba se realizó para evaluar la retroalimentación presentada al estudiante y el funcionamiento de los controles disponibles durante la actividad. También se verificó que estas opciones permitan orientar al niño y facilitar la continuidad de la sesión cuando se presenten dificultades.

Cuando la seña solicitada no se completó después de varios intentos, la interfaz volvió a presentar la imagen de referencia como apoyo. Como se observa en la Figura 74, esta ayuda muestra la postura que debe realizarse sin interrumpir la captura de video.

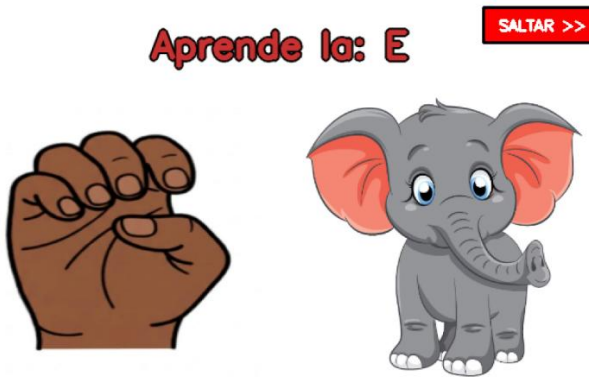
Figura 74
Ayuda visual presentada durante la actividad



Nota. Elaboración propia.

También se comprobó la opción SALTAR, utilizada cuando el estudiante no logra completar la seña o desea continuar con la siguiente actividad. Al seleccionar esta opción, el sistema cerró la actividad actual con el estado SALTO y presentó el siguiente recurso visual, como se muestra en la Figura 75

Figura 75
Continuación de la actividad después de utilizar la opción de salto



Nota. Elaboración propia.

Durante la misma prueba se verificaron las opciones de pausa y finalización disponibles en el dashboard docente. La pausa detuvo temporalmente el avance de la sesión, mientras que la finalización cerró la actividad activa y devolvió el sistema a su estado de espera. Estas opciones permiten que la docente mantenga el control de la práctica desde su dispositivo.

Los resultados permitieron comprobar la integración entre la interfaz del estudiante, los controles del dashboard y el estado de la actividad. La ayuda visual se presentó cuando fue necesaria y la opción de salto permitió avanzar sin detener la sesión.

4.3.1.8 Registro del resultado

El objetivo de esta prueba fue verificar el almacenamiento de la información generada al finalizar una actividad. El sistema guarda en la base de datos SQLite datos como el estudiante, el número de sesión, la categoría, la seña solicitada, el estado final, el nivel de confianza y el tiempo empleado. El estado se registra como EXITO cuando la seña se completa correctamente y como SALTO cuando el estudiante utiliza esta opción para continuar. No se almacenan las predicciones intermedias ni cada intento fallido.

Para verificar el registro, se realizaron actividades de las tres categorías y posteriormente se consultó la tabla registro_practicas. Como se observa en la Figura 76, las actividades finalizadas quedaron asociadas con el estudiante, la sesión y la categoría correspondiente.

Figura 76
Base de Datos

nombre	signo	est...	tiem...	categoria	confianz...
prue				Filter...	Filte
PRUEBA	A	EXITO	13.3	Abecedario y Vocales	99.361896514...
PRUEBA	A	EXITO	13.8	Abecedario y Vocales	78.184813261...
PRUEBA	A	EXITO	10.1	Abecedario y Vocales	95.863288640...
PRUEBA	A	EXITO	9.4	Abecedario y Vocales	91.060203313...
PRUEBA	A	EXITO	8.3	Abecedario y Vocales	88.814139366...
PRUEBA	A	EXITO	15.5	Abecedario y Vocales	92.647659778...
PRUEBA	A	SALTO	2.3	Abecedario y Vocales	0
PRUEBA	A	SALTO	9.5	Abecedario y Vocales	0
PRUEBA	E	SALTO	12.2	Abecedario y Vocales	0
PRUEBA	I	SALTO	1.4	Abecedario y Vocales	0
PRUEBA	O	SALTO	1.1	Abecedario y Vocales	0
PRUEBA	U	SALTO	0.5	Abecedario y Vocales	0
PRUEBA	1	SALTO	22.8	Números	0
PRUEBA	1	EXITO	16.7	Números	84.716659784...
PRUEBA	2	SALTO	33.1	Números	0
PRUEBA	3	EXITO	6.6	Números	99.996733665...
PRUEBA	4	EXITO	9	Números	87.714171409...
PRUEBA	5	EXITO	7.5	Números	99.328881502...
PRUEBA	1	EXITO	7	Números	99.822551012...
PRUEBA	ADIOS	EXITO	20.7	Gestos Dinámicos	100
PRUEBA	NO	SALTO	2.9	Gestos Dinámicos	0
PRUEBA	SI	SALTO	0.7	Gestos Dinámicos	0

Nota. Elaboración propia.

Cada fila almacena el nombre del estudiante, la seña solicitada, el estado final, el tiempo empleado, la fecha, el número de sesión, la categoría y la confianza obtenida durante el reconocimiento. Esta información permite relacionar los resultados con la actividad realizada y consultarlos posteriormente desde el dashboard.

El registro se genera únicamente cuando la actividad termina con el estado EXITO o SALTO. Por esta razón, la base de datos no almacena cada predicción incorrecta, el número de intentos individuales ni la clase predicha durante el procesamiento.

La prueba permitió comprobar la integración entre la interfaz de práctica y SQLite. Los resultados finales se almacenaron con los campos correspondientes y permanecieron disponibles para su consulta.

4.3.1.9 Consulta de resultados e historial

En esta evaluación se verificó que los resultados guardados durante las actividades pudieran consultarse posteriormente desde el dashboard. Para ello, se revisaron el reporte matricial, el detalle por seña y el historial de sesiones de un estudiante.

El reporte matricial presenta una vista general de las señas practicadas y permite identificar su estado mediante indicadores visuales. Esta información se genera a partir de los registros almacenados en SQLite y facilita la revisión del desempeño sin necesidad de consultar directamente la base de datos. Como se observa en la Figura 77 los resultados se organizan por estudiante y por seña.

Figura 77
Consulta de resultados mediante el reporte matricial



Nota. Elaboración propia.

Además de la vista general, el dashboard permite seleccionar una seña y consultar su información específica. En esta ventana se presentan la cantidad de actividades completadas, los saltos registrados y el tiempo promedio empleado. También se muestra un estado general calculado mediante los criterios configurados en el prototipo.

La Figura 78 muestra el detalle correspondiente a una de las señas practicadas. La opción disponible en esta ventana permite agregarla a las actividades de refuerzo cuando la docente considera que necesita mayor práctica.

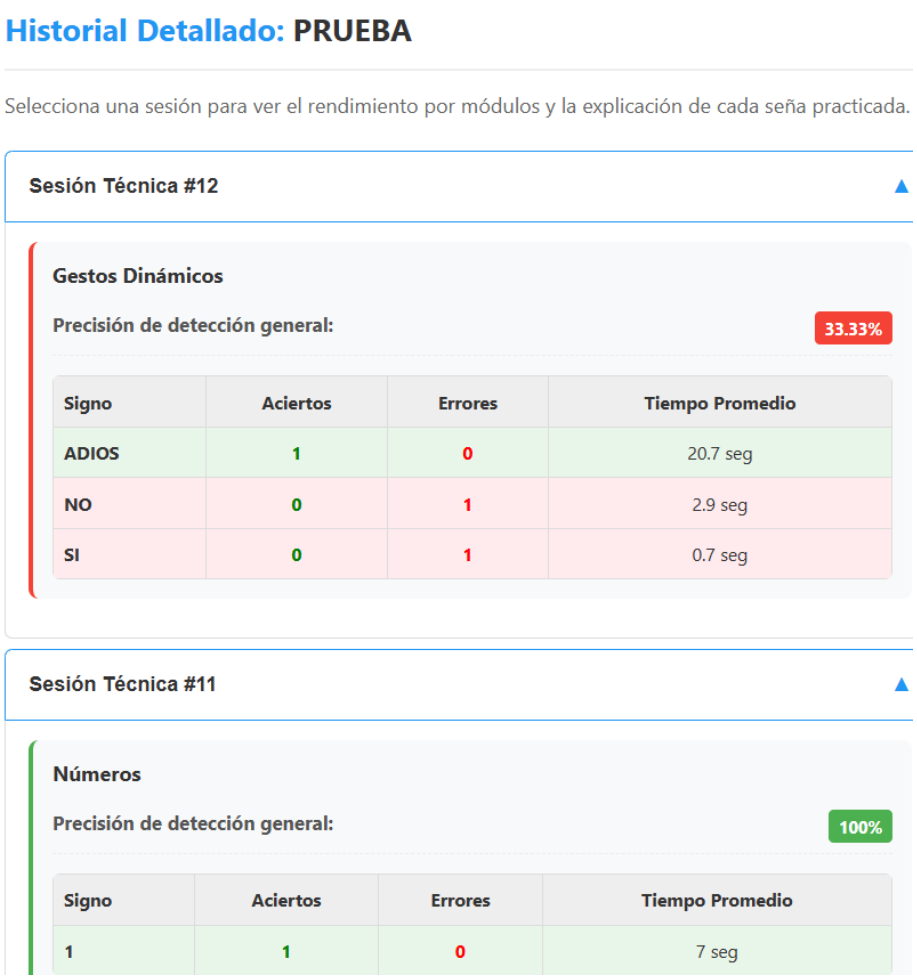
Figura 78
Detalle de los resultados obtenidos en una seña



Nota. Elaboración propia.

Finalmente, se consultó el historial detallado del estudiante, esta opción aparece al momento de aplastar sobre el nombre del estudiante. Esta opción organiza la información por sesión y categoría, permitiendo revisar las señas realizadas, los estados obtenidos y el tiempo empleado. Como se presenta en la Figura 79, los resultados de las sesiones anteriores permanecen disponibles para su consulta.

Figura 79
Historial detallado de las actividades realizadas



Nota. Elaboración propia.

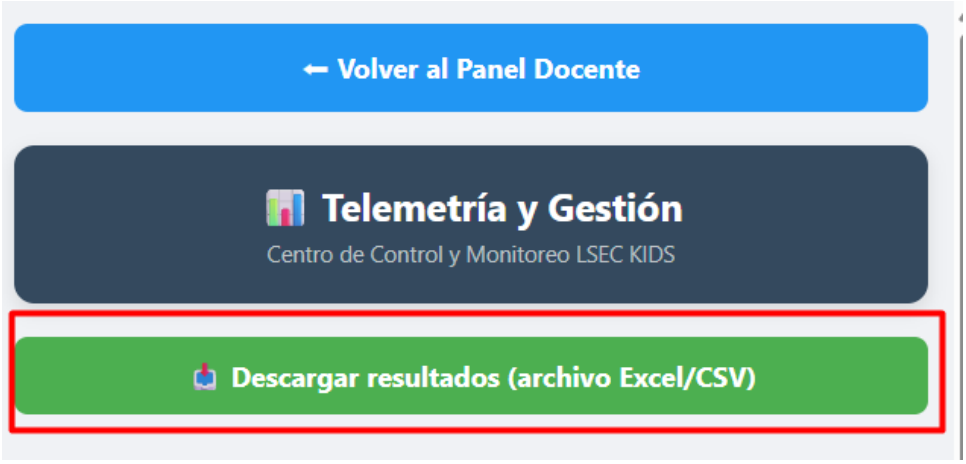
La información mostrada en el reporte matricial, el detalle por seña y el historial correspondió con los registros almacenados en SQLite. Esto permitió comprobar la integración entre la base de datos, el servidor Flask y las opciones de consulta disponibles en el dashboard

4.3.1.10 Exportación administrativa de resultados

Con el propósito de comprobar la correcta generación del archivo CSV disponible en el modo administrador, se verificó que la información exportada coincidiera con los registros almacenados en SQLite. Esta funcionalidad fue implementada para facilitar el análisis y la validación de los resultados, por lo que no constituye un reporte de uso cotidiano para la docente.

La exportación se realiza desde el panel administrativo. Como se muestra en la Figura 80, este panel incluye la opción para descargar los resultados generados durante las actividades.

Figura 80
Opción administrativa para descargar los resultados



Nota. Elaboración propia.

Después de seleccionar la opción de descarga, el sistema genera un archivo en formato CSV. Este archivo agrupa los resultados por seña e incluye la seña esperada, la cantidad de aciertos, los saltos, el total de actividades y el porcentaje de actividades completadas. La estructura del archivo obtenido se presenta en la Figura 81.

Figura 81
Archivo CSV generado para el análisis de resultados

	A	B	C	D	E
1	Signo (Espe	Aciertos (Ve	Fallos / Salt	Total de Int	Precision del
2	0	11	1	12	91.67
3	1	18	1	19	94.74
4	2	16	3	19	84.21
5	3	15	2	17	88.24
6	4	15	1	16	93.75
7	5	15	1	16	93.75
8	6	10	0	10	100.0
9	7	10	0	10	100.0
10	8	10	0	10	100.0
11	9	10	0	10	100.0
12	A	28	2	30	93.33
13	ADIOS	11	3	14	78.57
14	B	6	0	6	100.0
15	C	4	0	4	100.0
16	CH	6	0	6	100.0

Nota. Elaboración propia.

Para comprobar la correspondencia de la información, se compararon los valores del archivo CSV con los registros de la tabla registro_practicas. Antes de realizar esta comparación, se excluyeron 22 registros asociados con el usuario PRUEBA, correspondientes a 12 estados EXITO y 10 estados SALTO. De esta manera, se consideraron únicamente las 402 actividades realizadas por los participantes, a continuación, en la Tabla 56 se puede observar la comparación y diferencias que existió en cada archivo.

Tabla 56
Comparación de los resultados exportados con SQLite

Indicador comparado	SQLite	CSV	Resultado
Actividades válidas	402	402	Coincide
Estados EXITO	369	369	Coincide
Estados SALTO	33	33	Coincide
Actividades completadas	91,79%	91,79%	Coincide

Nota. Elaboración propia, 2026.

Como se observa en la **Tabla *56***, no se encontraron diferencias entre los totales almacenados en SQLite y los valores agrupados en el archivo CSV.

4.3.1.11 Resumen de las pruebas funcionales y de integración

Los resultados obtenidos de cada una de las pruebas de funcionamiento se pueden observar en la Tabla 57.

Tabla 57
Resumen de resultados de las pruebas funcionales

ID	Función comprobada	Estado
PFI-01	Acceso y autenticación	Cumple

PFI-02	Gestión del estudiante	Cumple
PFI-03	Configuración de la actividad	Cumple
PFI-04	Captura y detección	Cumple
PFI-05	Reconocimiento estático	Cumple
PFI-06	Reconocimiento dinámico	Cumple
PFI-07	Retroalimentación y controles	Cumple
PFI-08	Registro en SQLite	Cumple
PFI-09	Consulta e historial	Cumple
PFI-10	Exportación CSV	Cumple

Nota. Elaboración propia, 2026.

Los resultados permitieron comprobar el funcionamiento de las opciones evaluadas y la comunicación entre los módulos relacionados. El sistema completó el flujo desde el acceso y la configuración de la actividad hasta el almacenamiento, la consulta y la exportación de los resultados. De esta manera, se verificó el funcionamiento integrado del prototipo antes de analizar su rendimiento y los resultados obtenidos con los usuarios participantes.

4.3.2 Pruebas de rendimiento

Las pruebas de rendimiento se realizaron para observar el comportamiento de la Jetson Nano durante la ejecución continua del sistema. Se consideraron los fotogramas procesados por segundo, la temperatura y el porcentaje de memoria RAM utilizado.

La prueba se desarrolló durante 45 minutos. Las mediciones se registraron aproximadamente cada diez minutos, excepto la última, que fue tomada quince minutos después de la anterior. Durante los primeros registros se ejecutaron actividades de letras y números, mientras que en la última medición se utilizó un gesto dinámico.

La Tabla 58 presenta los valores obtenidos durante la ejecución.

Tabla 58
Mediciones registradas durante el funcionamiento continuo

Hora	Categoría	FPS	Temperatura	RAM	Observación
11:00	Letras	20	44,5 °C	61,4 %	Inicio de la observación.
11:10	Descanso	30	45,0 °C	61,6 %	Captura estable.
11:20	Letras	31	57,0 °C	61,6 %	Mayor temperatura registrada.
11:30	Números	31	53,5 °C	61,7 %	Sistema operativo.
11:45	Gestos dinámicos	9	48,5 °C	61,8 %	Disminución de FPS durante la secuencia dinámica.

Nota. Mediciones obtenidas desde el panel de monitoreo del sistema durante 45 minutos de ejecución. Elaboración propia, 2026.

En las actividades de letras y números se registraron entre 20 y 31 FPS, con un promedio de 28 FPS. Durante el reconocimiento dinámico se obtuvieron 9 FPS. Esta reducción se relaciona con el procesamiento de una secuencia de 30 fotogramas y la ejecución del modelo de palabras, que requiere analizar información temporal antes de generar el resultado.

La temperatura registrada estuvo entre 44,5 °C y 57 °C. El uso de memoria RAM varió entre 61,4 % y 61,8 %. Estos datos se presentan como una descripción del comportamiento del prototipo, debido a que en los requerimientos no se establecieron valores máximos o mínimos para estas variables.

La Figura 82 muestra la medición realizada durante el reconocimiento de una seña dinámica, momento en el que se registró la menor cantidad de FPS.

Figura 82
Monitoreo del sistema durante el reconocimiento de una seña dinámica



Nota. Registro obtenido desde el panel administrativo durante la prueba de funcionamiento continuo. Elaboración propia, 2026.

Al finalizar la prueba, el sistema continuó activo y no presentó cierres, bloqueos ni interrupciones. La Tabla 59 resume los principales resultados observados.

Tabla 59
Resumen de la prueba de funcionamiento continuo

Aspecto observado	Resultado
Tiempo de ejecución	45 minutos
Interrupciones	No se presentaron
FPS en señas estáticas	Entre 20 y 31 FPS
Promedio en señas estáticas	28 FPS
FPS en seña dinámica	9 FPS
Temperatura registrada	Entre 44,5 °C y 57 °C
Uso de memoria RAM	Entre 61,4 % y 61,8 %
Resultado general	Ejecución continua completada

Nota. Resumen de los resultados obtenidos durante la prueba. Elaboración propia, 2026.

4.4 Validación con los niños participantes

Las pruebas con usuarios se realizaron con cinco niños de 6 a 8 años, distribuidos entre participantes oyentes y no oyentes. El propósito fue observar el funcionamiento del prototipo

durante una interacción real y analizar los resultados registrados en las actividades de abecedario, números y gestos dinámicos.

Debido al número de participantes, los resultados se presentan de forma descriptiva y corresponden únicamente al grupo evaluado. La Tabla 60 resume la distribución de los usuarios y la cantidad de actividades registradas.

Tabla 60
Distribución de los participantes de la prueba

Grupo	Participantes	Edades	Actividades registradas
Oyentes	2	6 a 8 años	201
No oyentes	3	7 a 8 años	201
Total	**5**	**6 a 8 años**	**402**

Nota. Los participantes fueron identificados mediante códigos para proteger su identidad. Elaboración propia, 2026.

Durante cada sesión, la docente seleccionó al estudiante y configuró las categorías que serían practicadas. El niño observó la seña solicitada en la pantalla y la realizó frente a la cámara. El sistema registró únicamente el resultado final de cada actividad como EXITO o SALTO, junto con la categoría, la seña solicitada, la confianza y el tiempo empleado.

Antes de realizar el análisis se excluyeron 22 registros correspondientes al usuario denominado PRUEBA, debido a que fueron generados durante la comprobación técnica del sistema. Después de esta depuración quedaron 402 actividades válidas.

La **Figura *83*** presenta una de las sesiones realizadas con los usuarios.

Figura 83
Interfaz del estudiante durante el reconocimiento de una seña



Nota. Prueba realizada con uno de los usuarios participantes. Elaboración propia, 2026.

4.4.1 Resultados por categoría

Los resultados obtenidos se organizaron según las tres categorías disponibles en el sistema: abecedario y vocales, números y gestos dinámicos. Para el análisis se consideraron únicamente las 402 actividades válidas registradas después de excluir los datos del usuario PRUEBA.

El total de actividades de cada categoría se obtuvo mediante la suma de los registros con estado EXITO y SALTO:

$$\text{Actividades totales} = \text{Exitos} + \text{Saltos}$$

El porcentaje de actividades completadas se calculó dividiendo los registros con estado EXITO para el total de actividades:

$$Actividades\ completadas\ (\%) = \frac{Exitos}{Actividades\ totales} \times 100$$

Por otra parte, el tiempo promedio se obtuvo mediante la siguiente expresión:

$$Tiempo\ promedio = \frac{\sum Tiempos\ registrados}{Numero\ total\ de\ actividades}$$

Los resultados fueron redondeados a dos cifras decimales. Estos porcentajes representan el desempeño de los participantes durante el uso del prototipo y no corresponden a la exactitud obtenida durante el entrenamiento de los modelos. Como ejemplo, el porcentaje general se calculó de la siguiente manera:

$$\frac{369}{402} \times 100 = 91{,}79\%$$

La Tabla 61 muestra los resultados obtenidos por categoría.

Tabla 61
Resultados de las pruebas con usuarios por categoría

Categoría	Actividades	Éxitos	Saltos	Actividades completadas	Tiempo promedio
Abecedario y vocales	202	191	11	94,55%	9,14 s
Números	139	130	9	93,53%	8,48 s
Gestos dinámicos	61	48	13	78,69%	15,35 s
Total	**402**	**369**	**33**	**91,79%**	**9,85 s**

Nota. Elaboración propia, 2026.

El abecedario y los números presentaron los porcentajes más altos de actividades completadas, con 94,55 % y 93,53 %, respectivamente. En los gestos dinámicos se obtuvo 78,69 % y un tiempo promedio de 15,35 segundos. Esta categoría requirió que los usuarios completaran una

secuencia de movimiento, por lo que presentó más saltos y un mayor tiempo de ejecución que las señas estáticas.

En total, los participantes completaron correctamente 369 de las 402 actividades registradas, lo que corresponde al 91,79 %. Las 33 actividades restantes fueron omitidas mediante la opción de salto.

4.4.2 Resultados individuales

Los resultados individuales se organizaron según las categorías practicadas por cada niño. La cantidad de actividades no fue igual para todos los participantes, debido a que las señas fueron configuradas durante cada sesión. Por esta razón, el análisis considera principalmente el porcentaje de actividades completadas y el tiempo promedio registrado.

Para facilitar la presentación de los resultados, los participantes fueron agrupados de acuerdo con su condición auditiva. Primero se describen los resultados de los usuarios oyentes y, posteriormente, los correspondientes a los usuarios no oyentes.

4.4.2.1 Usuarios Oyentes

El grupo de usuarios oyentes estuvo conformado por dos participantes. En cada caso se analizaron las actividades realizadas en abecedario y vocales, números y gestos dinámicos, considerando los éxitos, saltos y tiempos registrados por el sistema.

Estudiante 1

El Estudiante 1 corresponde a un participante oyente de 6 años. La Tabla 62 presenta sus resultados.

Tabla 62
Resultados individuales del Estudiante 1

Categoría	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio

Abecedario y vocales	43	42	1	97,67%	15,05 s
Números	37	36	1	97,30%	13,77 s
Gestos dinámicos	9	6	3	66,67%	21,70 s
Total	**89**	**84**	**5**	**94,38%**	**15,19 s**

Nota. Resultados obtenidos de los registros almacenados en SQLite. Elaboración propia, 2026.

El participante presentó resultados superiores al 97 % en las actividades de abecedario y números. La mayor dificultad se observó en los gestos dinámicos, donde se registraron tres saltos y un tiempo promedio mayor.

Estudiante 2

El Estudiante 2 corresponde a un participante oyente de 8 años. Sus resultados se presentan en la Tabla 63.

Tabla 63
Resultados individuales del Estudiante 2

Categoría	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y vocales	52	49	3	94,23%	7,89 s
Números	33	25	8	75,76%	6,95 s
Gestos dinámicos	27	19	8	70,37%	13,75 s
Total	**112**	**93**	**19**	**83,04%**	**9,03 s**

Nota. Resultados obtenidos de los registros almacenados en SQLite. Elaboración propia, 2026.

El porcentaje más alto se obtuvo en el abecedario. En números y gestos dinámicos se registraron ocho saltos en cada categoría, lo que redujo el resultado general al 83,04 %.

4.4.2.2 Usuarios No Oyentes

El grupo de usuarios no oyentes estuvo conformado por tres participantes de 7 y 8 años. Al igual que en el grupo anterior, los resultados se organizaron por categoría para observar el desempeño individual durante la interacción con el prototipo.

Estudiante 3

El Estudiante 3 corresponde a un participante no oyente de 7 años. La Tabla 64 presenta sus resultados.

Tabla 64
Resultados individuales del Estudiante 3

Categoría	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y vocales	58	57	1	98,28%	6,99 s
Números	14	14	0	100,00%	6,13 s
Gestos dinámicos	8	8	0	100,00%	15,24 s
Total	**80**	**79**	**1**	**98,75%**	**7,66 s**

Nota. Resultados obtenidos de los registros almacenados en SQLite. Elaboración propia, 2026.

El participante completó todas las actividades de números y gestos dinámicos. En el abecedario se registró un solo salto, alcanzando un resultado general de 98,75 %.

Estudiante 4

El Estudiante 4 corresponde a un participante no oyente de 8 años. La Tabla 65 resume los resultados obtenidos.

Tabla 65
Resultados individuales del Estudiante 4

Categoría	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y vocales	31	25	6	80,65%	7,91 s
Números	25	25	0	100,00%	6,38 s
Gestos dinámicos	8	7	1	87,50%	14,46 s
Total	**64**	**57**	**7**	**89,06%**	**8,13 s**

Nota. Resultados obtenidos de los registros almacenados en SQLite. Elaboración propia, 2026.

El participante completó todas las actividades de números. La mayor cantidad de saltos se presentó en el abecedario, mientras que en los gestos dinámicos se registró un salto.

Estudiante 5

El Estudiante 5 corresponde a un participante no oyente de 8 años. La Tabla 66 presenta sus resultados.

Tabla 66
Resultados individuales del Estudiante 5

Categoría	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y vocales	18	18	0	100,00%	7,71 s
Números	30	30	0	100,00%	6,48 s
Gestos dinámicos	9	8	1	88,89%	14,68 s
Total	**57**	**56**	**1**	**98,25%**	**8,16 s**

Nota. Resultados obtenidos de los registros almacenados en SQLite. Elaboración propia, 2026.

El participante completó todas las actividades de abecedario y números. En los gestos dinámicos se registró un salto, alcanzando un resultado general de 98,25 %.

El desglose de las actividades realizadas por cada participante, organizado por categoría y seña, se presenta en el Anexo G.

4.4.3 Comparación entre usuarios oyentes y no oyentes

Una vez revisados los resultados individuales, se realizó una comparación descriptiva entre los participantes oyentes y no oyentes. Durante las sesiones se observó que los niños no oyentes conocían previamente parte del abecedario dactilológico y mostraban mayor familiaridad al formar las señas. Los niños oyentes necesitaron observar durante más tiempo la referencia visual y repetir algunas posiciones antes de completar la actividad.

La Tabla 67 presenta los resultados de los dos grupos según la categoría practicada

Tabla 67
Resultados registrados según el grupo de usuarios

Grupo	Categoría	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio

Oyentes	Abecedario	95	91	4	95,79%	11,13 s
	Números	70	61	9	87,14%	10,56 s
	Gestos dinámicos	36	25	11	69,44%	15,74 s
	Total	**201**	**177**	**24**	**88,06%**	**11,76 s**
No oyentes	Abecedario	107	100	7	93,46%	7,38 s
	Números	69	69	0	100,00%	6,37 s
	Gestos dinámicos	25	23	2	92,00%	14,79 s
	Total	**201**	**192**	**9**	**95,52%**	**7,95 s**
Total general	**Todas**	**402**	**369**	**33**	**91,79%**	**9,85 s**

Nota. Cada grupo aportó 201 actividades, aunque la cantidad de participantes fue diferente. Se excluyeron los registros correspondientes al usuario PRUEBA. Elaboración propia, 2026.

En el abecedario, ambos grupos superaron el 93 % de actividades completadas. Aunque los participantes oyentes presentaron un porcentaje ligeramente mayor, los niños no oyentes realizaron estas actividades en un tiempo promedio de 7,38 segundos, frente a los 11,13 segundos registrados por los niños oyentes. La familiaridad con el abecedario se reflejó principalmente en el menor tiempo requerido para formar las señas.

Las diferencias fueron más visibles en números y gestos dinámicos. Los niños no oyentes completaron el 100 % de las actividades de números y el 92 % de los gestos dinámicos. En el grupo oyente se alcanzó el 87,14 % y el 69,44 %, respectivamente.

En términos generales, el grupo no oyente presentó un 95,52 % de actividades completadas, mientras que el grupo oyente alcanzó el 88,06 %. La diferencia fue de 7,46 puntos porcentuales. Los niños no oyentes también necesitaron, en promedio, 3,81 segundos menos para completar las actividades.

El sistema utilizó los mismos modelos y el mismo procedimiento de reconocimiento para los dos grupos. Por tanto, la diferencia no se relaciona con una configuración distinta, sino con la manera en que las señas fueron realizadas. Una posición más definida y un movimiento más continuo pueden producir landmarks más consistentes y facilitar el reconocimiento.

Durante las sesiones se observó mayor coordinación y familiaridad en la ejecución de las señas por parte de los niños no oyentes.

En total, los cinco participantes completaron 369 de las 402 actividades válidas, lo que representa el 91,79 %. Los gestos dinámicos presentaron la mayor cantidad de saltos y el mayor tiempo promedio. La validación permitió comprobar que el prototipo pudo utilizarse con niños de las edades establecidas y registrar los resultados de sus actividades. Estos datos describen el funcionamiento del sistema durante las sesiones, pero no demuestran por sí solos una mejora en el aprendizaje.

4.5 Validación Operativa y Calidad del Sistema (ISO/IEC 25010)

Para complementar las pruebas técnicas y la validación realizada con los niños, se aplicó una encuesta a la docente que acompañó las sesiones. El instrumento estuvo compuesto por nueve preguntas relacionadas con la facilidad de uso, utilidad de las funciones, motivación, tiempo de respuesta, presentación visual, seguridad física y funcionamiento mediante la red local.

Para organizar los resultados se tomaron como referencia algunas características del modelo de calidad ISO/IEC 25010, entre ellas la idoneidad funcional, usabilidad, eficiencia de desempeño y portabilidad. La valoración de la seguridad durante el uso se relacionó con la calidad en uso. Esta aplicación se utilizó únicamente como guía para clasificar las respuestas y no como una certificación del cumplimiento de la norma.

El instrumento tuvo un enfoque mixto. Las preguntas 1 y 3 utilizaron una escala de 1 a 5, mientras que las demás recopilaron respuestas afirmativas y observaciones abiertas. Debido a que la encuesta fue respondida por una sola docente, los resultados se analizaron de forma descriptiva.

Para resumirlos se establecieron tres niveles de valoración cualitativa:

- **Alto:** respuesta favorable, calificación de 4 o 5 o aceptación sin dificultades importantes.
- **Medio:** funcionamiento aceptado, pero con una observación o necesidad clara de mejora.
- **Bajo:** respuesta desfavorable, calificación de 1 o 2 o dificultad que limita el uso de la función.

La **Tabla *68*** presenta la relación entre las respuestas de la docente, los atributos de calidad considerados y la valoración cualitativa obtenida.

Tabla 68
Resultados de la validación operativa bajo el modelo de calidad de software ISO/IEC 25010

Atributo considerado	Preguntas	Evidencia obtenida	Valoración cualitativa	Interpretación
Idoneidad funcional	3 y 9	La función “Signos a reforzar” obtuvo 2/5. La docente consideró que el sistema puede ser productivo, pero sugirió incorporar más temas, gráficos, niveles de dificultad y actividades de evaluación.	**Medio**	Las funciones principales permitieron desarrollar las actividades. Sin embargo, la presentación de “Signos a reforzar” debe mejorarse para que la información resulte más útil y clara para la docente.
Usabilidad	1, 4 y 6	El manejo del panel recibió 5/5 y la interacción fue calificada como “muy motivativa”. También se recomendó utilizar colores más llamativos e incorporar elementos visuales como caritas.	**Alto**	El panel fue considerado fácil de manejar y la interfaz generó interés en los niños. Las observaciones se relacionan con mejoras visuales y no con dificultades que impidan utilizar el sistema.
Eficiencia de desempeño	2 y 5	La docente indicó que las opciones Pausar, Saltar Signo y Finalizar Sesión respondieron de manera adecuada. También consideró apropiado el tiempo disponible para formar la seña.	**Alto**	Los controles y el tiempo de espera permitieron desarrollar la actividad sin retrasos que dificultaran la interacción.

Atributo considerado	Preguntas	Evidencia obtenida	Valoración cualitativa	Interpretación
Portabilidad y adaptabilidad	8	El funcionamiento mediante una red Wi-Fi local fue considerado de gran ayuda cuando no existe acceso a internet.	**Alto**	La respuesta respalda la utilidad del procesamiento local y del acceso al dashboard dentro de la misma red, sin depender permanentemente de servicios externos.
Calidad en uso y seguridad	7	La posición de la cámara, pantalla, placa y cables fue valorada como segura y cómoda con la respuesta “Sí, 100 %”.	**Alto**	La organización física del prototipo fue considerada adecuada para las sesiones realizadas con los niños.

Nota. Elaboración propia con base en los 9 ítems tabulados del instrumento de encuesta docente detallado en el Anexo F

Como se observa en la **Tabla *68***, cuatro de los cinco atributos considerados alcanzaron una valoración alta. La docente evaluó favorablemente la facilidad de manejo del panel, la motivación generada durante las actividades, la respuesta de los controles, el tiempo disponible para realizar las señas, el funcionamiento sin internet y la disposición física del prototipo.

La idoneidad funcional obtuvo una valoración media debido a la calificación de 2/5 asignada a “Signos a reforzar”. Aunque esta opción presenta las señas que necesitan mayor práctica, su utilidad no fue suficientemente clara para la docente. Por esta razón, se considera necesario mejorar la organización y presentación de esta información en una versión posterior.

Las respuestas abiertas también permitieron identificar posibles mejoras. Entre ellas se encuentran el uso de colores más llamativos, la incorporación de caritas u otros elementos de retroalimentación, el aumento progresivo de la dificultad y la inclusión de nuevos temas, gráficos y actividades de evaluación. Estas recomendaciones no corresponden a fallas que impidieran utilizar el prototipo, sino a propuestas para ampliar su aplicación pedagógica.

En general, la valoración docente mostró que el sistema pudo utilizarse en el contexto evaluado y que sus controles, interfaz y funcionamiento local fueron aceptados favorablemente. Sin embargo, los resultados deben interpretarse de manera descriptiva debido a que la encuesta fue respondida por una sola docente.

Para culminar la fase de pruebas, se evaluó el sistema desde la perspectiva del usuario final (docente) utilizando un instrumento estructurado bajo los lineamientos del estándar de calidad de software ISO/IEC 25010. Esta evaluación permitió medir la idoneidad funcional, usabilidad y eficiencia del prototipo lúdico en su entorno real de aplicación.

El instrumento de evaluación aplicado constó de preguntas con un enfoque mixto. Para medir los niveles de satisfacción, usabilidad e idoneidad, se utilizó una escala de valoración cuantitativa (escala de Likert del 1 al 5), complementada con respuestas cualitativas abiertas para justificar la puntuación. Como consta en el documento original respaldado en el **Anexo F**, la herramienta obtuvo calificaciones máximas (5/5) en los rubros de operatividad y utilidad pedagógica por parte del personal docente. Estas puntuaciones cuantitativas, junto con las observaciones y comentarios recabados durante la entrevista, han sido tabulados e interpretados técnicamente en la **Tabla *68*** bajo los atributos de calidad de la norma.

En conclusión, los resultados de la validación operativa y las pruebas técnicas de los algoritmos de clasificación demuestran que el sistema no solo es computacionalmente robusto, sino que posee una alta viabilidad pedagógica. La herramienta soluciona efectivamente las dificultades de atención y

evaluación individual diagnosticadas al inicio de la investigación, cumpliendo a cabalidad con el objetivo principal del proyecto.

CONCLUSIONES

- El dispositivo desarrollado reunió en una misma propuesta la presentación de recursos visuales, la práctica guiada de las señas, el reconocimiento y la retroalimentación en pantalla. Esta dinámica permitió que los niños de 6 a 8 años interactuaran directamente con el prototipo y realizaran actividades de abecedario, números y palabras. Por su alcance, el sistema constituye una herramienta de apoyo para la práctica del léxico dactilológico, sin reemplazar el acompañamiento de la docente ni la enseñanza formal de la lengua de señas.
- La evaluación de los modelos mostró que cada categoría necesitó una cantidad diferente de entrenamiento. El abecedario y las palabras alcanzaron su mayor F1-score con 200 épocas, obteniendo 99,06 % y 96,61 %, respectivamente. El modelo de números presentó su mejor resultado con 100 épocas y un F1-score de 99,30 %, mientras que al aumentar a 200 épocas su desempeño disminuyó ligeramente. Esto permitió comprobar que utilizar más épocas no siempre mejora el reconocimiento y que la configuración debe seleccionarse según el comportamiento de cada modelo.
- El procesamiento local permitió ejecutar la captura, el reconocimiento, la retroalimentación y el registro de las actividades sin depender de Internet. Durante 45 minutos de funcionamiento continuo no se presentaron cierres ni interrupciones. Las señas estáticas se procesaron con un promedio de 28 FPS, mientras que los gestos dinámicos alcanzaron 9 FPS. Esta diferencia se relaciona con el análisis de las secuencias de movimiento, que requiere procesar varios fotogramas antes de generar un resultado.
- En las pruebas realizadas con los niños se completaron 369 de las 402 actividades válidas, lo que representa el 91,79 %. El abecedario y los números alcanzaron los porcentajes más altos, con 94,55 % y 93,53 %, mientras que los gestos dinámicos

obtuvieron 78,69 %. También se observó que los niños no oyentes, quienes tenían mayor familiaridad con las señas, completaron el 95,52 % de las actividades con un tiempo promedio de 7,95 segundos. Los niños oyentes alcanzaron el 88,06 % y necesitaron 11,76 segundos. Estos resultados indican que tanto el tipo de seña como la experiencia previa influyen en su ejecución, ya que las posiciones definidas y los movimientos continuos facilitan una respuesta más consistente del sistema.

- El registro organizado de los resultados complementó la práctica realizada con el dispositivo, al conservar la información de cada estudiante, categoría y seña. Los estados obtenidos y los tiempos registrados permitieron consultar las actividades anteriores e identificar aquellas que presentaron más saltos o necesitaron mayor tiempo. De esta manera, además de proporcionar una respuesta inmediata en la pantalla, el prototipo genera información útil para el seguimiento de las actividades realizadas.

RECOMENDACIONES

- Dado que el reconocimiento de landmarks depende del contraste entre la mano y el fondo, se recomienda implementar el prototipo en aulas con iluminación controlada o luz natural uniforme. Para futuras versiones, se sugiere evaluar la integración de cámaras con sensores infrarrojos (IR) o de profundidad (como Intel RealSense) para reducir la tasa de falsos negativos en entornos con contraluz o sombras severas.
- Se sugiere que futuros trabajos de investigación amplíen el dataset construido en este proyecto. Al incorporar a múltiples usuarios de distintas edades y fisonomías realizando las señas, se aumentará la varianza de los datos de entrenamiento, lo que permitirá que las redes DNN y LSTM mejoren su capacidad de generalización frente a manos de diferentes tamaños.
- Para escalar el sistema a vocabularios dinámicos mucho más extensos (frases completas), se recomienda compilar los modelos .h5 de TensorFlow utilizando el motor de inferencia NVIDIA TensorRT. Esto aprovechará al máximo los núcleos CUDA de la Jetson Nano, permitiendo analizar secuencias de video más largas sin comprometer los fotogramas por segundo (FPS).
- Para futuras versiones, se sugiere incorporar una retroalimentación más específica cuando una seña no sea reconocida. El sistema podría indicar si el niño debe ajustar la posición de la mano, repetir el movimiento o mejorar el encuadre, en lugar de mostrar únicamente si la respuesta fue correcta o incorrecta.
- Se recomienda utilizar los resultados registrados para adaptar las actividades según el desempeño de cada estudiante. Las señas que presenten más intentos, saltos o mayor tiempo de respuesta podrían aparecer con mayor frecuencia, permitiendo reforzar las categorías en las que cada niño necesite más práctica.

- Debido a que el sistema almacena información relacionada con niños, se sugiere incorporar mecanismos de seguridad para controlar el acceso al dashboard y proteger los resultados registrados.

REFERENCIAS

A.D.A.M. Medical Encyclopedia. (2024). *Hipoacusia*. MedlinePlus. https://medlineplus.gov/spanish/ency/article/003044.htm

ADATA Technology Co. Ltd. (n.d.). *SD810 External SSD*. Retrieved https://www.adata.com/ec/consumer/category/ssds/external-solid-state-drive-sd810/

Alliance Memory. (2023). *16GB eMMC Storage Solution*. https://www.alliancememory.com/new-16gb-emmc-solution/

Amazon. (n.d.-a). *AC to DC 5V 4A Power Supply Adapter, Plug 5.5mm x 2.1mm*. Retrieved https://www.amazon.com/dp/B09JW28WQC

Amazon. (n.d.-b). *Ingcool 7 inch HDMI LCD 1024x600 Resolution Capacitive Touch Screen IPS Display Module*. Retrieved https://www.amazon.com/dp/B08H8HZRLQ

Amazon. (n.d.-c). *Pibiger IMX219 Camera Module with FOV62 M12 Lens for Raspberry Pi*. Retrieved https://www.amazon.com/-/es/dp/B0CGPT72NS

Asamblea Nacional del Ecuador. (2024). *La inclusión de las personas con discapacidad fue fortalecía*. https://asambleanacional.gob.ec

ASUSTeK Computer Inc. (n.d.). *Coral Dev Board*. Retrieved https://www.asus.com/networking-iot-servers/aiot-industrial-solutions/gpu-edge-ai-accelerators/coral-dev-board/

Consejo Nacional para la Igualdad de Discapacidades. (n.d.). *Estadísticas de Discapacidad*. Retrieved https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/

Consejo Nacional para la Igualdad de Discapacidades. (2018). *Estadísticas de discapacidad en Ecuador*. https://consejodiscapacidades.gob.ec

Coral. (2019). *Dev Board datasheet*. https://dlcdnets.asus.com/pub/ASUS/mb/AIOT/AI_Accelerator/Coral_Dev_Board_datasheet.pdf

Didácticas Electrónicas I+D. (n.d.). *Módulo inalámbrico Wi-Fi, Bluetooth para Jetson Nano*. Retrieved https://www.didacticaselectronicas.com/shop/ws-16578-modulo-inalambrico-wi-fi-bluetooth-para-jetson-nano-15839

Elecrow. (n.d.). *Raspberry Pi 4 Model B - 4GB*. Retrieved https://www.elecrow.com/raspberry-pi-4-model-b-4gb-ram.html

Excellence Wire Ind. Co. Ltd. (n.d.). *Cable de red RJ45 SSTP Cat6 fácil con herramienta de desconexión*. Retrieved https://www.exw.com.tw/es/product/C6SSWHA102B50010.html

Goodfellow, I., Bengio, Y., & Courville, A. (2016a). *Deep learning*. MIT Press. https://deeplearningbook.org

Goodfellow, I., Bengio, Y., & Courville, A. (2016b). *Deep Learning*. MIT Press. https://www.deeplearningbook.org/

Google Cloud. (n.d.). *Cloud sustainability*. Retrieved https://cloud.google.com/sustainability

Instituto Nacional de Estadística y Censos. (n.d.). *Censo de Población y Vivienda*. Retrieved https://www.ecuadorencifras.gob.ec/censo-de-poblacion-y-vivienda/

Instituto Nacional de Estadística y Censos. (2023). *Resultados del censo de población y vivienda*. https://ecuadorencifras.gob.ec

Intel. (n.d.). *Intel RealSense Depth Camera D435*. Retrieved https://www.intel.com/content/www/us/en/products/sku/128255/intel-realsense-depth-camera-d435/specifications.html

IONOS. (n.d.). *Servicios de almacenamiento en la nube: comparativa*. Retrieved https://www.ionos.com/es-us/digitalguide/servidores/herramientas/servicios-de-almacenamiento-en-la-nube-comparativa/

Kingston Technology. (n.d.-a). *Canvas Select Plus microSD Memory Card*. Retrieved https://www.kingston.com/en/memory-cards/canvas-select-plus-microsd-card-sdcs3

Kingston Technology. (n.d.-b). *¿Cuál es la diferencia entre USB 3.0, 3.2 Gen 1 y Gen 2?* Retrieved https://www.kingston.com/es/usb-flash-drives/usb-30

LG Electronics. (n.d.). *18.5" HD Monitor - 19BA300*. Retrieved https://www.lg.com/global/business/monitors-pcs/monitors/lg-monitor/19ba300/

Logitech. (n.d.). *C270 HD Webcam*. Retrieved https://www.logitech.com/en-us/shop/p/c270-hd-webcam

Mercado Libre Ecuador. (n.d.). *Micro SD Kingston 64GB V10 100 MB/s Canvas Select Plus*. Retrieved https://www.mercadolibre.com.ec/micro-sd-kingston-64gb-v10-100-mbs-canvas-select-plus/up/MECU3507565995

Merck Manual Professional Edition. (2024). *Hearing loss*. Merck Manual. https://www.merckmanuals.com/professional/ear-nose-and-throat-disorders/hearing-loss/hearing-loss

Naciones Unidas. (2015). *Objetivos de Desarrollo Sostenible*. https://www.un.org/sustainabledevelopment/es/

Neotech. (n.d.). *Cable USB macho a macho*. Retrieved https://neotech.com.ec/tienda/cable-usb-macho-a-macho-en-quito-manta-ecuador/

NVIDIA. (2019). *Jetson Nano Developer Kit user guide*. https://developer.nvidia.com/embedded/dlc/jetson_nano_developer_kit_user_guide

Organización Mundial de la Salud. (2021). *Sordera y pérdida de la audición*. https://www.who.int/es/news-room/fact-sheets/detail/deafness-and-hearing-loss

Organización Mundial de la Salud. (2026). *Sordera y pérdida de la audición*. https://www.who.int/es/news-room/fact-sheets/detail/deafness-and-hearing-loss

OSA Electronics Company Limited. (n.d.). *NVIDIA Jetson Nano Acrylic Case Kit*. Retrieved https://www.osaelectronics.com/product/nvidia-jetson-nano-acrylic-case-kit/

Pibiger. (n.d.). *IMX219*. Retrieved https://www.pibiger-tech.com/pi-camera/imx219/

Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education. https://www.mheducation.com/highered/product/software-engineering-a-practitioners-approach-pressman.html

Raspberry Pi Ltd. (n.d.-a). *Raspberry Pi Camera Module 3 product brief*. Retrieved https://pip-assets.raspberrypi.com/categories/786-raspberry-pi-camera-module-3/documents/RP-008151-DS-1-camera-module-3-product-brief.pdf

Raspberry Pi Ltd. (n.d.-b). *Touch Display 2*. Retrieved https://www.raspberrypi.com/documentation/accessories/touch-display-2.html

Raspberry Pi Ltd. (2024). *Raspberry Pi 4 Model B datasheet*. https://pip.raspberrypi.com/documents/RP-008341-DS-raspberry-pi-4-datasheet.pdf

Raspberry Pi Ltd. (2026a). *Raspberry Pi 5*. https://www.raspberrypi.com/products/raspberry-pi-5/

Raspberry Pi Ltd. (2026b). *Raspberry Pi 5 product brief*. https://pip.raspberrypi.com/documents/RP-008348-DS-raspberry-pi-5-product-brief.pdf

Samsung Electronics Co. Ltd. (n.d.). *Galaxy Tab A9*. Retrieved https://www.samsung.com/ae/tablets/galaxy-tab-a/galaxy-tab-a9-wifi-graphite-64gb-sm-x110nzaamea/

Sánchez Fuentes, S., Jiménez Hernández, D., Sancho Requena, P., & Moreno-Medina, I. (2019). Validación de instrumento para medir las percepciones de los docentes sobre el diseño universal para el aprendizaje. *Revista Latinoamericana de Educación Inclusiva*, *13*(1), 89–103. https://doi.org/10.4067/S0718-73782019000100089

Secretaría Nacional de Planificación y Desarrollo. (2017). *Plan Nacional para el Buen Vivir 2017–2021: Planificamos para toda una vida*. https://www.gobiernoelectronico.gob.ec/wp-content/uploads/downloads/2017/09/Plan-Nacional-para-el-Buen-Vivir-2017-2021.pdf

Shenzhen Xunlong Software Co. Ltd. (2024). *Orange Pi 5 user manual*. https://orangepi.net/wp-content/uploads/2024/09/OrangePi_5_RK3588S_User-Manual_v2.1.1.pdf

TP-Link Technologies Co. Ltd. (n.d.). *TL-WN823N: Mini Adaptador USB Inalámbrico N 300Mbps*. Retrieved https://www.tp-link.com/ec/home-networking/adapter/tl-wn823n/

youyeetoo. (n.d.). *Jetson Nano B01 Subkit Nano Kit - NVIDIA Jetson Nano 4GB Alternative*. Retrieved https://youyeetoo.com/products/jetson-nano-b01-subkit-nano-kit

Yuko Technology. (n.d.). *ANKER POWER BANK 20000mAh 18W (A1363)*. Retrieved https://yukotech.ec/perifericos/134-anker-power-bank-20000mah-18w-a1363.html

ANEXOS

Anexo A: Formato Entrevista semiestructurada a usuario especialista

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA ENCIENCIAS APLICADAS

CARRERA DE TELECOMUNICACIONES

ENTREVISTA

TÍTULO:

Guía de Entrevista Semiestructurada sobre estrategias didácticas y necesidades tecnológicas en la enseñanza de léxico dactilológico para niños de 6 a 8 años.

OBJETIVO:

Recopilar información, desde la experiencia docente, sobre las metodologías actuales, las dificultades de aprendizaje y las necesidades de innovación tecnológica en la enseñanza de lengua de señas básica. Los datos obtenidos permitirán fundamentar y establecer los requerimientos técnicos para el diseño de un dispositivo lúdico interactivo basado en visión artificial.

PROPÓSITO E INSTRUCCIONES:

Estimado/a docente: El presente instrumento de investigación tiene fines estrictamente académicos y forma parte del desarrollo de un proyecto de titulación en Ingeniería en Telecomunicaciones. Su experiencia en el aula es fundamental para comprender los desafíos reales que enfrentan los niños y para garantizar que la herramienta tecnológica propuesta sea verdaderamente útil.

A continuación, se le formularán una serie de preguntas abiertas. Por favor, siéntase libre de expresar sus opiniones y experiencias con total sinceridad. La información proporcionada durante esta entrevista será tratada con absoluta confidencialidad y anonimato, utilizándose únicamente para el análisis y diseño del sistema.

Pregunta 1: ¿Qué tipo de recursos didácticos o estrategias pedagógicas utiliza actualmente para enseñar el abecedario y las señas básicas a niños de entre 6 y 8 años?

Pregunta 2: Desde su experiencia, ¿cuáles son las principales dificultades o desafíos que presentan los niños de esta edad al momento de coordinar y memorizar las posturas exactas de las manos?

Pregunta 3: Al trabajar con un grupo de niños simultáneamente, ¿qué tan complejo resulta evaluar y corregir de manera individual la ejecución precisa de cada seña?

Pregunta 4: ¿Considera que los materiales tradicionales (tarjetas, dibujos, repetición) logran mantener la atención sostenida de los niños en estas edades, o percibe que hace falta mayor interactividad?

Pregunta 5: Considerando el avance de la tecnología, ¿cree que hace falta implementar herramientas digitales interactivas enfocadas específicamente en la lengua de señas dentro del contexto educativo ecuatoriano?

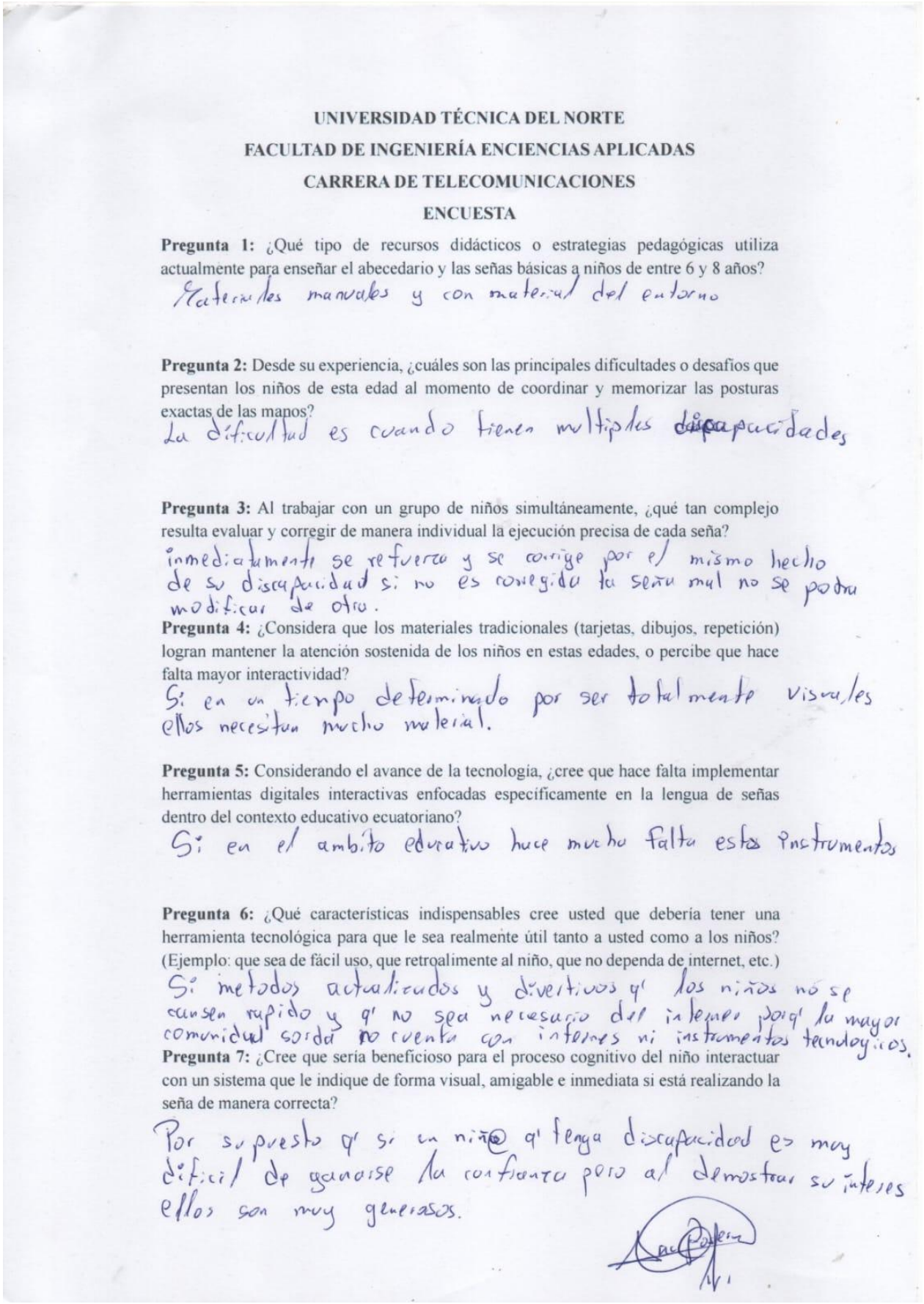
Pregunta 6: ¿Qué características indispensables cree usted que debería tener una herramienta tecnológica para que le sea realmente útil tanto a usted como a los niños? (Ejemplo: que sea de fácil uso, que retroalimente al niño, que no dependa de internet, etc.)

Pregunta 7: ¿Cree que sería beneficioso para el proceso cognitivo del niño interactuar con un sistema que le indique de forma visual, amigable e inmediata si está realizando la seña de manera correcta?

Estudiante
Srta. Alfonsina Chamorro

Director
MSc. Jaime Michilena

Anexo A.1: Imagen de entrevista



UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA ENCIENCIAS APLICADAS

CARRERA DE TELECOMUNICACIONES

ENCUESTA

Pregunta 1: ¿Qué tipo de recursos didácticos o estrategias pedagógicas utiliza actualmente para enseñar el abecedario y las señas básicas a niños de entre 6 y 8 años?

Materiales manuales y con material del entorno

Pregunta 2: Desde su experiencia, ¿cuáles son las principales dificultades o desafíos que presentan los niños de esta edad al momento de coordinar y memorizar las posturas exactas de las manos?

La dificultad es cuando tienen multiples discapacidades

Pregunta 3: Al trabajar con un grupo de niños simultáneamente, ¿qué tan complejo resulta evaluar y corregir de manera individual la ejecución precisa de cada seña?

inmediatamente se refuerza y se corrige por el mismo hecho de su discapacidad si no es corregido la seña mal no se podra modificar de otro.

Pregunta 4: ¿Considera que los materiales tradicionales (tarjetas, dibujos, repetición) logran mantener la atención sostenida de los niños en estas edades, o percibe que hace falta mayor interactividad?

Si en un tiempo determinado por ser totalmente visuales ellos necesitan mucho material.

Pregunta 5: Considerando el avance de la tecnología, ¿cree que hace falta implementar herramientas digitales interactivas enfocadas específicamente en la lengua de señas dentro del contexto educativo ecuatoriano?

Si en el ambito educativo hace mucho falta estos instrumentos

Pregunta 6: ¿Qué características indispensables cree usted que debería tener una herramienta tecnológica para que le sea realmente útil tanto a usted como a los niños? (Ejemplo: que sea de fácil uso, que retroalimente al niño, que no dependa de internet, etc.)

Si metodos actualizados y divertidos q' los niños no se cansen rapido y q' no sea necesario del internet porq' la mayor comunidad sorda no cuenta con internet ni instrumentos tecnologicos.

Pregunta 7: ¿Cree que sería beneficioso para el proceso cognitivo del niño interactuar con un sistema que le indique de forma visual, amigable e inmediata si está realizando la seña de manera correcta?

Por supuesto q' si un niñ@ q' tenga discapacidad es muy difícil de ganarse la confianza pero al demostrar su interes ellos son muy generosos.

Anexo B: Matriz de revisión de trabajos relacionados

Objetivo: Extraer criterios técnicos, funcionales y pedagógicos a partir de trabajos relacionados con reconocimiento de señas, visión artificial y herramientas educativas interactivas, con el fin de identificar requerimientos aplicables al diseño e implementación del sistema propuesto.

En el presente anexo se realiza un resumen de los trabajos relacionados analizados para el desarrollo del proyecto. Esta revisión permite identificar tecnologías, metodologías, herramientas de software, componentes de hardware, estrategias de interacción y elementos lúdicos que pueden ser considerados en la definición de los requerimientos del sistema.

Trabajo relacionado 1

Autores: Deyvis Quinta Lipe y Pablo José Guerrero

Editorial / Universidad: Revista Criterio / Universidad Pública de El Alto.

Título: "Traducción inteligente de lenguaje de señas mediante aprendizaje automático y visión artificial".

Año: 2025.

Ciudad: La Paz.

Resumen del contenido:

El trabajo presenta una propuesta orientada a la traducción inteligente de lenguaje de señas mediante el uso de aprendizaje automático y visión artificial. La investigación se enfoca en la integración de inteligencia artificial en dispositivos móviles, considerando la optimización de modelos para equipos con recursos limitados.

El estudio resalta la importancia de aplicar tecnologías inteligentes para facilitar la comunicación de personas con discapacidad auditiva, mediante sistemas capaces de procesar información visual y generar una respuesta comprensible para el usuario.

Aspectos relevantes identificados:

- Uso de aprendizaje automático aplicado a la traducción de lenguaje de señas.
- Aplicación de visión artificial para el procesamiento de información visual.
- Consideración de dispositivos móviles o plataformas con recursos limitados.
- Enfoque hacia el desarrollo de herramientas accesibles e inclusivas.
- Importancia de la eficiencia del sistema durante el procesamiento de datos.

Aporte al proyecto:

Este trabajo aporta criterios para justificar el uso de visión artificial y aprendizaje automático en el sistema propuesto. Además, permite considerar la necesidad de que el sistema sea eficiente, accesible y adaptable al entorno de uso, especialmente cuando se trabaja con recursos computacionales limitados.

Requerimientos derivados:

- El sistema debe emplear técnicas de visión artificial para el reconocimiento de gestos.
- El sistema debe procesar la información visual de manera eficiente.
- El sistema debe considerar una arquitectura accesible para el usuario final.
- El sistema debe presentar una respuesta clara a partir del gesto identificado.

Trabajo relacionado 2

Autor(es): Jimmy Alejandro Plazas López, Juan José Gutiérrez Leguizamón, Marco Javier Suárez Barón y Juan Sebastián González Sanabria

Editorial / Universidad: Revista Tecnura / Universidad Pedagógica y Tecnológica de Colombia.

Título: “Reconocimiento de lengua de señas colombiana mediante redes neuronales convolucionales y captura de movimiento”.

Año: 2022.

Ciudad: Sogamoso.

Resumen del contenido:

El artículo presenta el diseño de un modelo predictivo computacional para el reconocimiento de la Lengua de Señas Colombiana. Para ello, se emplearon técnicas de inteligencia artificial, redes neuronales profundas y captura de datos en tiempo real mediante cámaras de video de dispositivos móviles.

El modelo utiliza imágenes capturadas como conjunto de datos de entrenamiento y aplica redes neuronales convolucionales para clasificar los gestos. Además, el trabajo utiliza herramientas como TensorFlow, OpenCV y MediaPipe. Como resultado, se obtuvo un modelo capaz de identificar y traducir 39 señas diferentes, alcanzando una tasa de éxito del 97,6 % en un ambiente controlado.

Aspectos relevantes identificados:

- Captura de gestos en tiempo real mediante cámara.
- Uso de redes neuronales convolucionales para la clasificación de señas.
- Aplicación de herramientas como TensorFlow, OpenCV y MediaPipe.
- Evaluación del rendimiento del modelo mediante métricas categóricas.
- Reconocimiento de señas en un entorno controlado.

Aporte al proyecto:

Este trabajo aporta una referencia técnica importante para el reconocimiento automático de gestos mediante visión artificial. Además, respalda el uso de captura en tiempo real, redes neuronales convolucionales y herramientas de procesamiento de imágenes, aspectos

relacionados directamente con el modelo de reconocimiento que se plantea en el presente proyecto.

Requerimientos derivados:

- El sistema debe capturar los gestos del usuario mediante una cámara.
- El sistema debe realizar el reconocimiento de gestos en tiempo real.
- El sistema debe utilizar herramientas de visión artificial para el procesamiento de imágenes.
- El modelo debe ser evaluado mediante métricas de rendimiento.
- El sistema debe presentar una clasificación clara del gesto reconocido.

Trabajo relacionado 3

Autor: David Santiago Hinojosa González

Editorial / Universidad: Universidad Técnica del Norte.

Título: "Traductor de lenguaje de señas para personas con discapacidad auditiva basado en visión artificial".

Año: 2024.

Ciudad: Ibarra.

Resumen del contenido:

El trabajo desarrolla un prototipo funcional de traducción de lenguaje de señas basado en visión artificial. El sistema utiliza una placa NVIDIA Jetson Nano para realizar la detección de letras del alfabeto ASL mediante inferencia en tiempo real.

La implementación emplea Python como lenguaje de programación, junto con librerías de visión artificial como OpenCV y MediaPipe. Además, utiliza el modelo YOLOv8 entrenado con un dataset de imágenes de gestos de manos. El sistema incluye la captura de gestos

mediante cámara, procesamiento local, visualización del resultado y reproducción de la traducción mediante un sintetizador de voz.

Aspectos relevantes identificados:

- Uso de una placa NVIDIA Jetson Nano para procesamiento local.
- Captura de gestos mediante cámara.
- Uso de Python, OpenCV y MediaPipe.
- Aplicación de YOLOv8 para detección y clasificación de gestos.
- Inferencia en tiempo real.
- Uso de dataset de imágenes para entrenamiento del modelo.
- Retroalimentación visual y auditiva del resultado.

Aporte al proyecto:

Este trabajo constituye una referencia directa para el componente técnico del sistema, debido a que aborda el reconocimiento de señas mediante visión artificial, procesamiento local e inferencia en tiempo real. Además, aporta criterios para la selección de hardware, software, cámara, librerías y modelo de inteligencia artificial.

Requerimientos derivados:

- El sistema debe contar con una cámara para capturar los gestos realizados por el usuario.
- El sistema debe procesar los gestos mediante técnicas de visión artificial.
- El sistema debe ejecutar el modelo de reconocimiento de forma local.
- El sistema debe utilizar librerías compatibles con tareas de visión artificial.
- El sistema debe mostrar el resultado del reconocimiento al usuario.
- El sistema debe considerar una arquitectura capaz de realizar inferencia en tiempo real.

Trabajo relacionado 4

Autor: Julio Israel Caicedo Sandoval

Editorial / Universidad: Universidad Técnica del Norte.

Título: "Juguete electrónico lúdico como herramienta educativa para la adquisición de la lectoescritura, en niños de 5 a 8 años con necesidades especiales asociadas a la discapacidad intelectual y al trastorno del espectro autista, en la Unidad Educativa Agustín Cueva Dávila".

Año: 2019.

Ciudad: Ibarra.

Resumen del contenido:

El proyecto desarrolla una herramienta educativa con enfoque lúdico para apoyar el proceso de adquisición de la lectoescritura en niños de 5 a 8 años con necesidades educativas especiales. La herramienta integra juegos orientados al aprendizaje de habilidades metalingüísticas, utilizando recursos visuales y auditivos para estimular la participación de los niños.

El trabajo considera una metodología de enseñanza basada en actividades interactivas, repetición, recursos multimedia y evaluación observativa. Además, se enfoca en el diseño de una herramienta amigable para niños, con el objetivo de fortalecer el proceso de enseñanza-aprendizaje.

Aspectos relevantes identificados:

- Uso de actividades lúdicas como estrategia educativa.
- Aplicación de recursos visuales y auditivos.
- Enfoque en niños de 5 a 8 años con necesidades educativas especiales.
- Uso de repetición para reforzar el aprendizaje.
- Evaluación mediante instrumentos de observación.
- Diseño de una herramienta amigable y adaptada al usuario infantil.

Aporte al proyecto:

Este trabajo aporta criterios pedagógicos para el diseño de una herramienta educativa dirigida a niños de edades similares. Aunque no se enfoca en visión artificial, sirve como referencia

para incorporar elementos lúdicos, retroalimentación visual, recursos amigables y actividades interactivas dentro del sistema propuesto.

Requerimientos derivados:

- El sistema debe incluir una interfaz amigable para niños.
- El sistema debe incorporar recursos visuales que apoyen el aprendizaje.
- El sistema debe incluir actividades interactivas.
- El sistema debe permitir la repetición de ejercicios para reforzar el aprendizaje.
- El sistema debe facilitar la observación del desempeño del usuario.
- El sistema debe ser comprensible y fácil de utilizar en un entorno educativo.

Trabajo relacionado 5

Autor: Miguel Agudo Ballesteros

Editorial / Universidad: Universidad Politécnica de Madrid.

Título: "Desarrollo de un juego interactivo para niños con necesidades educativas especiales".

Año: 2016.

Ciudad: Madrid.

Resumen del contenido:

El trabajo presenta el desarrollo de un juego interactivo orientado a niños con necesidades educativas especiales. El sistema incluye diferentes modos de juego, tales como actividades de asociación, deletreo y niveles de dificultad. Además, considera la interacción mediante gestos, audio, repetición, puntuación y pruebas funcionales para verificar el comportamiento del juego.

El proyecto se enfoca en la implementación de una herramienta interactiva que permita al niño participar activamente mediante acciones dentro del entorno del juego. También contempla

respuestas ante aciertos y errores, lo que permite mantener la dinámica de interacción y reforzar el aprendizaje.

Aspectos relevantes identificados:

- Uso de modos de juego adaptados al usuario.
- Incorporación de niveles de dificultad.
- Interacción mediante gestos o acciones del usuario.
- Uso de audio para reforzar instrucciones y errores.
- Inclusión de puntuación y repetición.
- Pruebas funcionales para verificar el comportamiento del sistema.
- Separación entre lógica del juego e interfaz.

Aporte al proyecto:

Este trabajo aporta criterios para el diseño de actividades interactivas dentro del sistema propuesto. Además, permite considerar elementos como retroalimentación, repetición, niveles de dificultad, audio, puntuación y validación funcional, los cuales pueden contribuir a mantener la atención de los niños durante el proceso de aprendizaje.

Requerimientos derivados:

- El sistema debe incluir actividades interactivas para el usuario.
- El sistema debe proporcionar retroalimentación ante aciertos y errores.
- El sistema debe considerar instrucciones visuales o auditivas.
- El sistema debe permitir repetir actividades para reforzar el aprendizaje.
- El sistema debe registrar o mostrar el resultado de la interacción del usuario.
- El sistema debe contar con pruebas funcionales para verificar su correcto comportamiento.

Anexo C: Matrices comparativas para la selección del hardware

Objetivo

- Presentar las características técnicas y los criterios considerados para comparar y seleccionar los componentes de hardware utilizados en la construcción del prototipo.

Para seleccionar el hardware, se compararon los componentes necesarios para el funcionamiento del prototipo. Las matrices muestran las características principales que permitieron elegir las opciones más adecuadas para el sistema.

C.1 Criterios generales de comparación

Antes de presentar las matrices comparativas, se establecieron los criterios técnicos que permitieron analizar las alternativas de cada categoría de hardware. Estos criterios se definieron de acuerdo con las características del sistema, como el procesamiento local de los modelos, la captura de los gestos, la interacción mediante la pantalla táctil, el almacenamiento de la información y el acceso al dashboard desde la red local.

Como se muestra en la Tabla C.1, los criterios varían según la función que cumple cada componente dentro del prototipo. Esta tabla no determina todavía cuál alternativa fue seleccionada, sino que resume los aspectos utilizados posteriormente para identificar sus diferencias y comprobar su relación con los requerimientos del sistema.

Tabla C.1
Matriz de criterios considerados para la selección del hardware

Categoría	Criterios considerados
Placa embebida	Procesador, frecuencia de operación, memoria RAM, unidad gráfica, aceleración para inteligencia artificial, almacenamiento, interfaces para cámara y pantalla, puertos disponibles, conectividad, alimentación, compatibilidad con el entorno de software, documentación y disponibilidad.

Categoría	Criterios considerados
Dispositivo de captura visual	Resolución, tipo de sensor, conexión, captura de video, compatibilidad con la placa embebida, uso de puertos, tamaño, facilidad de integración y necesidad de información de profundidad.
Medio de interacción visual	Tamaño y resolución de pantalla, capacidad táctil, tipo de conexión, compatibilidad con la placa embebida, portabilidad, alimentación y facilidad de uso dentro del prototipo.
Medio de almacenamiento	Capacidad, compatibilidad con la placa, posibilidad de instalar el sistema operativo, funcionamiento local, facilidad de reemplazo, respaldo de la información y necesidad de puertos adicionales.
Medio de conectividad	Tipo de conexión, estabilidad, alcance, acceso al dashboard, funcionamiento sin internet, compatibilidad con la Jetson Nano y facilidad de configuración.
Fuente de alimentación	Voltaje, corriente suministrada, tipo de conexión, estabilidad, funcionamiento prolongado y compatibilidad con la placa y los periféricos.
Estructura física	Material, protección de los componentes, estabilidad, acceso a conexiones, organización del cableado, facilidad de fabricación, adaptación a la cámara y pantalla, y adecuación visual para niños.

Nota. Elaboración propia a partir de las características técnicas consideradas para la selección de los componentes del prototipo.

C.2 Comparación de las placas embebidas

Para seleccionar la placa encargada del procesamiento local se analizaron seis alternativas con capacidad para ejecutar aplicaciones en sistemas embebidos. La comparación consideró el procesador, memoria RAM, aceleración para inteligencia artificial, almacenamiento, interfaces, conectividad, alimentación y compatibilidad con las herramientas utilizadas en el proyecto.

Como se presenta en la Tabla C.2, las características evaluadas se colocaron en las filas, mientras que las placas se organizaron en las columnas. Esta distribución permite identificar con mayor claridad las diferencias entre las alternativas y determinar cuál se ajusta mejor a los requerimientos del sistema.

Tabla C.2
Matriz comparativa para la selección de la placa embebida

Característica	Raspberry Pi 4	Raspberry Pi 5	Coral Dev Board	NVIDIA Jetson Nano	Jetson Orin Nano	Orange Pi 5
Procesador	Broadcom BCM2711, cuatro núcleos Cortex-A72	Broadcom BCM2712, cuatro núcleos Cortex-A76	NXP i.MX 8M, cuatro núcleos Cortex-A53 y Cortex-M4F	NVIDIA Tegra X1, cuatro núcleos Cortex-A57	Seis núcleos Cortex-A78AE	Rockchip RK3588S, cuatro Cortex-A76 y cuatro Cortex-A55
Frecuencia del procesador	Hasta 1,8 GHz	2,4 GHz	Hasta 1,5 GHz	Hasta 1,43 GHz	Depende de la configuración y modo de energía	Hasta 2,4 GHz
Memoria RAM	1, 2, 4 u 8 GB LPDDR4	1, 2, 4, 8 o 16 GB LPDDR4X	1 GB LPDDR4	4 GB LPDDR4	4 u 8 GB LPDDR5, según versión	4, 8, 16 o 32 GB LPDDR4/LPDDR4X
Unidad gráfica	VideoCore VI	VideoCore VII	GC7000 Lite	GPU NVIDIA Maxwell de 128 núcleos CUDA	GPU NVIDIA Ampere	ARM Mali-G610 MP4
Aceleración para inteligencia artificial	No incorpora acelerador dedicado	No incorpora acelerador dedicado	Edge TPU de hasta 4 TOPS	GPU con núcleos CUDA	GPU con núcleos CUDA y Tensor Cores	NPU integrada de hasta 6 TOPS
Almacenamiento	Tarjeta microSD	Tarjeta microSD	8 GB eMMC y ranura microSD	eMMC o microSD, según el kit utilizado	microSD y almacenamiento NVMe, según el kit	microSD, SPI Flash y SSD NVMe
Interfaz para cámara	MIPI CSI	Conectores MIPI para cámara o pantalla	MIPI CSI-2	MIPI CSI	MIPI CSI	MIPI CSI
Salida para pantalla	Dos puertos micro-HDMI y MIPI DSI	Dos puertos micro-HDMI y conectores MIPI	HDMI y MIPI DSI	HDMI y DisplayPort, según la placa portadora	DisplayPort o HDMI, según el kit	HDMI
Conectividad cableada	Ethernet Gigabit	Ethernet Gigabit	Ethernet Gigabit	Ethernet Gigabit	Ethernet Gigabit	Ethernet Gigabit
Wi-Fi y Bluetooth	Integrados	Integrados	Integrados	Requiere módulo o adaptador externo	Depende del kit o módulo instalado	Requiere módulo o adaptador externo
Puertos para periféricos	USB y GPIO de 40 pines	USB y GPIO de 40 pines	USB y cabecera de expansión	USB y GPIO de 40 pines	USB, GPIO y conectores de expansión	USB y GPIO

Alimentación	USB-C de 5 V	USB-C de 5 V–5 A	USB-C de 5 V	5 V–4 A mediante conector de alimentación	Depende del kit y modo de energía	5 V–4 A, según fabricante
Sistema operativo principal	Raspberry Pi OS	Raspberry Pi OS	Mendel Linux	NVIDIA JetPack, basado en Ubuntu	NVIDIA JetPack	Orange Pi OS, Ubuntu, Debian o Android
Compatibilidad con Python y OpenCV	Alta	Alta	Alta	Alta	Alta	Alta, con posibles ajustes
Compatibilidad con TensorFlow y TensorFlow Lite	Compatible, sin aceleración especializada	Compatible, sin aceleración especializada	TensorFlow Lite con modelos compatibles con Edge TPU	Compatible con TensorFlow, Keras y TensorFlow Lite	Compatible con TensorFlow y herramientas NVIDIA	Compatible, pero puede requerir adaptación
Compatibilidad con MediaPipe	Posible, según versión y rendimiento	Posible	Requiere verificar compatibilidad del entorno	Compatible con el entorno utilizado	Compatible con el entorno NVIDIA	Puede requerir configuración adicional
Documentación y soporte	Amplios	Amplios	Documentación específica de Coral	Amplios para inteligencia artificial embebida	Amplios y actualizados	Menores frente a Raspberry Pi y NVIDIA
Ventaja principal	Disponibilidad y facilidad de desarrollo	Mayor rendimiento que Raspberry Pi 4	Inferencia eficiente mediante Edge TPU	Equilibrio entre procesamiento, compatibilidad y costo	Mayor capacidad para inteligencia artificial	Procesador y NPU de alto rendimiento
Limitación principal	Sin aceleración dedicada para IA	Sin aceleración dedicada para IA	Restricciones en los modelos compatibles con Edge TPU	Plataforma con recursos menores que generaciones recientes y sin Wi-Fi integrado	Mayor costo para el alcance del proyecto	Mayor complejidad de configuración y adaptación de librerías
Resultado de la comparación	No seleccionada	No seleccionada	No seleccionada	**Seleccionada**	No seleccionada	No seleccionada

Nota. Elaboración propia con base en las especificaciones técnicas de los fabricantes y proveedores consultados para la selección de la placa embebida.

Como se observa en la Tabla C.2, la Jetson Orin Nano y la Orange Pi 5 presentan una capacidad de procesamiento mayor que la NVIDIA Jetson Nano. Sin embargo, la selección no se basó únicamente en escoger la alternativa con las especificaciones más altas, sino en encontrar una placa que cumpliera los requerimientos del proyecto y fuera compatible con los componentes y herramientas disponibles.

La NVIDIA Jetson Nano fue seleccionada porque incorpora una GPU con núcleos CUDA, permite conectar directamente la cámara mediante MIPI CSI y dispone de salida HDMI para la pantalla táctil. También es compatible con Python, OpenCV, MediaPipe, TensorFlow/Keras y TensorFlow Lite, herramientas utilizadas para desarrollar y ejecutar el sistema de reconocimiento. La Jetson Orin Nano ofrecía un rendimiento superior, pero su costo y capacidad superaban las necesidades definidas para el prototipo.

C.3 Comparación de los dispositivos de captura visual

Para la selección del dispositivo de captura visual se analizaron cuatro alternativas capaces de obtener imágenes o video para el posterior reconocimiento de las señas. La comparación consideró la resolución, el tipo de sensor, la velocidad de captura, la interfaz de conexión, la compatibilidad con la NVIDIA Jetson Nano y la facilidad de integración dentro del prototipo.

Como se presenta en la Tabla C.3, las características evaluadas se organizaron en las filas y los dispositivos considerados en las columnas. Esta distribución permite identificar las diferencias entre las alternativas y establecer cuál se ajusta mejor a los requerimientos de captura continua, detección de la mano y extracción de puntos característicos.

Tabla C.3
Matriz comparativa de los dispositivos de captura visual

Característica	Cámara USB Logitech C270	Cámara CSI IMX219	Raspberry Pi Camera Module 3	Intel RealSense D435
Tipo de dispositivo	Cámara web RGB	Módulo de cámara RGB para sistemas embebidos	Módulo de cámara RGB para Raspberry Pi	Cámara RGB-D con medición de profundidad
Sensor	Sensor de imagen integrado	Sony IMX219	Sony IMX708	Sistema estereoscópico de profundidad y sensor RGB
Resolución principal	HD 1280 × 720 píxeles	8 MP, hasta 3280 × 2464 píxeles	Aproximadamente 12 MP	Profundidad de hasta 1280 × 720 píxeles y captura RGB
Captura de video	720p a 30 fps	1080p a 30 fps, 720p a 60 fps y 640 × 480 a 90 fps	Permite captura de video en diferentes resoluciones	Captura simultánea de imagen RGB y profundidad
Captura de profundidad	No	No	No	Sí
Enfoque	Fijo	Fijo, según el módulo utilizado	Enfoque automático	Enfoque y profundidad mediante sistema estereoscópico
Campo de visión	Aproximadamente 55° diagonal	Depende del lente del módulo	Disponible en versiones estándar y gran angular	Campo amplio para percepción de profundidad
Interfaz de conexión	USB	MIPI CSI	MIPI CSI	USB 3.0
Uso de puerto USB	Sí	No	No	Sí
Compatibilidad con NVIDIA Jetson Nano	Compatible como cámara USB estándar	Compatible mediante interfaz CSI	Requiere verificar controladores y adaptación	Compatible mediante Intel RealSense SDK
Alimentación	Mediante USB	Mediante la interfaz CSI	Mediante la interfaz CSI	Mediante USB
Tamaño para integración	Medio	Compacto	Compacto	Mayor que un módulo CSI convencional
Complejidad de configuración	Baja	Media	Media o alta en una plataforma diferente a Raspberry Pi	Alta debido al uso del SDK y datos de profundidad
Ventaja principal	Fácil conexión y disponibilidad	Conexión directa, tamaño reducido y resolución suficiente	Mayor resolución y enfoque automático	Captura información RGB y profundidad
Limitación principal	Menor resolución y uso de un puerto USB	No proporciona profundidad y depende de la compatibilidad CSI	Orientada principalmente al ecosistema Raspberry Pi	Mayor costo, tamaño y complejidad para una función no requerida
Resultado de la comparación	No seleccionada	**Seleccionada**	No seleccionada	No seleccionada

Nota. Elaboración propia con base en las especificaciones técnicas de los fabricantes y proveedores consultados para la selección de la placa embebida

Análisis de la comparación

Como se observa en la Tabla C.3, la Intel RealSense D435 ofrece funciones avanzadas de captura de profundidad, mientras que la Raspberry Pi Camera Module 3 dispone de una mayor resolución y enfoque automático. Sin embargo, estas características no eran indispensables para el reconocimiento desarrollado, debido a que el sistema utiliza los puntos de referencia extraídos de imágenes RGB mediante MediaPipe Hands.

La cámara CSI IMX219 fue seleccionada porque presenta una resolución de 8 MP, permite capturar video con una velocidad adecuada y se conecta directamente con la NVIDIA Jetson Nano mediante la interfaz MIPI CSI. Su tamaño compacto también facilita la instalación dentro de la estructura del prototipo y evita ocupar uno de los puertos USB disponibles.

C.4 Comparación de los medios de interacción visual

Para seleccionar el medio de interacción visual se analizaron cuatro alternativas que permitían mostrar las actividades del sistema y, en algunos casos, recibir las acciones del usuario. La comparación consideró la visualización de la interfaz, la función táctil, el tipo de conexión, la compatibilidad con la NVIDIA Jetson Nano, la portabilidad y la facilidad de integración dentro del prototipo.

Como se muestra en la **Tabla C.4**, los dispositivos se colocaron en las columnas y las características evaluadas en las filas. Esta organización permite observar las diferencias entre una pantalla utilizada únicamente para visualización y aquellas que también permiten controlar el sistema mediante acciones táctiles.

Tabla C.4
Matriz comparativa de los medios de interacción visual

Característica	Monitor HDMI	Pantalla HDMI táctil	Pantalla táctil DSI	Tablet
Tipo de dispositivo	Monitor externo convencional	Pantalla compacta con función táctil	Pantalla táctil para sistemas embebidos	Dispositivo móvil independiente
Visualización de la interfaz	Sí	Sí	Sí	Sí
Interacción táctil	No	Sí	Sí	Sí
Conexión de video	HDMI	HDMI	MIPI DSI	Red local mediante navegador
Conexión para la función táctil	No aplica	USB	Interfaz DSI o conexión adicional, según el modelo	Integrada en el dispositivo
Conexión directa con la Jetson Nano	Sí	Sí	Depende de la compatibilidad del modelo	No; se conecta mediante la red local
Necesidad de configuración adicional	Baja	Baja o media	Media o alta	Media
Dependencia de la red local	No	No	No	Sí
Portabilidad	Baja	Alta	Alta	Alta
Integración dentro del prototipo	Baja	Alta	Media	No se integra físicamente
Uso de puertos de la Jetson Nano	HDMI	HDMI y USB	Puerto DSI compatible	No ocupa puertos de video
Alimentación	Fuente externa	Fuente externa o USB, según el modelo	Mediante la placa o conexión adicional	Batería interna
Facilidad de uso para los niños	Media, requiere periféricos externos	Alta, permite interactuar directamente con la interfaz	Alta, cuando existe compatibilidad	Media, depende de un dispositivo adicional
Uso para el dashboard docente	Posible	Posible desde el prototipo	Posible desde el prototipo	Sí, mediante navegador
Ventaja principal	Mayor tamaño de visualización	Combina visualización y control táctil en un mismo dispositivo	Integración compacta mediante interfaz dedicada	Permite acceder al dashboard de forma inalámbrica
Limitación principal	No permite interacción directa y ocupa mayor espacio	Requiere conexión HDMI y USB para la función táctil	La compatibilidad con la Jetson Nano depende del modelo y los controladores	Depende de la red local y de un equipo externo
Resultado de la comparación	No seleccionado	**Seleccionada**	No seleccionada	No seleccionada

Nota. Elaboración propia con base en las especificaciones técnicas de los fabricantes y proveedores consultados para los medios de interacción visual

Como se observa en la Tabla C.4, el monitor HDMI permite visualizar correctamente la interfaz, pero no ofrece interacción táctil y requiere mayor espacio para su instalación. La pantalla DSI táctil presenta una integración compacta, aunque su funcionamiento depende de la compatibilidad entre el modelo de pantalla, la placa embebida y los controladores disponibles.

La pantalla HDMI táctil fue seleccionada porque permite mostrar las actividades y recibir las acciones del usuario directamente desde el prototipo. La conexión HDMI se utiliza para transmitir la imagen, mientras que la conexión USB permite controlar la función táctil. Esto evita utilizar teclado o mouse durante las actividades y facilita la interacción de los niños con el sistema.

C.5 Comparación de los medios de almacenamiento

Para seleccionar el medio de almacenamiento se analizaron alternativas capaces de alojar el sistema operativo, la aplicación, los modelos de reconocimiento, la base de datos SQLite, los recursos visuales y los registros generados durante las actividades.

Como se muestra en la Tabla C.5, la comparación consideró la capacidad, el tipo de conexión, la posibilidad de ejecutar el sistema operativo, el funcionamiento sin internet, la facilidad de respaldo y la integración con la NVIDIA Jetson Nano.

*Tabla **C.5***
Matriz comparativa de los medios de almacenamiento

Característica	Tarjeta microSD de 64 GB	Memoria eMMC	Memoria USB externa	SSD externo	Almacenamiento en la nube
Almacenamiento	Memoria flash removible	Memoria flash integrada	Memoria flash removible	Unidad de estado sólido externa	Almacenamiento remoto
Capacidad	64 GB	16 GB	Variable según el dispositivo	Variable según el modelo	Variable según proveedor
Interfaz de conexión	Ranura microSD	Integrada en la placa o módulo	Puerto USB	USB Type-C o USB 3.2	Conexión a internet
Instalación del sistema operativo	Sí	Sí, cuando está integrada en la placa	No se consideró como medio principal	Posible con configuración adicional	No
Funcionamiento sin internet	Sí	Sí	Sí	Sí	No
Almacenamiento de la aplicación y modelos	Sí	Sí, con limitación de capacidad	Sí	Sí	Sí, pero requiere conexión
Almacenamiento de la base de datos	Sí	Sí	No recomendable como medio principal	Sí	No directamente durante el funcionamiento local
Uso de puertos externos	No	No	Ocupa un puerto USB	Ocupa un puerto USB	No ocupa un puerto físico, pero requiere red
Velocidad	Depende de la clase de la tarjeta	Adecuada para almacenamiento integrado	Depende del estándar USB y del dispositivo	Mayor velocidad de lectura y escritura	Depende de la conexión a internet
Facilidad de instalación	Alta	Depende del modelo de placa	Alta	Media	Media
Facilidad de reemplazo	Alta	Baja	Alta	Media	No aplica
Posibilidad de clonación y respaldo	Alta	Media	Alta para archivos	Alta	Alta para respaldo remoto
Riesgo de desconexión accidental	Bajo cuando está protegida dentro del prototipo	Bajo	Alto	Medio	Depende de la conexión a internet
Portabilidad	Alta	Integrada en la placa	Alta	Media	Accesible desde diferentes dispositivos
Dependencia de servicios externos	No	No	No	No	Sí
Adecuación para funcionamiento local	Alta	Alta, pero con menor capacidad	Media	Alta	Baja
Ventaja principal	Capacidad suficiente y facilidad de clonación	Mayor integración y menor riesgo de extracción	Facilidad para transferir y respaldar archivos	Mayor capacidad y velocidad	Acceso remoto y respaldo externo
Limitación principal	Puede dañarse o extraerse si no se protege	Capacidad limitada y reemplazo más complejo	Ocupa un puerto USB y puede desconectarse	Mayor costo, tamaño y complejidad de integración	Requiere internet y depende de servicios externos
Resultado	**Seleccionada**	No seleccionada	No seleccionado	No seleccionado	No seleccionado

Nota. Elaboración propia con base en las especificaciones técnicas

Análisis de la comparación

Como se observa en la Tabla C.5, se seleccionó una tarjeta microSD de 64 GB por su capacidad para almacenar el sistema operativo, la aplicación, los modelos, la base de datos y los recursos del prototipo. Además, facilita la creación de respaldos y su reemplazo durante el mantenimiento. Las demás alternativas presentaban menor capacidad, requerían más espacio o dependían de una conexión a internet.

C.6 Comparación de los medios de conectividad

Para seleccionar el medio de conectividad se analizaron alternativas que permitieran establecer comunicación entre la NVIDIA Jetson Nano y el dispositivo utilizado por la docente para ingresar al dashboard. Se consideraron el cable Ethernet, el adaptador Wi-Fi USB TP-Link TL-WN823N y la conectividad Bluetooth mediante un módulo compatible con la placa.

Como se muestra en la Tabla C.6, la comparación tomó en cuenta el tipo de conexión, velocidad, estabilidad, movilidad, funcionamiento local, acceso al dashboard y facilidad de integración. Las alternativas se colocaron en las columnas y las características analizadas en las filas para identificar con mayor claridad sus diferencias.

Tabla C.6
Matriz comparativa de los medios de almacenamiento

Característica	Cable Ethernet Cat 6	Adaptador Wi-Fi USB TP-Link TL-WN823N	Bluetooth mediante módulo Intel 8265AC
Tipo de conexión	Cableada	Inalámbrica	Inalámbrica de corto alcance
Estándar de comunicación	IEEE 802.3, mediante Ethernet	IEEE 802.11b/g/n	Bluetooth 4.2
Interfaz con la Jetson Nano	Puerto Ethernet RJ45	Puerto USB 2.0	Módulo compatible, según la configuración de la placa
Velocidad de referencia	Hasta 1 Gbps, limitada por el puerto Ethernet Gigabit	Hasta 300 Mbps	Menor que Ethernet y Wi-Fi; orientada a periféricos y transmisión de datos de corto alcance
Banda de operación	No aplica	2,4 GHz	2,4 GHz
Alcance	Depende de la longitud del cable	Adecuado para una red local dentro del aula	Corto alcance

Estabilidad de la conexión	Alta	Media–alta, según la cobertura y las interferencias	Media
Movilidad del prototipo	Baja, debido al cableado	Alta	Alta
Necesidad de cableado permanente	Sí	No	No
Uso de un puerto externo	Puerto RJ45	Puerto USB	Depende del módulo instalado
Funcionamiento sin internet	Sí, dentro de una red local	Sí, mediante una red local o punto de acceso	Sí
Acceso al dashboard mediante navegador	Sí	Sí	No directamente
Consulta y descarga de reportes	Sí	Sí	No es adecuada para esta función
Creación de un punto de acceso local	No por sí solo	Sí, mediante configuración en modo punto de acceso	No para el acceso web planteado
Facilidad de configuración	Alta	Media, puede requerir controlador y configuración inicial	Media o alta, según el módulo y el servicio requerido
Alimentación	No requiere alimentación independiente	Mediante el puerto USB	Mediante la interfaz del módulo
Compatibilidad con la Jetson Nano	Directa	Compatible mediante USB y controlador	Compatible mediante módulo externo
Uso dentro del proyecto	Configuración, mantenimiento y conexión cableada estable	Creación de la red local y acceso inalámbrico al dashboard	Posible conexión de periféricos cercanos
Ventaja principal	Proporciona una conexión estable y confiable	Permite movilidad y acceso inalámbrico desde otro dispositivo	Permite comunicar periféricos sin utilizar cables
Limitación principal	Limita la ubicación del dispositivo y requiere un punto de red cercano	Puede verse afectado por interferencias y requiere configuración del controlador	No permite acceder directamente al dashboard web ni descargar reportes
Resultado de la comparación	No seleccionado	**Seleccionado**	No seleccionado

Nota. Elaboración propia con base en las especificaciones técnicas

Como se observa en la Tabla C.6, se seleccionó el adaptador Wi-Fi USB TP-Link TL-WN823N porque permite acceder al dashboard de forma inalámbrica dentro de la red local, sin depender de internet. La conexión Ethernet se mantiene como apoyo para tareas de configuración y mantenimiento, mientras que Bluetooth no resulta adecuado para acceder al sistema desde un navegador.

C.7 Comparación de las alternativas de alimentación

Para seleccionar la fuente de alimentación se analizaron tres alternativas capaces de suministrar energía al prototipo: alimentación mediante un puerto USB de computadora, batería externa y fuente AC/DC conectada al puerto de alimentación de la NVIDIA Jetson Nano.

Como se muestra en la Tabla C.7, la comparación consideró el voltaje, la corriente disponible, la potencia, la continuidad de funcionamiento, la estabilidad, la portabilidad y la capacidad de alimentar la placa junto con los periféricos utilizados. El cálculo detallado del consumo energético no se incluye en este anexo, debido a que se desarrolla posteriormente en el apartado Consideraciones y dimensionamiento energético de operación.

Tabla C.7
Matriz comparativa de los medios de almacenamiento

Característica	Puerto USB de computadora	Batería externa Anker PowerCore Select A1363	Fuente AC/DC Security-01 de 5 V–4 A
Tipo de alimentación	Alimentación desde un puerto USB de una computadora	Batería portátil recargable	Fuente externa conectada a la red eléctrica
Modelo de referencia	Cable USB 3.0 macho a macho	Anker PowerCore Select 20000 mAh A1363	Security-01 AC to DC Power Supply Adapter
Voltaje de salida	5 V, según el puerto USB	5 V, 9 V o 12 V, según el modo de salida	5 V DC
Corriente disponible	Limitada por el tipo de puerto USB; no garantiza los 4 A requeridos	Hasta 3 A en la salida de 5 V, según la ficha consultada	4 A
Potencia nominal máxima	Depende del estándar y capacidad del puerto	Hasta 18 W, según el modo de salida	20 W
Capacidad energética	No aplica; depende de que la computadora permanezca encendida	20 000 mAh o 74 Wh	No aplica; funciona mientras exista suministro eléctrico
Tipo de conexión	Puerto USB	Puertos USB-A	Conector DC tipo barril de 5,5 × 2,1 mm
Alimentación continua	Depende del funcionamiento de la computadora	Limitada por la carga disponible	Sí
Autonomía	Depende de la computadora	Temporal y condicionada por el nivel de carga	Continua mientras esté conectada a la red eléctrica
Estabilidad de salida	Puede ser insuficiente para la placa y los periféricos	Depende de la regulación, carga y modo de salida	Adecuada para una salida constante de 5 V–4 A

Compatibilidad con la Jetson Nano	No adecuada como fuente principal	Puede presentar limitaciones de corriente o activación de salida	Compatible con el puerto de alimentación correspondiente
Alimentación de periféricos	Limitada	Puede resultar insuficiente al conectar varios periféricos	Permite atender la carga prevista del prototipo
Portabilidad	Baja, porque depende de una computadora	Alta	Media, porque requiere un tomacorriente
Uso prolongado	No recomendado	Limitado por la autonomía	Adecuado
Dependencia de otro equipo	Sí, de una computadora	No	No, pero depende de la red eléctrica
Facilidad de integración	Baja para el funcionamiento final	Media	Alta
Ventaja principal	Disponibilidad para pruebas básicas	Permite utilizar el sistema temporalmente sin tomacorriente	Proporciona alimentación estable y continua
Limitación principal	No garantiza la corriente requerida y puede provocar funcionamiento inestable	La salida de 5 V alcanza hasta 3 A y su autonomía es limitada	Reduce la movilidad al requerir conexión eléctrica
Resultado de la comparación	No seleccionada	No seleccionada	**Seleccionada**

Nota. Elaboración propia con base en las especificaciones técnicas

C.8 Comparación de las alternativas de estructura física

Para seleccionar la estructura física del prototipo se analizaron tres alternativas capaces de sostener e integrar la NVIDIA Jetson Nano, la cámara CSI, la pantalla HDMI táctil, la fuente de alimentación y el cableado. Las opciones consideradas fueron el montaje directo sobre una mesa, una caja cerrada de acrílico o plástico y una base de madera construida para el dispositivo.

Como se muestra en la Tabla C.8, la comparación consideró la protección de los componentes, ventilación, estabilidad, acceso a las conexiones, organización del cableado, facilidad de adaptación y adecuación al entorno infantil.

Tabla C.8
Matriz comparativa de los medios de almacenamiento

Característica	Montaje directo sobre mesa	Caja cerrada de acrílico o plástico	Base de madera

Tipo de estructura	Montaje abierto sin una base fija	Carcasa cerrada para proteger principalmente la placa embebida	Base física diseñada para sostener e integrar el prototipo
Material principal	No aplica; los componentes se colocan directamente sobre una superficie	Láminas de acrílico o plástico	Madera con acabado de color blanco
Protección de los componentes	Baja, debido a que los componentes permanecen expuestos	Media–alta	Media, según la disposición de los componentes
Ventilación	Alta por la exposición directa de los componentes	Parcial, depende de las aberturas de la carcasa	Adecuada por la disposición abierta
Estabilidad	Baja o media, porque los componentes no están fijados	Alta para la placa instalada dentro de la carcasa	Alta, al sujetar los elementos principales sobre una misma base
Acceso a conexiones	Alto	Medio	Alto
Acceso para mantenimiento	Alto, aunque sin organización definida	Medio, debido a que puede ser necesario desmontar la carcasa	Alto, porque los componentes permanecen accesibles
Organización del cableado	Limitada	Media	Adecuada
Integración de la Jetson Nano	Posible, pero sin protección ni fijación estable	Alta para la placa embebida	Alta
Integración de la pantalla táctil	Posible, pero sin posición fija	Limitada, porque la carcasa está orientada principalmente a la placa	Alta, permite ubicarla en una posición visible
Integración de la cámara CSI	Posible, pero sin soporte fijo	Requiere un soporte adicional	Alta, permite fijarla en el área de captura
Integración de la fuente y periféricos	Desordenada o temporal	Puede requerir elementos externos adicionales	Permite organizar los componentes y conexiones
Facilidad de construcción	Alta	Media, requiere piezas con dimensiones específicas	Alta mediante fabricación local
Facilidad de modificación	Alta	Media o baja	Alta
Portabilidad	Baja, porque los componentes se transportan por separado	Media	Alta, al mantener los elementos principales en una sola estructura
Adecuación visual para niños	Baja	Baja o media	Alta, por su forma de nube y acabado visual
Uso recomendado	Pruebas iniciales, conexión y depuración	Protección individual de la placa embebida	Integración final del prototipo

Ventaja principal	Permite acceder fácilmente a los componentes durante las pruebas	Mejora la protección y presentación de la placa embebida	Ofrece estabilidad, organización, facilidad de adaptación y una presentación apropiada para el entorno infantil
Limitación principal	Deja expuestos los componentes y dificulta su organización	Puede limitar la ventilación, el mantenimiento y la integración de la pantalla, cámara y cableado	Proporciona menor protección frente al polvo o golpes que una carcasa totalmente cerrada
Resultado de la comparación	No seleccionado	No seleccionada	**Seleccionada**

Nota. Elaboración propia con base en las especificaciones técnicas

Como se observa en la Tabla C.8, se seleccionó la base de madera porque permite integrar de manera estable la pantalla, la cámara, la Jetson Nano y el cableado. Además, facilita la organización de los componentes y ofrece una presentación adecuada para niños, aunque brinda menos protección que una carcasa cerrada.

Anexo D: Matrices comparativas para la selección del software

Objetivo

- Presentar las características y los criterios técnicos considerados para comparar y seleccionar las herramientas de software utilizadas en el desarrollo, entrenamiento y funcionamiento del sistema.

Para el desarrollo del prototipo se compararon distintas herramientas de software según su compatibilidad con la NVIDIA Jetson Nano, facilidad de integración, ejecución local y recursos necesarios. En este anexo se presentan las matrices comparativas y la configuración final utilizada en la computadora de entrenamiento y en la placa.

D.1 Criterios generales considerados para la selección del software

Antes de comparar las diferentes herramientas, se establecieron criterios de acuerdo con la función que debía cumplir cada una dentro del sistema. Se tomaron en cuenta aspectos como la compatibilidad con la NVIDIA Jetson Nano, la integración con Python, el soporte para visión artificial, la ejecución de modelos y la posibilidad de mantener el funcionamiento dentro de una red local.

Como se muestra en la **Tabla D.1**, los criterios varían según la categoría del software analizado. Esta tabla resume los aspectos utilizados posteriormente para comparar las alternativas y no representa todavía la selección final.

Tabla D.1
Matriz comparativa de los medios de almacenamiento

Categoría	Criterios considerados
Sistema operativo	Compatibilidad con la NVIDIA Jetson Nano, disponibilidad de controladores, soporte para cámara y pantalla, compatibilidad con CUDA, ejecución local, conexión de periféricos, documentación, estabilidad y licencia.
Lenguaje de programación	Compatibilidad con la plataforma embebida, integración con OpenCV, MediaPipe, TensorFlow y SQLite, facilidad de desarrollo, rendimiento, disponibilidad de librerías, documentación y mantenimiento del código.
Procesamiento de imágenes	Captura de imágenes y video, procesamiento continuo de fotogramas, conversión de color, redimensionamiento, integración con la cámara, compatibilidad con Python y Jetson Nano, consumo de recursos y licencia.

Detección de manos y extracción de características	Detección de la mano, cantidad y tipo de puntos característicos, disponibilidad de modelos preentrenados, integración con Python y OpenCV, requerimientos de procesamiento y adecuación para ejecución local.
Framework de aprendizaje automático	Construcción de redes neuronales, compatibilidad con modelos densos y LSTM, entrenamiento, evaluación mediante métricas, integración con Python, exportación de modelos, soporte para sistemas embebidos, documentación y licencia.
Motor de inferencia	Formato de los modelos, compatibilidad con la Jetson Nano, ejecución local, consumo de memoria, tiempo de carga, soporte de las capas utilizadas y facilidad de integración con la aplicación principal.
Servidor local	Integración con Python, consumo de recursos, funcionamiento dentro de una red local, gestión de rutas, acceso desde navegadores, facilidad de configuración y mantenimiento.
Base de datos	Almacenamiento local, integración con Python, organización de registros, necesidad de un servidor externo, facilidad de consulta, respaldo, mantenimiento y exportación de resultados.
Interfaz web	Compatibilidad con navegadores, funcionamiento local, adaptación a teléfonos y computadoras, facilidad de uso, integración con el servidor y presentación de los resultados.

Nota. Elaboración propia con base en las especificaciones técnicas

D.2 Comparación de los sistemas operativos

Para seleccionar el sistema operativo del prototipo se analizaron cuatro alternativas: NVIDIA JetPack con Jetson Linux, Ubuntu de propósito general, Raspberry Pi OS y Windows.

La comparación tomó en cuenta la compatibilidad con la NVIDIA Jetson Nano, la disponibilidad de controladores, el acceso a la GPU, el soporte para la cámara CSI, la integración con las herramientas de visión artificial y la posibilidad de ejecutar el sistema localmente.

Como se muestra en la Tabla D.2, los sistemas operativos se ubicaron en las columnas y los criterios de comparación en las filas. Esta organización permite observar las diferencias entre un entorno preparado específicamente para la plataforma NVIDIA Jetson y los sistemas diseñados para computadoras o placas de otras familias.

Tabla D.2
Matriz comparativa de los sistemas operativos considerados

Característica	NVIDIA JetPack SDK con Jetson Linux	Ubuntu de propósito general	Raspberry Pi OS	Windows
Tipo de entorno	Conjunto de software y sistema basado en Linux, preparado para plataformas NVIDIA Jetson	Distribución Linux de propósito general	Sistema operativo oficial para computadoras Raspberry Pi	Sistema operativo de propósito general para computadoras
Plataforma principal	NVIDIA Jetson	Computadoras y plataformas ARM o x86 compatibles	Raspberry Pi	Computadoras compatibles
Compatibilidad con la Jetson Nano utilizada	Alta, mediante una versión de JetPack compatible con el modelo	Limitada si se instala sin el conjunto de controladores de NVIDIA	No está diseñado para Jetson Nano	No se consideró para instalarse en la Jetson Nano
Controladores para el hardware NVIDIA	Incluidos para la plataforma compatible	Requieren instalación y configuración específica	No incluye controladores para la Jetson Nano	No incluye el entorno requerido por la Jetson Nano
Acceso a la GPU NVIDIA	Sí	Depende de los controladores y la versión instalada	No para la GPU de Jetson	No aplicable al prototipo
Soporte para CUDA y TensorRT	Incluido dentro del entorno compatible	Requiere configuración adicional y compatibilidad con la plataforma	No dispone del conjunto de herramientas para Jetson	Puede utilizar herramientas de IA en una computadora compatible, pero no corresponde al entorno del prototipo
Compatibilidad con cámara MIPI CSI	Alta mediante los controladores de Jetson Linux	Depende de la configuración y los controladores disponibles	Orientada a cámaras compatibles con Raspberry Pi	No proporciona soporte directo para la cámara CSI utilizada en la Jetson Nano
Compatibilidad con pantalla HDMI y USB táctil	Sí	Sí, con la configuración adecuada	Sí en hardware Raspberry Pi	Sí en computadoras compatibles
Compatibilidad con Python	Sí	Sí	Sí	Sí
Compatibilidad con OpenCV	Sí	Sí	Sí	Sí
Compatibilidad con MediaPipe	Posible según la versión instalada y la arquitectura	Posible según la arquitectura y las dependencias	Puede presentar limitaciones según el modelo de placa	Sí en la computadora utilizada para el desarrollo

Compatibilidad con TensorFlow y TensorFlow Lite	Sí, utilizando versiones compatibles con Jetson	Sí, según la arquitectura y versión	Sí, con posibles limitaciones de procesamiento	Sí, utilizado para preparar y entrenar los modelos
Ejecución sin internet	Sí	Sí	Sí	Sí
Uso de los recursos de la Jetson Nano	Alto, porque integra controladores y herramientas específicas	Parcial si no se configura el entorno NVIDIA	No adecuado	No adecuado
Facilidad de instalación en el prototipo	Alta mediante imagen compatible o herramientas de NVIDIA	Media o baja, debido a la configuración requerida	No aplicable	No aplicable
Documentación para la plataforma	Amplia y específica para NVIDIA Jetson	Amplia, pero general	Amplia para Raspberry Pi	Amplia para computadoras
Ventaja principal	Integra el sistema, los controladores y las herramientas requeridas para utilizar la GPU y los periféricos de la Jetson Nano	Ofrece un entorno Linux flexible y de propósito general	Presenta integración directa con las placas Raspberry Pi	Facilita la preparación del dataset, desarrollo y entrenamiento en la computadora
Limitación principal	La versión debe ser compatible específicamente con el modelo de Jetson utilizado	Una instalación genérica no garantiza el funcionamiento de todos los componentes de la Jetson Nano	No está preparado para el hardware NVIDIA Jetson	No permite desplegar directamente el sistema en la Jetson Nano
Uso dentro del proyecto	Ejecución de la aplicación y los modelos en el prototipo	No utilizado como instalación independiente	No utilizado	Preparación de datos, programación y entrenamiento en la computadora
Resultado de la comparación	**Seleccionado para el despliegue**	No seleccionado	No seleccionado	**Utilizado en el entorno de entrenamiento**

Nota. Elaboración propia con base en la documentación

Como se observa en la Tabla D.2, se seleccionó NVIDIA JetPack con Jetson Linux por su compatibilidad con la GPU, la cámara y los controladores de la Jetson Nano. Windows se utilizó en la computadora de entrenamiento y desarrollo. Para la Jetson Nano original corresponde emplear una versión de JetPack 4.6.3, ya que JetPack 6 está destinado a plataformas más recientes.

D.3 Comparación de los lenguajes de programación

Para implementar los módulos del sistema se analizaron Python 3, C++ y JavaScript/Node.js. La comparación consideró su compatibilidad con la NVIDIA Jetson Nano, integración con las herramientas de visión artificial y aprendizaje automático, conexión con la base de datos, desarrollo del servidor local y facilidad de mantenimiento.

Como se muestra en la Tabla D.3, los lenguajes se organizaron en las columnas y las características evaluadas en las filas. Esta distribución permite distinguir el lenguaje utilizado para desarrollar la lógica principal del sistema de aquellos que podían utilizarse para tareas de alto rendimiento o para el desarrollo de la interfaz web.

Tabla D.3
Matriz comparativa de los lenguajes de programación considerados

Característica	Python 3	C++	JavaScript / Node.js
Tipo de lenguaje	Lenguaje de alto nivel y propósito general	Lenguaje compilado de propósito general	Lenguaje orientado principalmente al desarrollo web; Node.js permite ejecutarlo en el servidor
Tipo de ejecución	Interpretado	Compilado	Interpretado o ejecutado mediante un motor JavaScript
Plataformas compatibles	Linux, Windows y macOS	Linux, Windows, macOS y sistemas embebidos	Navegadores web, Linux, Windows y macOS
Compatibilidad con NVIDIA Jetson Nano	Alta mediante el entorno Linux de la placa	Alta mediante compiladores compatibles	Compatible mediante navegador o instalación de Node.js
Integración con OpenCV	Alta y directa mediante la biblioteca para Python	Alta mediante la biblioteca nativa	Posible, pero menos directa para el alcance del proyecto
Integración con MediaPipe Hands	Alta mediante su interfaz para Python	Posible mediante la implementación correspondiente	Posible mediante soluciones para entornos web, pero diferente a la utilizada en el prototipo
Integración con TensorFlow/Keras	Alta	Posible mediante sus interfaces y librerías	Posible mediante TensorFlow.js, pero no corresponde al entorno de entrenamiento utilizado

Integración con TensorFlow Lite	Alta mediante el intérprete para Python	Alta mediante las bibliotecas correspondientes	Limitada para la implementación planteada
Integración con NumPy y scikit-learn	Directa	Requiere alternativas o integraciones adicionales	No directa
Integración con SQLite	Directa mediante el módulo sqlite3	Posible mediante bibliotecas o conectores	Posible mediante paquetes adicionales
Desarrollo de servidor local	Directo mediante Flask	Posible, pero requiere mayor configuración	Alto mediante Node.js y frameworks web
Desarrollo de interfaz web	Permite generar las rutas y datos mediante Flask	No es su uso principal	Alta para controlar acciones y actualizar elementos del dashboard
Captura y procesamiento continuo de video	Sí, mediante OpenCV	Sí, con mayor control de bajo nivel	Posible, pero menos conveniente para la implementación local desarrollada
Ejecución de los modelos de reconocimiento	Directa mediante TensorFlow, Keras y TensorFlow Lite	Posible mediante bibliotecas compatibles	Posible con herramientas diferentes a las utilizadas
Rendimiento de ejecución	Adecuado para el alcance del prototipo	Alto	Adecuado para interfaces y servicios web
Facilidad de desarrollo	Alta	Media o baja	Alta para aplicaciones web
Complejidad de depuración	Baja o media	Alta	Media
Disponibilidad de librerías	Amplia para visión artificial, inteligencia artificial, bases de datos y desarrollo web	Amplia para procesamiento de alto rendimiento	Amplia para desarrollo web y servicios de red
Facilidad de mantenimiento	Alta por la legibilidad y estructura del código	Media, por la mayor complejidad del lenguaje	Alta en los módulos web
Licencia o disponibilidad	Código abierto y uso libre	Estandarizado y de uso libre	De uso libre mediante navegadores y entornos compatibles
Función posible dentro del proyecto	Captura, procesamiento, reconocimiento, base de datos y servidor local	Procesamiento de alto rendimiento o integración de bajo nivel	Interfaz web y posibles servicios del dashboard
Ventaja principal	Integra directamente las herramientas utilizadas en todos los módulos principales	Ofrece mayor rendimiento y control sobre los recursos del sistema	Facilita el desarrollo de interfaces web interactivas
Limitación principal	Puede presentar menor rendimiento que un lenguaje compilado en tareas intensivas	Requiere mayor tiempo de desarrollo, compilación y depuración	Presenta menor integración directa con las herramientas de visión artificial utilizadas
Uso dentro del proyecto	Lógica principal, procesamiento visual, inferencia, almacenamiento y servidor Flask	No utilizado como lenguaje principal	Utilizado de manera complementaria en el dashboard

Resultado de la comparación	Seleccionado como lenguaje principal	No seleccionado	Complementario para la interfaz web

Nota. Elaboración propia con base en la documentación

Como se observa en la Tabla D.3, Python 3 fue seleccionado como lenguaje principal por su facilidad de integración con las librerías de visión artificial, aprendizaje automático, almacenamiento y desarrollo web utilizadas en el prototipo. C++ requería una implementación más compleja, mientras que JavaScript se empleó únicamente como apoyo en la interfaz del dashboard.

D.4 Comparación de las herramientas de procesamiento de imágenes

Para seleccionar la herramienta de procesamiento de imágenes se analizaron PIL/Pillow, MATLAB con Image Processing Toolbox y OpenCV. Estas alternativas coinciden con las presentadas en las fichas técnicas del anexo actual.

La comparación consideró la capacidad para capturar video, procesar continuamente los fotogramas, realizar operaciones sobre las imágenes e integrarse con Python, la cámara IMX219, MediaPipe Hands y la NVIDIA Jetson Nano.

Como se muestra en la Tabla D.4, las herramientas se organizaron en las columnas y las características evaluadas en las filas. Esta distribución permite identificar cuál alternativa responde mejor a las necesidades de captura y preparación de las imágenes utilizadas por el sistema.

Tabla D.4
Matriz comparativa de los lenguajes de programación considerados

Característica	PIL/Pillow	MATLAB con Image Processing Toolbox	OpenCV
Tipo de herramienta	Biblioteca de procesamiento de imágenes	Entorno de cálculo técnico con herramientas para procesamiento de imágenes	Biblioteca de visión artificial y procesamiento de imágenes
Desarrollador o comunidad	Python Imaging Library Contributors	MathWorks	Comunidad OpenCV
Lenguaje principal	Python	Lenguaje MATLAB	Python, C++, Java y otros

Plataformas compatibles	Linux, Windows y macOS	Linux, Windows y macOS	Linux, Windows, macOS, Android y otras plataformas
Compatibilidad con Python	Directa	Requiere utilizar el entorno o mecanismos de integración de MATLAB	Directa
Compatibilidad con NVIDIA Jetson Nano	Sí, mediante un entorno Python compatible	Limitada para el despliegue planteado	Alta mediante instalación en Linux
Lectura de imágenes	Sí	Sí	Sí
Guardado de imágenes	Sí	Sí	Sí
Captura directa desde cámara	Limitada; no es su función principal	Posible mediante herramientas adicionales	Sí
Captura continua de video	Limitada	Posible, según la configuración	Sí
Procesamiento de fotogramas	Básico y orientado principalmente a imágenes individuales	Avanzado dentro del entorno MATLAB	Adecuado para trabajar cuadro a cuadro
Conversión de espacios de color	Permite conversiones básicas	Sí	Sí, mediante funciones específicas
Redimensionamiento y recorte	Sí	Sí	Sí
Filtrado y transformación	Operaciones básicas	Amplias herramientas de análisis y filtrado	Amplias funciones para procesamiento visual
Integración con la cámara IMX219	No directa	Requiere configuración y herramientas adicionales	Sí, mediante el flujo de captura configurado en la Jetson Nano
Integración con MediaPipe Hands	Posible para imágenes individuales, pero no directa para el flujo desarrollado	Menos directa debido al entorno utilizado	Alta, debido al intercambio de fotogramas con Python
Integración con NumPy	Sí	Requiere conversión entre entornos cuando se usa desde Python	Directa
Adecuación para visión artificial	Básica	Alta dentro del entorno MATLAB	Alta
Procesamiento local	Sí	Sí, en equipos compatibles con MATLAB	Sí
Consumo de recursos	Bajo para operaciones básicas	Depende del entorno y las herramientas instaladas	Adecuado para el procesamiento implementado
Licencia	Código abierto	Comercial	Código abierto
Función posible en el proyecto	Manipulación de imágenes individuales y recursos gráficos	Análisis, simulación o validación de imágenes	Captura, preparación y visualización de los fotogramas
Ventaja principal	Facilidad de uso para operaciones básicas con imágenes	Dispone de herramientas avanzadas de análisis y procesamiento	Integra captura de video, procesamiento visual y Python en una misma biblioteca
Limitación principal	No está orientada a la captura continua ni al procesamiento de video	Requiere licencia y presenta menor integración con el entorno basado en Python y Jetson Nano	Puede requerir configuración de versiones y dependencias en la Jetson Nano

Uso dentro del proyecto	No utilizada como herramienta principal	No utilizada	Captura y procesamiento de los fotogramas
Resultado de la comparación	No seleccionada como herramienta principal	No seleccionada	**Seleccionada**

Nota. Elaboración propia con base en la documentación

D.5 Comparación de las herramientas de detección de manos y extracción de características

Para seleccionar la herramienta encargada de detectar la mano y obtener sus características se analizaron OpenPose, YOLO Pose/Keypoints y MediaPipe Hands. Estas alternativas corresponden a las consideradas en las fichas técnicas actuales del anexo.

La comparación consideró el tipo de detección, los puntos característicos proporcionados, la necesidad de entrenamiento adicional, el consumo de recursos y la integración con Python, OpenCV y la NVIDIA Jetson Nano.

Como se presenta en la Tabla D.5, las herramientas se ubicaron en las columnas y las características evaluadas en las filas. Esta organización permite identificar cuál alternativa proporciona de manera más directa la información de la mano requerida por los modelos de reconocimiento.

Tabla D.5
Matriz comparativa de los lenguajes de programación considerados

Característica	OpenPose	YOLO Pose / Keypoints	MediaPipe Hands
Tipo de herramienta	Herramienta de estimación de pose y detección de puntos clave	Modelo de aprendizaje profundo para estimación de puntos clave	Solución especializada en detección, seguimiento y extracción de landmarks de la mano
Orientación principal	Detección de puntos del cuerpo, rostro, pies y manos	Detección de objetos y estimación de puntos definidos por el modelo	Detección y seguimiento específico de manos
Detección de manos	Sí	Posible mediante un modelo preparado o adaptado para esta tarea	Sí
Extracción de características	Puntos clave del cuerpo y de las manos	Keypoints definidos durante la configuración o entrenamiento del modelo	Landmarks que representan la palma y los dedos
Cantidad de puntos de la mano	Hasta 21 puntos por mano cuando se activa el modelo correspondiente	Variable; depende del modelo y del conjunto de puntos definido	21 landmarks por mano

Coordenadas proporcionadas	Coordenadas bidimensionales y valores de confianza; otras salidas dependen de la configuración	Coordenadas de los keypoints y nivel de confianza	Coordenadas normalizadas X, Y y una estimación relativa de Z
Detección de una o varias manos	Posible, según la configuración y los recursos disponibles	Depende del modelo entrenado	Permite configurar la cantidad máxima de manos
Seguimiento en video	Sí, con suficiente capacidad de procesamiento	Sí, según el modelo y la velocidad de ejecución	Sí, orientado al seguimiento continuo
Modelo previamente preparado para manos	Sí, dentro de la configuración de OpenPose	No necesariamente; puede requerir entrenamiento o adaptación	Sí
Necesidad de entrenamiento adicional	No para la detección general proporcionada por la herramienta	Puede ser necesario para obtener puntos específicos de la mano	No para detectar y extraer los landmarks
Compatibilidad con Python	Posible mediante instalación y configuración de las interfaces correspondientes	Sí	Sí
Integración con OpenCV	Posible	Sí	Directa mediante el intercambio de fotogramas
Compatibilidad con NVIDIA Jetson Nano	Limitada por la instalación y la demanda de procesamiento	Posible, dependiendo del tamaño, dependencias y optimización del modelo	Posible mediante un entorno Python compatible
Requerimiento computacional	Alto	Medio o alto, según el modelo	Moderado
Complejidad de instalación	Alta	Media o alta	Baja o media
Ejecución local	Sí	Sí	Sí
Adecuación para procesamiento continuo	Posible, pero puede requerir mayor capacidad de hardware	Depende del modelo y su optimización	Alta para el alcance del prototipo
Adecuación para gestos estáticos	Sí	Sí, con un modelo adecuado	Sí
Adecuación para gestos dinámicos	Permite obtener puntos en cada fotograma	Permite generar keypoints por fotograma	Permite formar secuencias de landmarks a partir de varios fotogramas
Formato de información aprovechado por los modelos	Requiere seleccionar y preparar los puntos de la mano	Requiere adaptar los keypoints generados	Proporciona 21 puntos que pueden transformarse en 63 valores X, Y y Z
Ventaja principal	Detecta puntos del cuerpo y de las manos dentro de una misma herramienta	Puede personalizarse para diferentes objetos y conjuntos de keypoints	Entrega directamente landmarks estructurados de la mano y se integra con Python y OpenCV
Limitación principal	Presenta mayor consumo computacional y complejidad de instalación	Puede requerir recolección de datos, etiquetado y entrenamiento adicional para puntos específicos de la mano	Puede verse afectada por oclusiones, iluminación, distancia y posiciones complejas
Uso dentro del proyecto	No utilizado	No utilizado	Detección de la mano y extracción de los datos

			empleados por los modelos
Resultado de la comparación	No seleccionada	No seleccionada	**Seleccionada**

Nota. Elaboración propia con base en la documentación

D.6 Comparación de los frameworks de aprendizaje automático

Para seleccionar la herramienta utilizada en la construcción, entrenamiento y evaluación de los modelos de reconocimiento se analizaron TensorFlow/Keras, PyTorch, Scikit-learn y MATLAB Deep Learning Toolbox. Estas son las mismas alternativas incluidas en el anexo actual.

La comparación consideró la capacidad para crear redes neuronales, trabajar con los landmarks extraídos mediante MediaPipe Hands, implementar modelos para señas estáticas y dinámicas, calcular métricas y exportar los modelos para su posterior ejecución en la NVIDIA Jetson Nano.

Como se presenta en la Tabla D.6, las herramientas se ubicaron en las columnas y las características evaluadas en las filas. Esta organización permite diferenciar los frameworks orientados al aprendizaje profundo de las herramientas empleadas principalmente para el procesamiento de datos y la evaluación de resultados.

Tabla D.6
Matriz comparativa de los lenguajes de programación considerados

Característica	TensorFlow/Keras	PyTorch	Scikit-learn	MATLAB Deep Learning Toolbox
Tipo de herramienta	Framework de aprendizaje automático y aprendizaje profundo	Framework de aprendizaje profundo	Biblioteca de aprendizaje automático	Herramienta de aprendizaje profundo integrada en MATLAB
Lenguaje principal utilizado	Python	Python	Python	MATLAB
Compatibilidad con Windows	Sí	Sí	Sí	Sí
Compatibilidad con Linux	Sí	Sí	Sí	Sí, mediante una instalación compatible
Integración con Python	Directa	Directa	Directa	Requiere trabajar en MATLAB o utilizar mecanismos de integración
Compatibilidad con NVIDIA Jetson Nano	Posible mediante versiones compatibles con la plataforma	Posible, según versión y dependencias	Posible para funciones de procesamiento y evaluación	Limitada para el entorno de despliegue planteado
Trabajo con datos numéricos de landmarks	Sí	Sí	Sí	Sí
Construcción de redes neuronales densas	Sí	Sí	Limitada; dispone de modelos básicos, pero no es su función principal	Sí
Construcción de modelos secuenciales	Sí, mediante Keras	Sí	Limitada	Sí
Soporte para redes LSTM	Sí	Sí	No como herramienta principal	Sí
Adecuación para señas estáticas	Alta	Alta	Media, dependiendo del algoritmo	Alta
Adecuación para señas dinámicas	Alta mediante redes recurrentes o LSTM	Alta mediante redes recurrentes o LSTM	Baja	Alta
Entrenamiento con GPU	Sí, en entornos compatibles	Sí	Limitado y dependiente del algoritmo	Sí, con hardware y herramientas compatibles
Definición de capas y arquitectura	Alta mediante la API de Keras	Alta, con mayor control sobre la arquitectura	Limitada para redes profundas	Alta mediante herramientas gráficas y código

Facilidad de construcción de modelos	Alta	Media	Alta para modelos clásicos	Alta dentro del entorno MATLAB
Cálculo de exactitud	Sí	Sí	Sí	Sí
Cálculo de precision, recall y F1-score	Posible mediante métricas propias o herramientas complementarias	Posible mediante funciones y bibliotecas complementarias	Sí, de forma directa	Sí
Generación de matriz de confusión	Posible	Posible	Sí, de forma directa	Sí
División del dataset	Posible, aunque puede apoyarse en Scikit-learn	Posible	Sí mediante train_test_split	Sí
Codificación de etiquetas	Posible	Posible	Sí mediante LabelEncoder y otras herramientas	Sí
Conversión para sistemas embebidos	Sí, mediante TensorFlow Lite	Requiere procesos o herramientas adicionales	No aplica directamente	Requiere herramientas adicionales
Formato de guardado de modelos	.h5, .keras, SavedModel y conversión a .tflite	.pt o .pth	.pkl, .joblib u otros	Formatos propios de MATLAB
Ejecución local de modelos	Sí	Sí	Sí para modelos compatibles	Sí, dentro de un entorno adecuado
Documentación disponible	Amplia	Amplia	Amplia	Amplia
Licencia	Código abierto	Código abierto	Código abierto	Comercial
Función dentro del proyecto	Construcción, entrenamiento y ejecución de los modelos	Alternativa para crear redes neuronales	División de datos, codificación de etiquetas, métricas y matriz de confusión	Alternativa para diseñar y entrenar modelos
Ventaja principal	Integra la creación de modelos mediante Keras y permite convertirlos a TensorFlow Lite	Proporciona flexibilidad y control durante el diseño de las redes	Facilita la preparación de los datos y el cálculo de métricas	Ofrece herramientas completas para el diseño y análisis de modelos
Limitación principal	En la Jetson Nano requiere versiones y dependencias compatibles	El despliegue embebido requiere pasos adicionales respecto a TensorFlow Lite	No está orientada principalmente a redes profundas ni modelos LSTM	Requiere licencia y presenta menor integración con el entorno final en Python
Uso dentro del proyecto	Entrenamiento de abecedario, números y palabras; ejecución del modelo dinámico y conversión de modelos estáticos	No utilizado	Herramienta complementaria durante la preparación y evaluación	No utilizado
Resultado de la comparación	**Seleccionado como framework principal**	No seleccionado	**Seleccionado como herramienta complementaria**	No seleccionado

Nota. Elaboración propia con base en la documentación

D.7 Herramientas para la inferencia y el despliegue de los modelos

Una vez finalizado el entrenamiento, los modelos se prepararon para ejecutarse localmente en la NVIDIA Jetson Nano. La forma de despliegue se definió según el tipo de seña y la estructura de cada modelo. Los modelos de abecedario y números, destinados a reconocer señas estáticas, se convirtieron al formato TensorFlow Lite. En cambio, el modelo de palabras se mantuvo en formato Keras .h5, debido a que procesa secuencias de landmarks mediante una arquitectura LSTM.

También se utilizaron archivos de codificación de etiquetas para relacionar la salida numérica de los modelos con la letra, número o palabra correspondiente. Como se presenta en la Tabla D.7, estas herramientas no se consideraron alternativas excluyentes, sino elementos que cumplen funciones diferentes dentro de la ejecución de los modelos. TensorFlow/Keras forma parte del entorno seleccionado para el desarrollo de los modelos.

Tabla D.7
Matriz de herramientas y formatos utilizados para la inferencia de los modelos

Característica	TensorFlow Lite	TensorFlow/Keras	LabelEncoder y Pickle
Tipo de herramienta	Motor de inferencia optimizado	Framework de aprendizaje profundo	Herramientas para codificación y almacenamiento de etiquetas
Formato utilizado	.tflite	.h5	.pickle o .pkl
Modelos relacionados	Abecedario y números	Palabras dinámicas	Abecedario, números y palabras
Tipo de seña procesada	Estática	Dinámica	No realiza reconocimiento
Información de entrada	Vector de landmarks correspondiente a una muestra	Secuencia de landmarks obtenida durante varios fotogramas	Índice numérico generado por el modelo
Función principal	Ejecutar los modelos estáticos con una carga reducida	Ejecutar el modelo LSTM de palabras	Convertir el índice predicho en el nombre de la clase
Ejecución local	Sí	Sí	Sí

Compatibilidad con Python	Sí, mediante el intérprete de TensorFlow Lite	Sí	Sí
Compatibilidad con la Jetson Nano	Sí, mediante una versión compatible	Sí, mediante versiones compatibles de TensorFlow y Keras	Sí
Consumo de recursos	Menor en comparación con la carga del modelo completo	Mayor que TensorFlow Lite	Bajo
Tiempo de carga	Reducido por el formato optimizado	Depende del tamaño y arquitectura del modelo	Reducido
Compatibilidad con modelos densos	Alta	Alta	No aplica
Compatibilidad con el modelo LSTM utilizado	No se utilizó para este modelo	Alta	No aplica
Necesidad de conversión	Sí, desde el modelo entrenado en Keras	No, se conserva el archivo entrenado	Se guarda después de ajustar el codificador
Archivos principales	modelo_abecedario_pro.tflite y modelo_numeros.tflite	Modelo de palabras en formato .h5	Codificadores de clases en formato .pickle
Ventaja principal	Reduce la carga necesaria para ejecutar los modelos estáticos	Conserva directamente la arquitectura secuencial del modelo de palabras	Mantiene la correspondencia entre la salida numérica y la clase reconocida
Limitación principal	La conversión puede presentar incompatibilidades con algunas operaciones o capas	Requiere cargar TensorFlow/Keras y consume más recursos	Debe coincidir exactamente con las clases utilizadas durante el entrenamiento
Uso final en el proyecto	Inferencia de letras y números	Inferencia de palabras	Interpretación de la clase obtenida por cada modelo
División del dataset	Posible, aunque puede apoyarse en Scikit-learn	Posible	Sí mediante train_test_split
Codificación de etiquetas	Posible	Posible	Sí mediante LabelEncoder y otras herramientas
Conversión para sistemas embebidos	Sí, mediante TensorFlow Lite	Requiere procesos o herramientas adicionales	No aplica directamente
Formato de guardado de modelos	.h5, .keras, SavedModel y conversión a .tflite	.pt o .pth	.pkl, .joblib u otros
Ejecución local de modelos	Sí	Sí	Sí para modelos compatibles
Documentación disponible	Amplia	Amplia	Amplia
Licencia	Código abierto	Código abierto	Código abierto
Función dentro del proyecto	Construcción, entrenamiento y ejecución de los modelos	Alternativa para crear redes neuronales	División de datos, codificación de etiquetas, métricas y matriz de confusión
Ventaja principal	Integra la creación de modelos mediante Keras y permite convertirlos a TensorFlow Lite	Proporciona flexibilidad y control durante el diseño de las redes	Facilita la preparación de los datos y el cálculo de métricas

Limitación principal	En la Jetson Nano requiere versiones y dependencias compatibles	El despliegue embebido requiere pasos adicionales respecto a TensorFlow Lite	No está orientada principalmente a redes profundas ni modelos LSTM
Uso dentro del proyecto	Entrenamiento de abecedario, números y palabras; ejecución del modelo dinámico y conversión de modelos estáticos	No utilizado	Herramienta complementaria durante la preparación y evaluación
Resultado de la comparación	**Seleccionado como framework principal**	No seleccionado	**Seleccionado como herramienta complementaria**

Nota. Elaboración propia con base en la documentación

Como se observa en la Tabla D.7, los modelos de abecedario y números se convirtieron a TensorFlow Lite para facilitar su ejecución local. El modelo de palabras se mantuvo en formato .h5, ya que procesa secuencias de movimiento mediante capas LSTM. Además, los archivos LabelEncoder permiten convertir las salidas numéricas de los modelos en las letras, números o palabras que se muestran en la interfaz.

D.8 Herramientas utilizadas para el servidor local, la base de datos y la interfaz web

Para implementar el acceso docente, el almacenamiento de los resultados y la presentación del dashboard se utilizaron Flask, SQLite, HTML, CSS y JavaScript. Además, se empleó el formato CSV para exportar los registros de las actividades.

A diferencia de los apartados anteriores, estas herramientas no deben presentarse como alternativas que compitieron entre sí, porque cada una cumple una función distinta dentro del sistema. Por esta razón, la Tabla D.8 muestra su integración y uso dentro de la aplicación final, en lugar de señalar una única herramienta seleccionada.

Como se presenta en la Tabla D.8, Flask permite ejecutar el servidor local mediante Python; SQLite conserva los datos generados por las sesiones; HTML, CSS y JavaScript conforman el dashboard; y CSV se utiliza para exportar los resultados. El sistema permite acceder al panel docente mediante una dirección IP local desde el navegador de otro dispositivo.

Tabla D.8
Matriz de herramientas utilizadas para el servidor local, almacenamiento e interfaz web

Característica	Flask	SQLite	HTML	CSS	JavaScript	CSV
Tipo de herramienta	Framework web para Python	Motor de base de datos relacional	Lenguaje de marcado	Lenguaje de estilos	Lenguaje de programación para la interfaz web	Formato de archivo tabular
Función principal	Ejecutar el servidor local y gestionar las rutas del dashboard	Almacenar y consultar la información generada por el sistema	Organizar el contenido de las páginas	Definir la presentación visual de la interfaz	Gestionar acciones y cambios dinámicos en el navegador	Exportar los resultados registrados
Entorno de ejecución	NVIDIA Jetson Nano	NVIDIA Jetson Nano	Navegador web	Navegador web	Navegador web	Computadora, teléfono o programa compatible
Integración con Python	Directa	Directa mediante el módulo sqlite3	Mediante plantillas y rutas de Flask	Mediante los archivos HTML	Mediante las páginas servidas por Flask	Mediante lectura y escritura de archivos
Necesidad de servidor externo	No	No	Depende del servidor Flask local	Depende del servidor Flask local	Depende del servidor Flask local	No
Funcionamiento sin internet	Sí	Sí	Sí, dentro de la red local	Sí, dentro de la red local	Sí, dentro de la red local	Sí
Acceso desde otro dispositivo	Sí, mediante dirección IP y puerto local	No directamente	Sí, desde el navegador	Sí, como parte de la página	Sí, como parte de la página	Sí, después de descargar el archivo
Gestión de usuarios	Procesa el inicio de sesión y las solicitudes del docente	Almacena los datos de docentes y estudiantes	Presenta los formularios	Organiza visualmente los formularios	Valida o controla acciones de la interfaz	No aplica
Gestión de actividades	Recibe y procesa las acciones enviadas desde el dashboard	Conserva sesiones, prácticas y resultados	Muestra controles y actividades	Define su apariencia	Gestiona botones y actualizaciones	Presenta una copia exportada de los registros
Consulta de resultados	Recupera la información desde SQLite y la envía al navegador	Ejecuta consultas sobre los datos almacenados	Muestra tablas, indicadores e historial	Organiza la presentación de los resultados	Permite filtros, cambios o actualizaciones visuales	Permite revisar los resultados fuera del sistema
Persistencia de los datos	No almacena datos por sí solo	Sí	No	No	No	Sí, como archivo exportado
Facilidad de mantenimiento	Alta por su integración con Python	Alta debido a su estructura local y compacta	Alta	Alta	Media	Alta
Consumo de recursos	Bajo o moderado	Bajo	Bajo	Bajo	Bajo o moderado	Bajo

Ventaja principal	Permite crear un servidor local ligero integrado con Python	No requiere instalar ni administrar un servidor de base de datos independiente	Estructura las páginas del dashboard	Facilita adaptar la interfaz a diferentes pantallas	Mejora la interacción del usuario con el dashboard	Facilita el análisis y respaldo de resultados
Limitación principal	No está pensado para despliegues web de gran escala sin configuración adicional	Presenta menor capacidad para múltiples usuarios simultáneos que una base de datos cliente-servidor	No incorpora lógica ni almacenamiento por sí solo	Solo controla la apariencia visual	Depende del navegador y debe coordinarse con el servidor	No actualiza automáticamente la información después de ser descargado
Uso dentro del proyecto	Servidor local y gestión de rutas	Base de datos de docentes, estudiantes, prácticas y resultados	Estructura del dashboard	Diseño visual del dashboard	Acciones dinámicas de la interfaz	Exportación de resultados
Resultado	**Utilizado**	**Utilizado**	**Utilizado**	**Utilizado**	**Utilizado**	**Utilizado como formato de exportación**

Nota. Elaboración propia con base en la documentación

Como se observa en la Tabla D.8, Flask se utilizó para ejecutar el servidor local y conectar el dashboard con la aplicación. SQLite permitió almacenar la información dentro del prototipo sin depender de internet. HTML, CSS y JavaScript se emplearon para desarrollar la interfaz web, mientras que el formato CSV permitió descargar los resultados para su revisión.

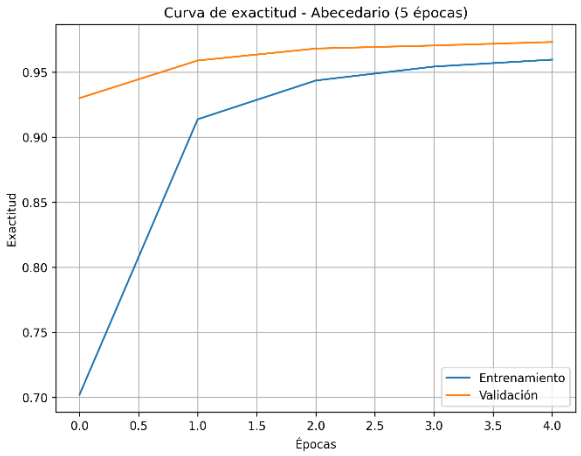
Anexo E: Resultados detallados de los modelos y matrices de confusión

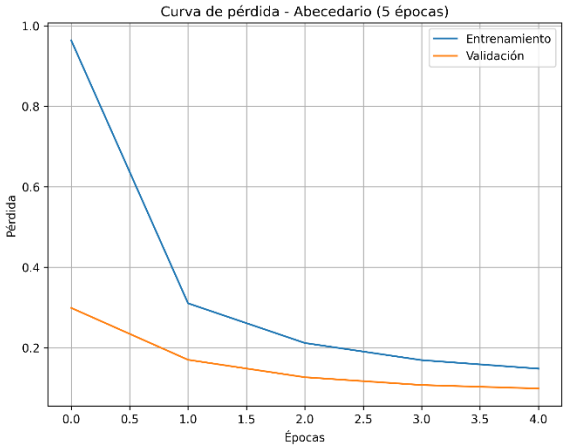
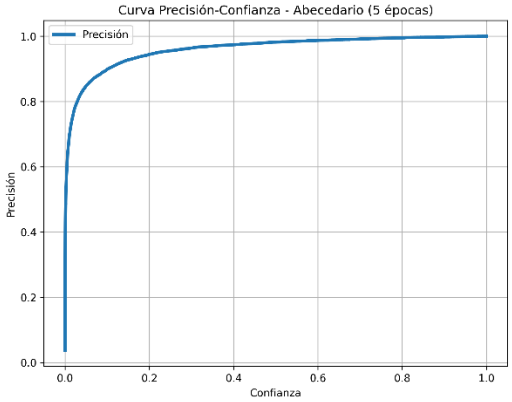
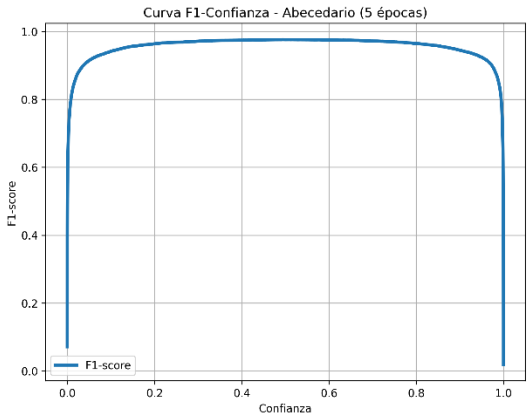
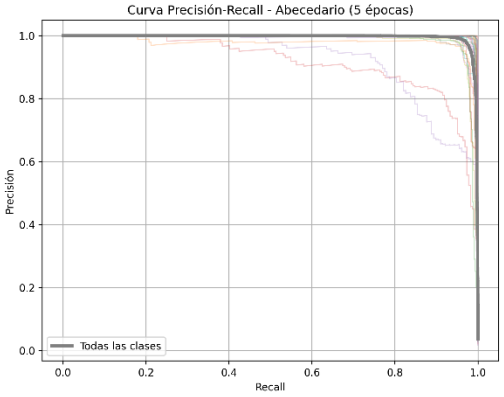
En este anexo se presentan los resultados completos obtenidos durante el entrenamiento de los modelos de reconocimiento del abecedario, números y palabras. Para cada modelo se realizaron pruebas con 5, 25, 50, 100 y 200 épocas. Los resultados se organizaron en tablas que incluyen la métrica evaluada, los principales cambios observados y la gráfica correspondiente. Se consideraron la exactitud, pérdida, precisión–confianza, F1–confianza, precisión–recall y matriz de confusión. Finalmente, se compararon los resultados para seleccionar la cantidad de épocas utilizada en cada modelo.

Resultados del modelo del abecedario

En este apartado se presentan los resultados del modelo encargado de reconocer las señas del abecedario. El entrenamiento se evaluó con cinco cantidades de épocas para observar los cambios en la exactitud, la pérdida y los errores de clasificación. Las Tablas E1 a E5 muestran los resultados de cada entrenamiento, mientras que la Tabla E6 presenta la comparación final.

Tabla E1.
Rendimiento del modelo a 5 épocas

Métrica	Análisis del resultado	Gráfica
Exactitud	Alcanzó un 97% de precisión global, demostrando una asimilación rápida pero aún inestable.	

Pérdida	Disminuyó drásticamente (0.14 en entrenamiento), mostrando la corrección de errores tempranos.	
Precisión-confianza	Se mantuvo elevada, aunque presentó falsos positivos en los umbrales de menor seguridad.	
F1-confianza	El valor F1 fue del 97%, evidenciando un balance inicial aceptable pero mejorable.	
Precisión-recall	Curva cercana a la zona superior, conservando una detección aceptable para la mayoría de clases.	

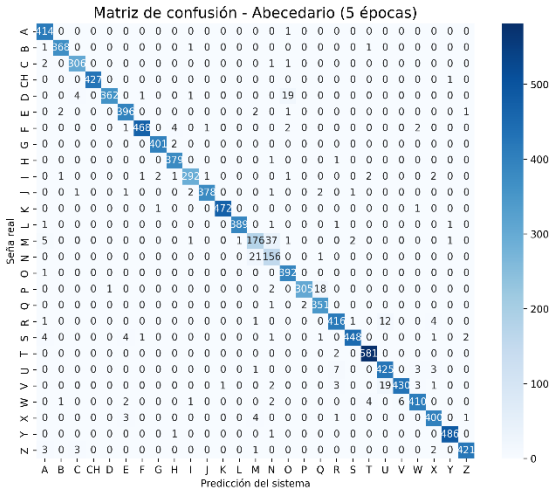
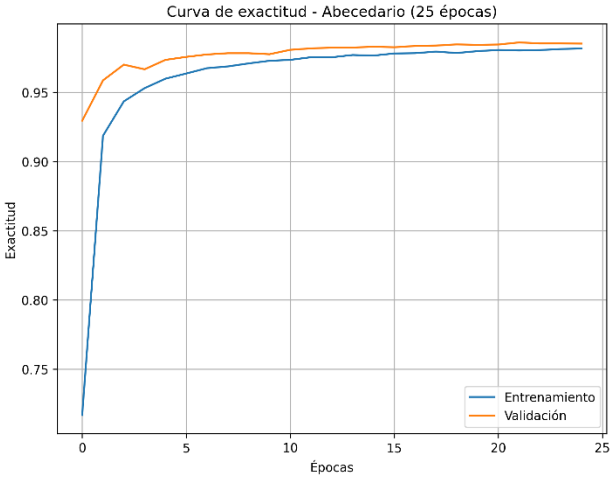
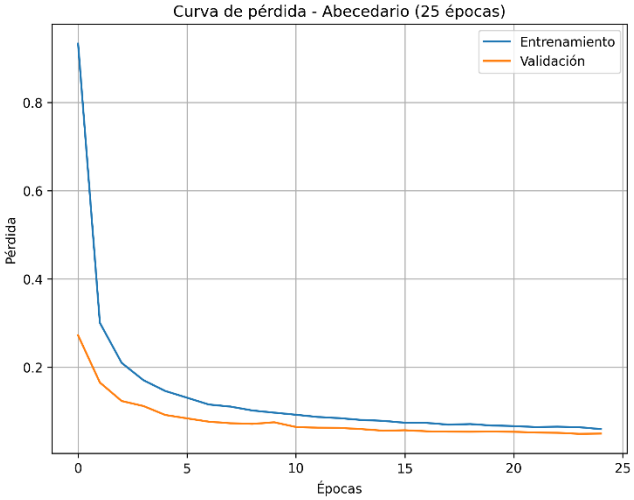
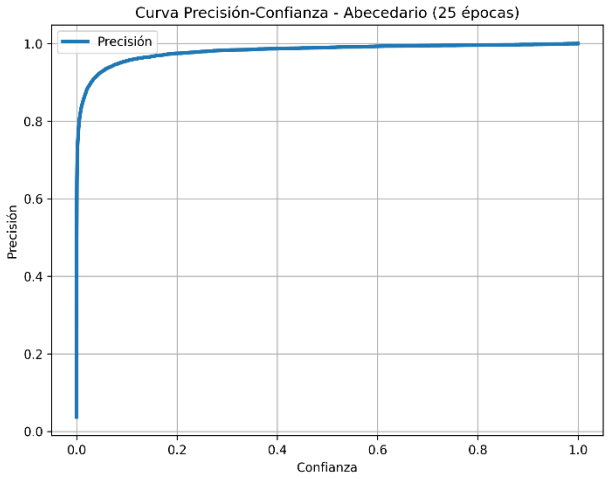
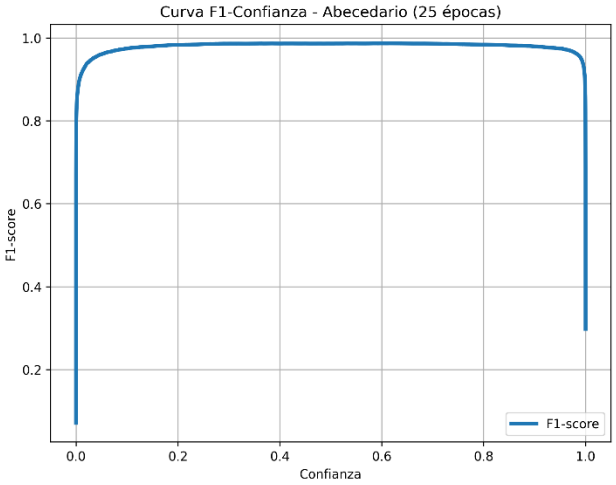
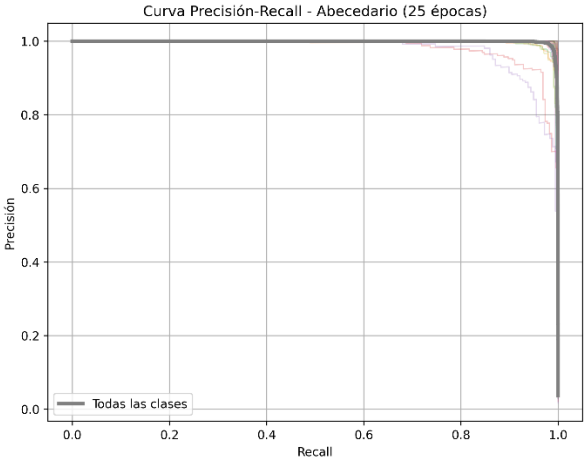
Matriz de confusión	Concentración en la diagonal, con confusiones marcadas entre señas similares como la M y la N.	

Tabla E2. *Rendimiento del modelo a 25 épocas*

Métrica	Análisis del resultado	Gráfica
Exactitud	Incrementó al 99%, mostrando curvas de entrenamiento y validación más unidas.	
Pérdida	Cayó a 0.05, reduciendo la brecha de error a la mitad respecto a las 5 épocas iniciales.	

Precisión-confianza	Presentó mayor estabilidad frente al aumento del nivel de confianza del algoritmo.	
F1-confianza	Aumentó a 99% global, logrando un equilibrio superior entre sensibilidad y precisión.	
Precisión-recall	El área bajo la curva mejoró, acercando el rendimiento general al escenario ideal (1.0).	

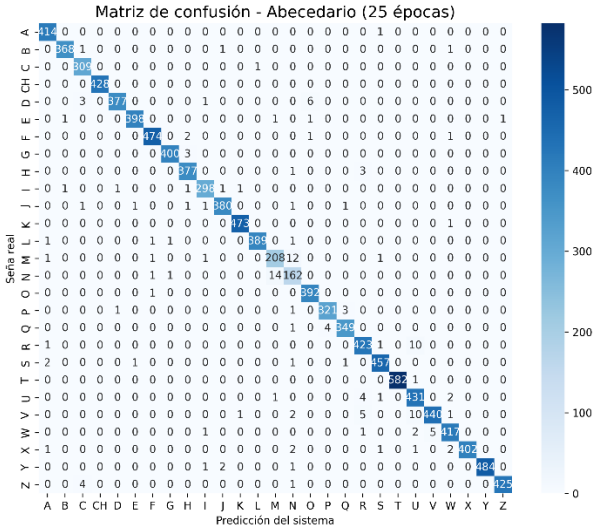
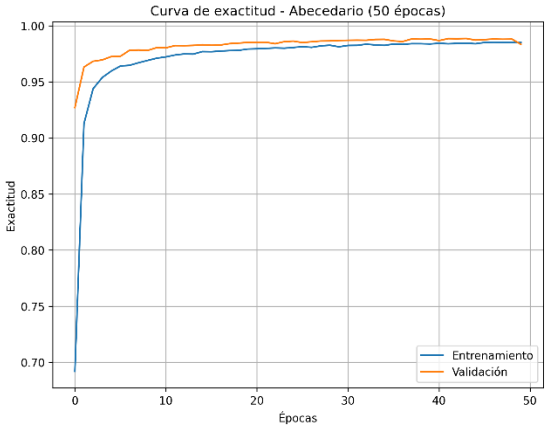
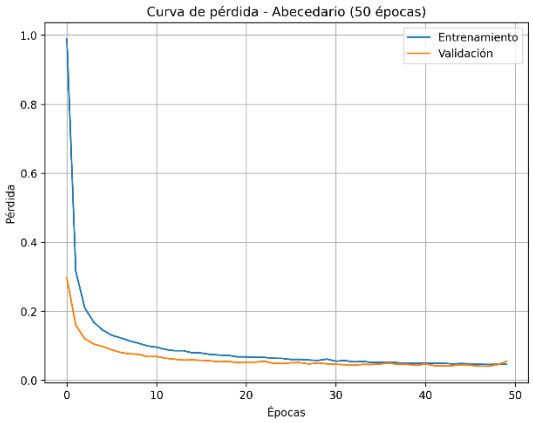
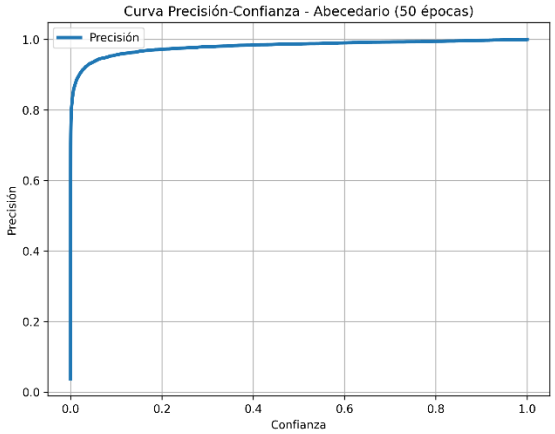
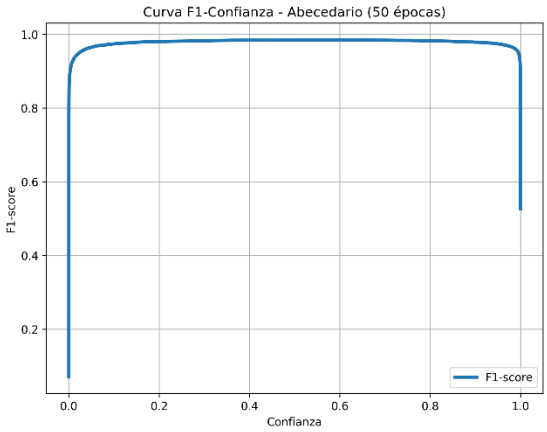
Matriz de confusión	Reducción significativa de errores; la letra M mejoró su detección reduciendo los falsos positivos.	

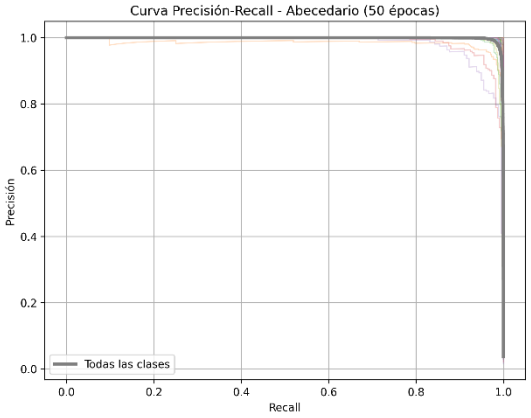
Tabla E3. *Rendimiento del modelo a 50 épocas*

Métrica	Análisis del resultado	Gráfica
Exactitud	Se mantuvo estable en un 98%, experimentando micro-ajustes en los pesos neuronales.	
Pérdida	Continuó su tendencia a la baja (0.046), consolidando el aprendizaje de patrones complejos.	

Precisión-confianza	Curva estable, indicando que el modelo realiza predicciones con alta certidumbre.
F1-confianza	Sostuvo el balance general (98%), preparando a la red para su fase de convergencia.
Precisión-recall	Margen de caída nulo, demostrando que no se sacrifica detección por exactitud.
Matriz de confusión	Diagonal principal sólida, aunque con ligeras variaciones transitorias en M Y N.







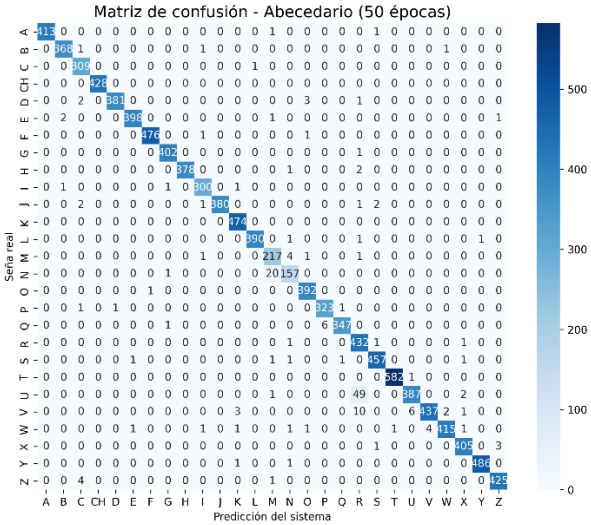
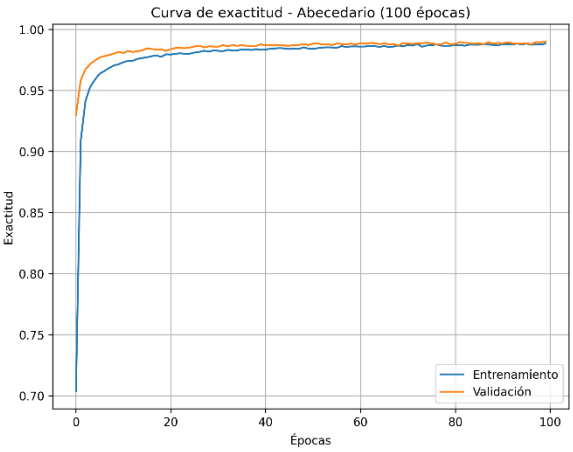
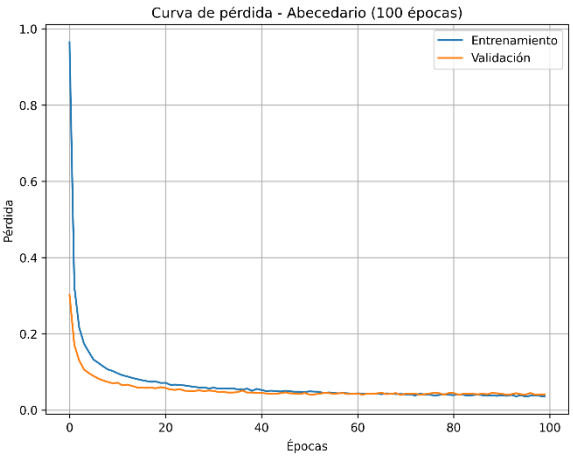
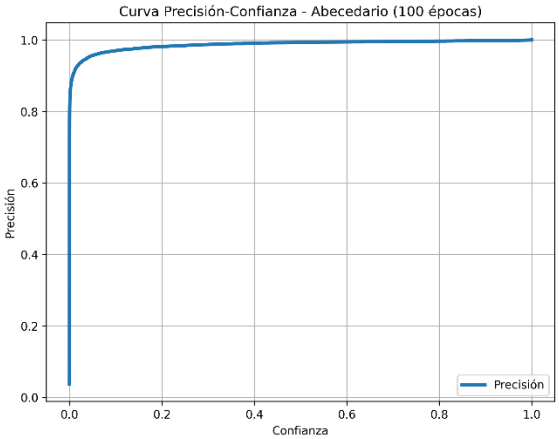


Tabla E4. *Rendimiento del modelo a 100 épocas*

Métrica	Análisis del resultado	Gráfica
Exactitud	Alcanzó su punto de máxima eficiencia (99%). Las curvas se acoplan sin signos de sobreajuste.	
Pérdida	Brecha mínima histórica entre entrenamiento (0.036) y validación (0.041). Es el punto de aprendizaje útil.	
Precisión-confianza	Inestabilidades eliminadas por completo. Confianza máxima en todo el umbral.	

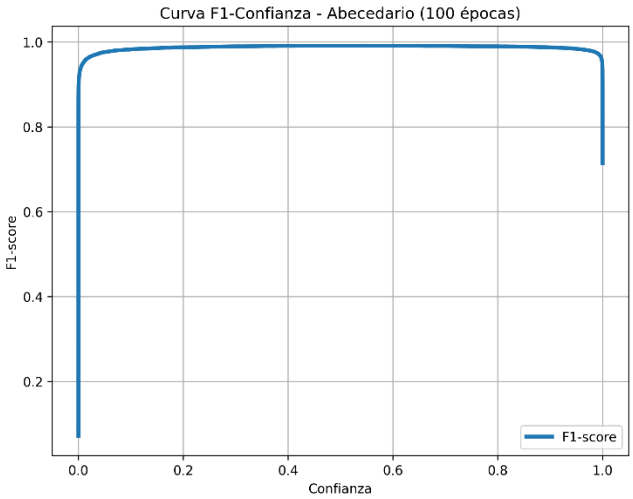
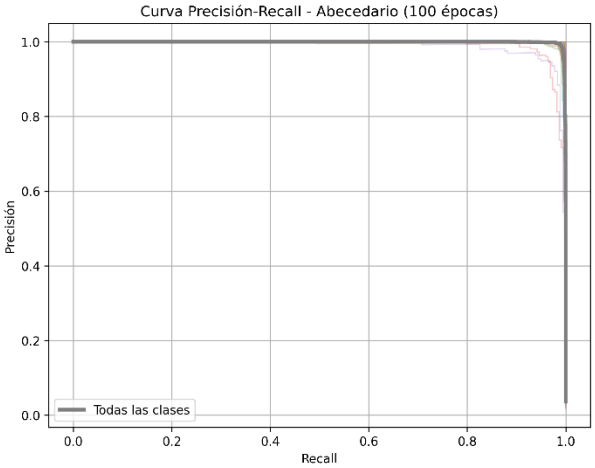
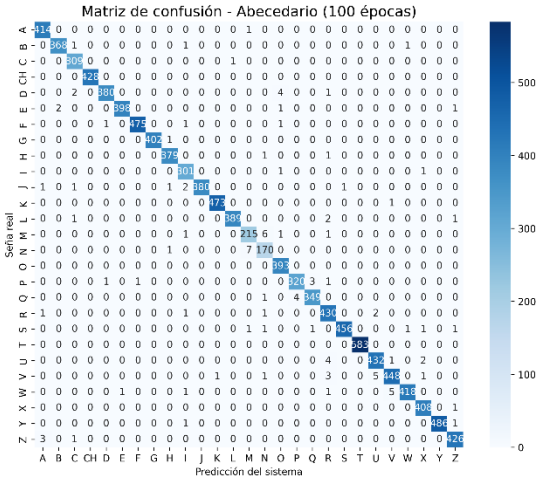
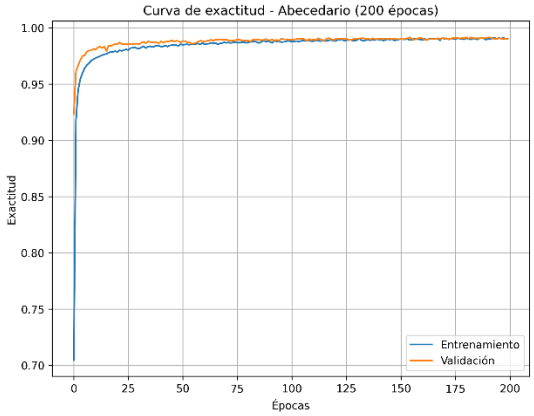
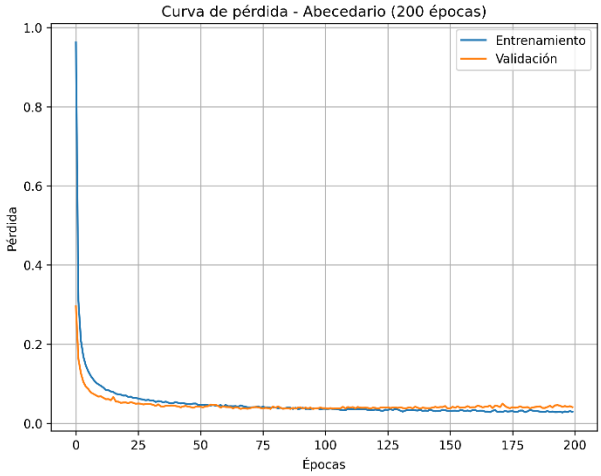
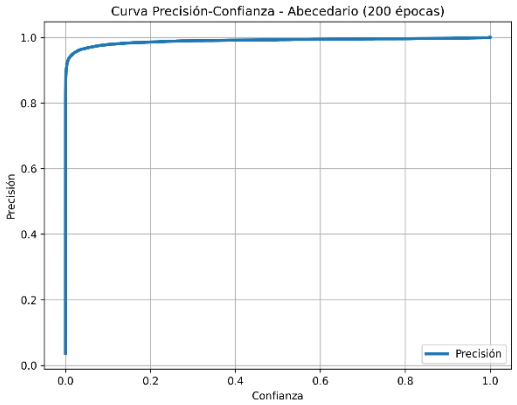
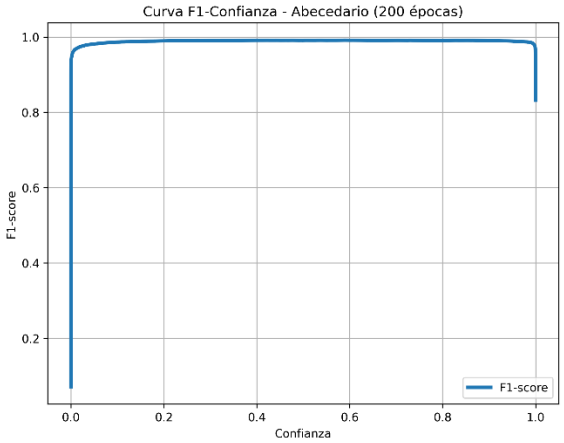
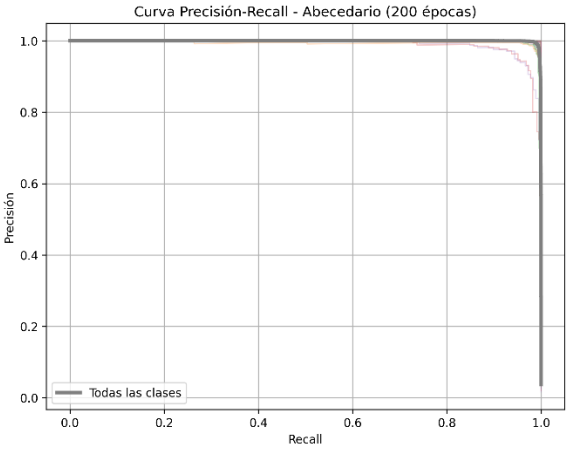
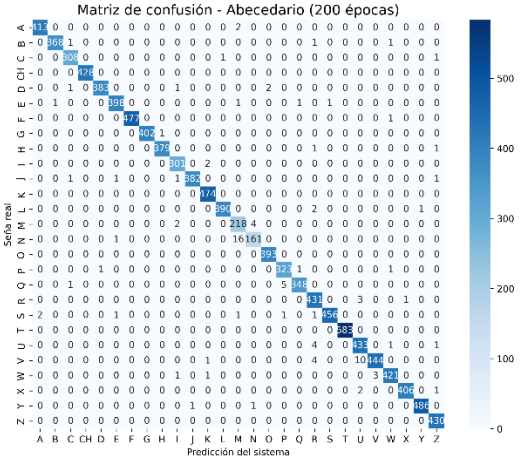
F1-confianza	El equilibrio llega al 99% absoluto. El sistema generaliza perfectamente los gestos.	
Precisión-recall	Área matemática perfecta; la red clasifica correctamente casi la totalidad de las muestras.	
Matriz de confusión	Diagonal principal limpia y sin desviaciones. Las confusiones iniciales de la M y N desaparecieron.	

Tabla E5. *Rendimiento del modelo a 200 épocas*

Métrica	Análisis del resultado	Gráfica

Exactitud	Estancada en 99%. No existen mejoras de predicción frente a la iteración de 100 épocas.	
Pérdida	La pérdida de entrenamiento baja, pero la de validación se congela, indicando sobreajuste (*overfitting*) incipiente.	
Precisión-confianza	Curvas matemáticamente idénticas a la fase de 100 épocas.	
F1-confianza	Puntuación estancada (99%), confirmando rendimientos decrecientes.	

Precisión-recall	Comportamiento sin alteraciones respecto al corte anterior.	
Matriz de confusión	Resultados idénticos a las 100 épocas. Se comprueba que continuar el entrenamiento es innecesario.	

La comparación mostró que el modelo mejoró al aumentar las épocas. La exactitud pasó de 97,32 % con 5 épocas a 99,06 % con 200 épocas. Además, los errores de clasificación disminuyeron de 288 a 101. Por esta razón, se seleccionó el modelo entrenado con 200 épocas.

Resultados del modelo de números

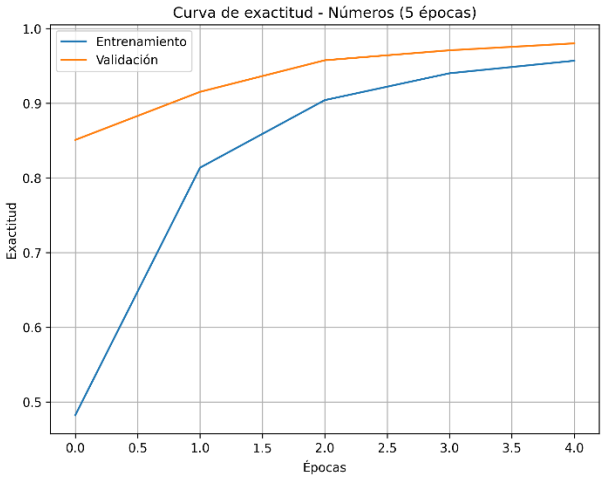
En este apartado se presentan los resultados del modelo utilizado para reconocer las señas de los números del 0 al 9. Se analizaron los cambios obtenidos con diferentes cantidades de épocas, especialmente en la exactitud, pérdida y distribución de los errores.

Tabla E6. *Rendimiento del modelo a 5 épocas*

Métrica	Análisis del resultado	Gráfica

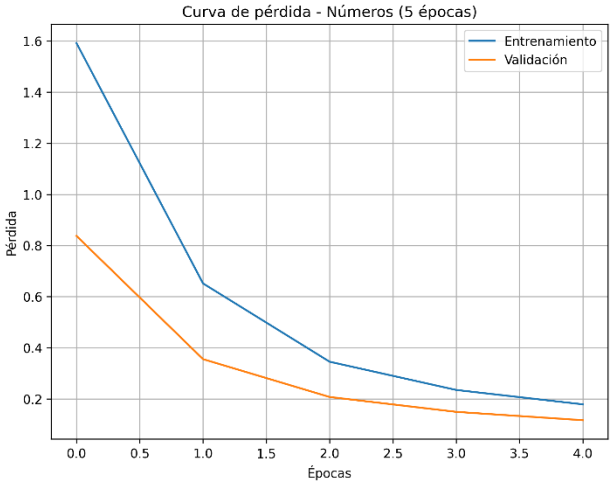
Exactitud

Arranca con un sorprendente 98% global, demostrando que los números son gestos espacialmente más sencillos de asimilar que el abecedario.



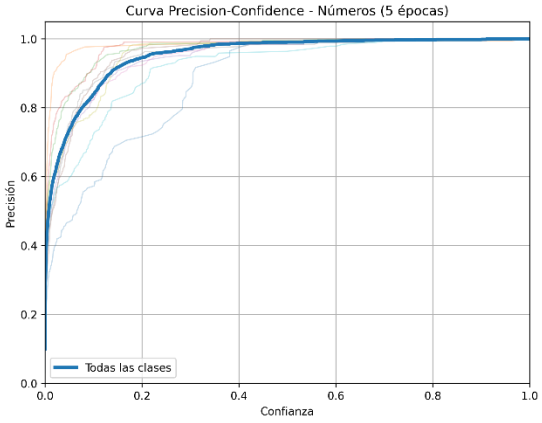
Pérdida

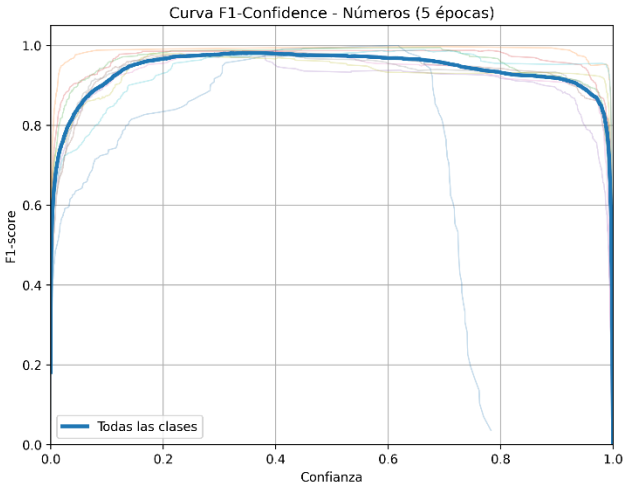
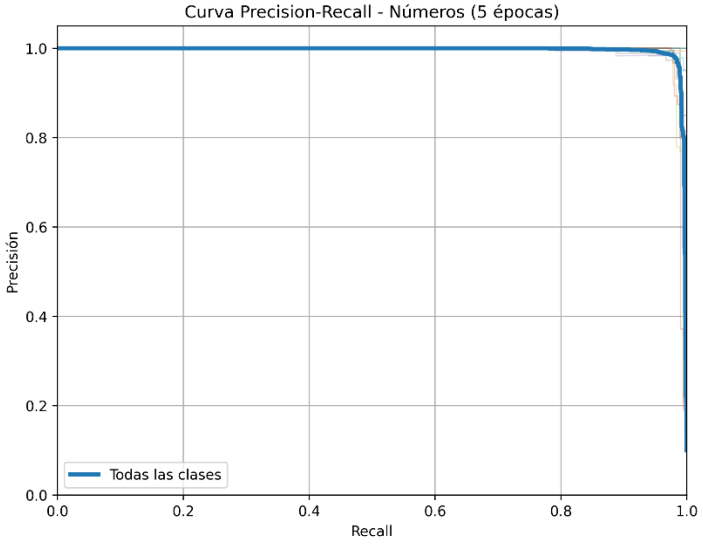
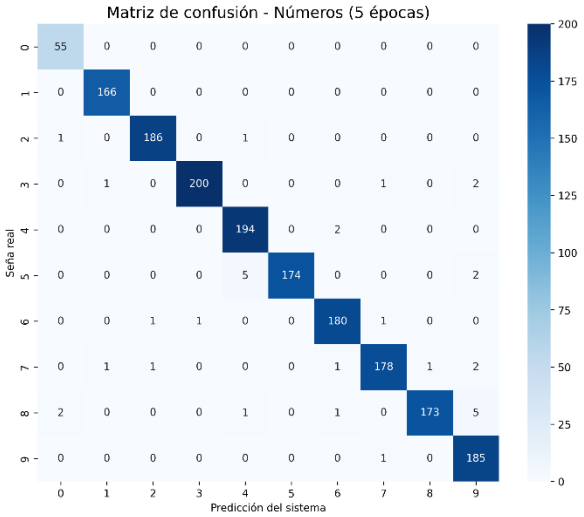
Aunque la exactitud es alta, el margen de error en entrenamiento (0.17) revela que las predicciones aún son titubeantes.



Precisión-confianza

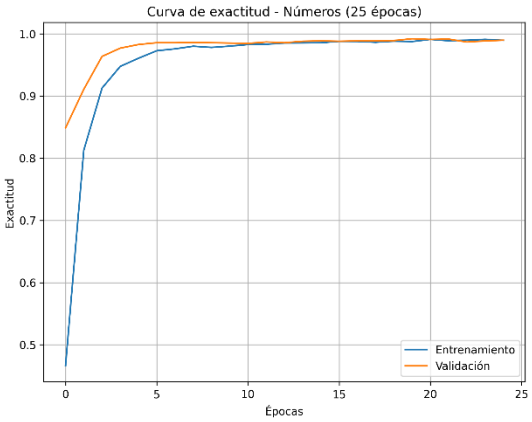
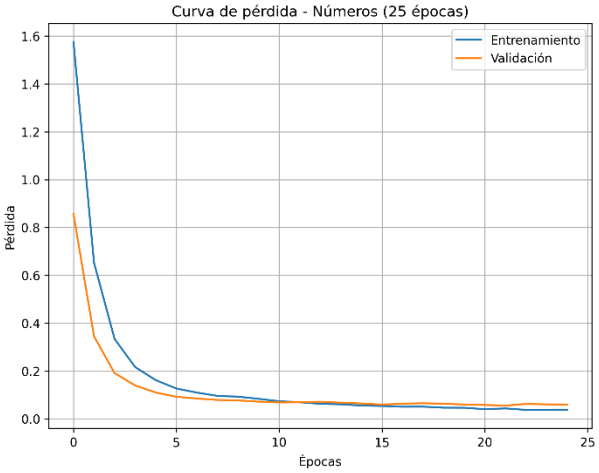
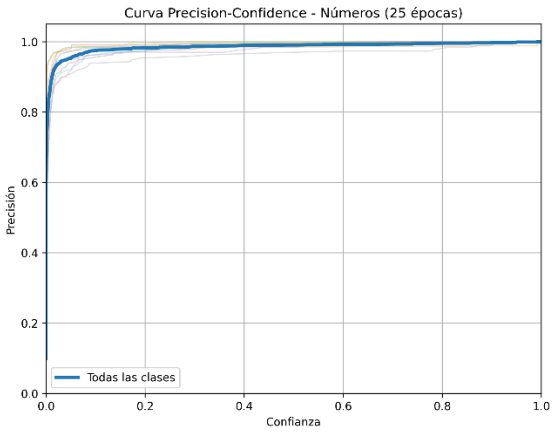
Se mantiene elevada en general, pero decae en los umbrales de baja confianza, reflejando inseguridad inicial.

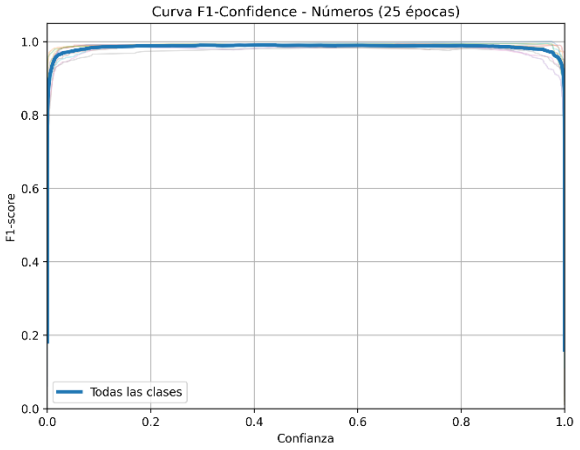
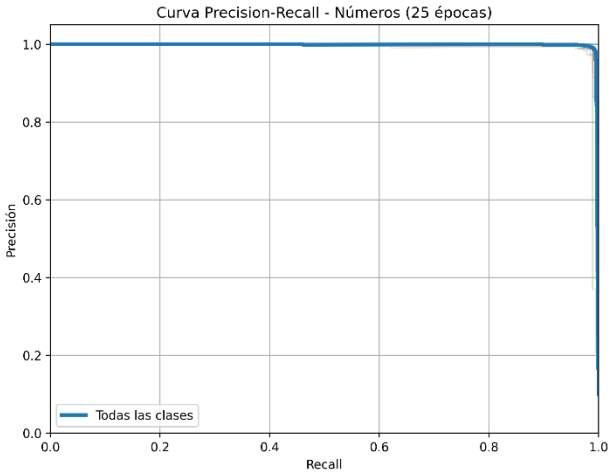
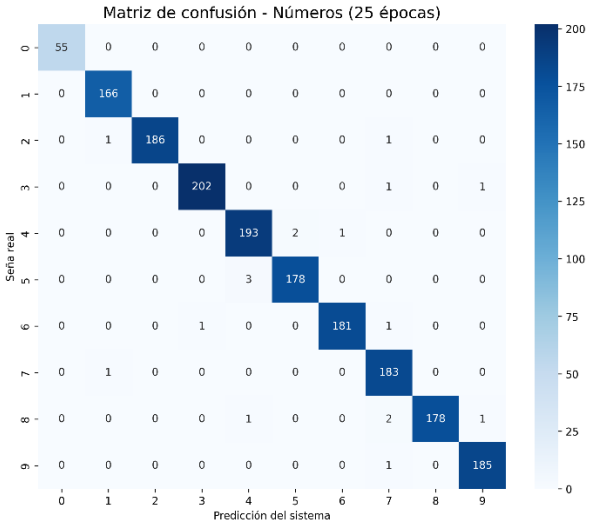


F1-confianza	El balance inicial es sólido (98%), pero la red aún muestra dudas con números específicos que requieren mayor contexto.	
Precisión-recall	Curva alta y estable, lo que indica que el sistema no está perdiendo muestras (falsos negativos) en esta etapa temprana.	
Matriz de confusión	Excelente agrupación en la diagonal principal. Sin embargo, los números "0" y "9" presentan las precisiones más bajas (95% y 94%), indicando ligera confusión morfológica inicial.	

Análisis de Entrenamiento (Números): 25 Épocas

Tabla E7. *Rendimiento del modelo a 25 épocas*

Métrica	Análisis del resultado	Gráfica
Exactitud	Sube a casi el 99%. Las curvas de validación y entrenamiento se emparejan, mostrando un aprendizaje sano y continuo.	
Pérdida	Descenso drástico. La validación cae a 0.05, lo que significa que la IA ya no solo "adivina", sino que entiende los patrones geométricos.	
Precisión-confianza	Las fluctuaciones se suavizan enormemente. El modelo se vuelve mucho más firme al emitir su clasificación.	

F1-confianza	Alcanza casi el 99% de equilibrio general. La sensibilidad de detección se iguala a la exactitud.	
Precisión-recall	El área bajo la curva se ensancha positivamente, rozando el escenario ideal para la detección en tiempo real.	
Matriz de confusión	Los errores del "0" y "9" detectados en la fase inicial se corrigen; el sistema aprende a discriminar la forma de los dedos.	

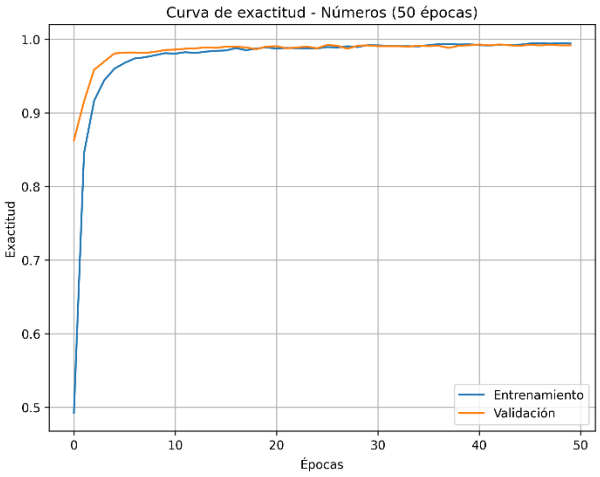
Análisis de Entrenamiento (Números): 50 Épocas

Tabla E8. *Rendimiento del modelo a 50 épocas*

Métrica	Análisis del resultado	Gráfica

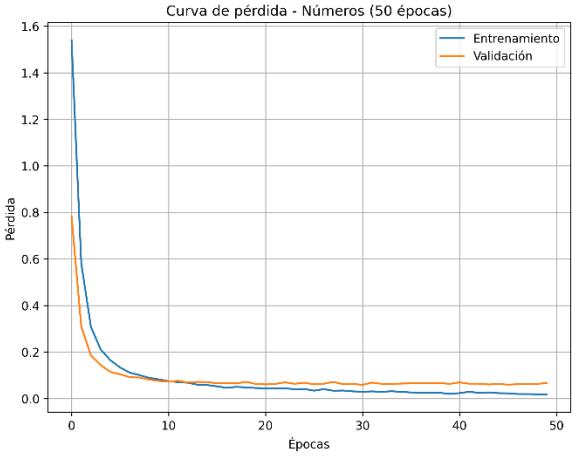
Exactitud

Supera la barrera del 99.1%. El modelo realiza micro-ajustes en los pesos neuronales para perfeccionar señas complejas.



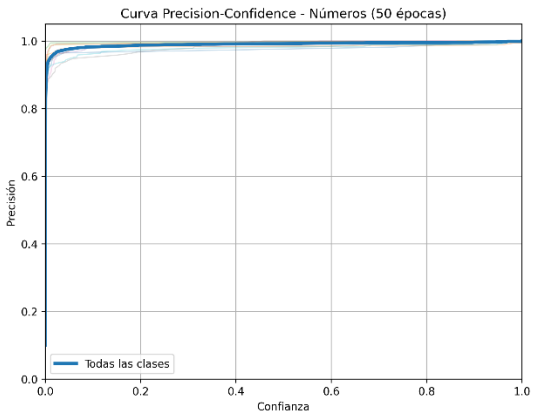
Pérdida

La pérdida de entrenamiento baja (0.017), pero la de validación se aplana (0.06). Primer indicio de que la red se acerca a su límite de aprendizaje útil.



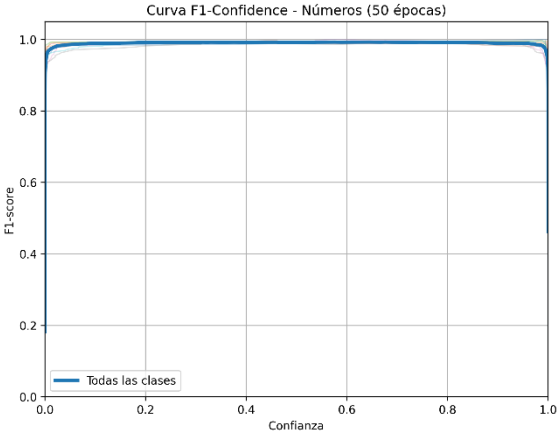
Precisión-confianza

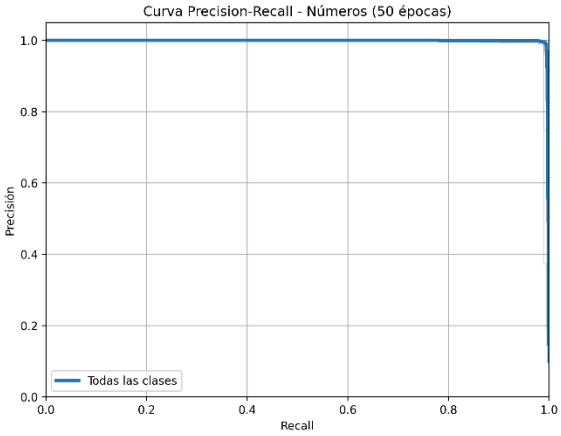
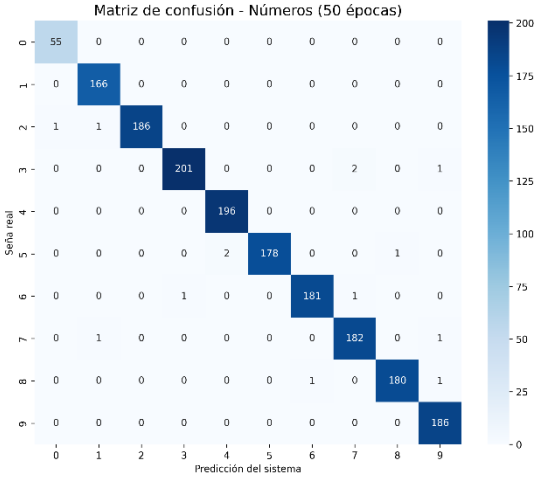
La curva es estable, indicando altísima seguridad en la mayoría de las clases evaluadas.



F1-confianza

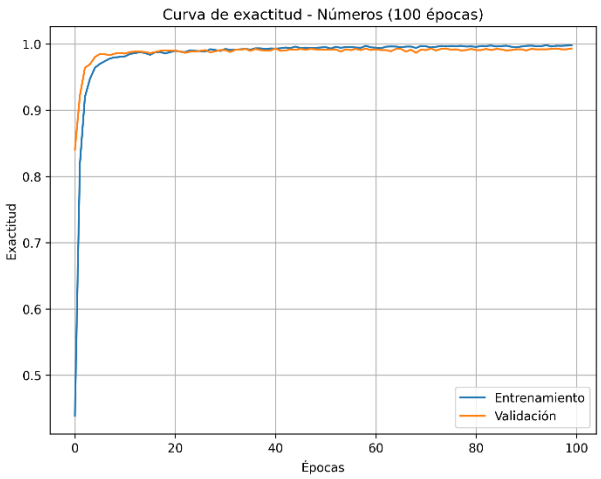
Sostiene su rendimiento global (99.1%), preparando al sistema para su iteración definitiva.



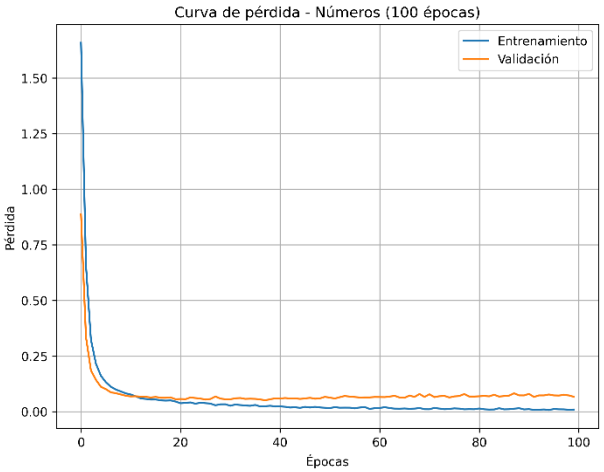
Precisión-recall	Comportamiento óptimo. El algoritmo es capaz de reconocer todos los números sin sacrificar precisión.	
Matriz de confusión	Diagonal principal prácticamente perfecta para los 10 dígitos (0 al 9).	

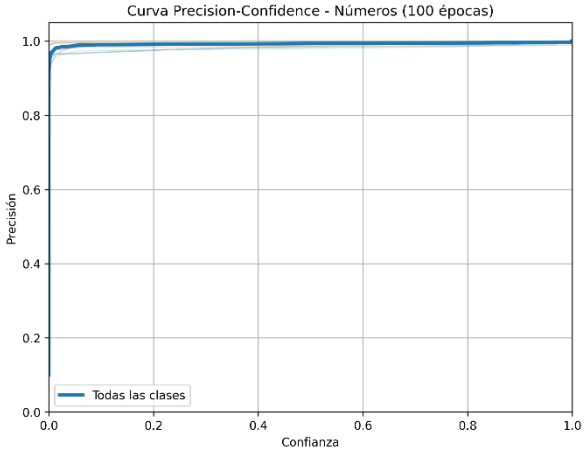
Análisis de Entrenamiento (Números): 100 Épocas

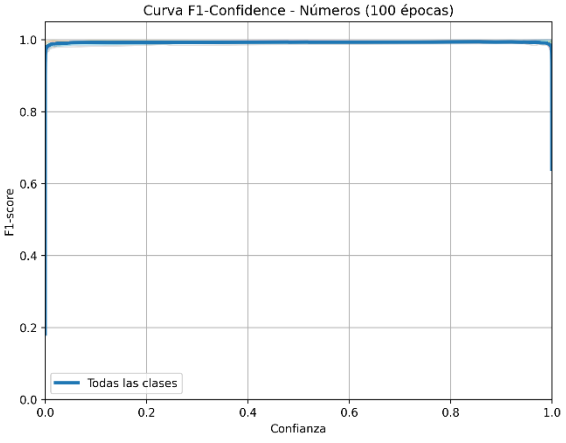
Tabla E9. *Rendimiento del modelo a 100 épocas*

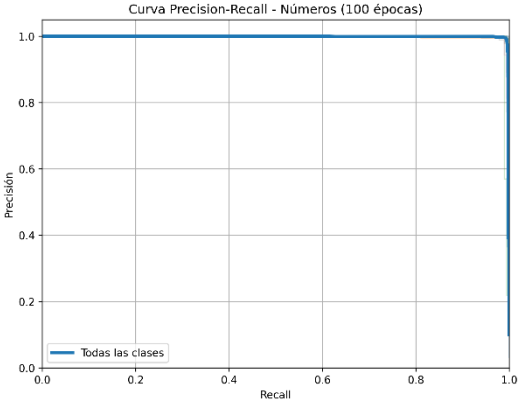
Métrica	Análisis del resultado	Gráfica
Exactitud	Punto de máxima eficiencia (99.3%). El sistema alcanza el "sweet spot" (punto óptimo) de convergencia matemática.	

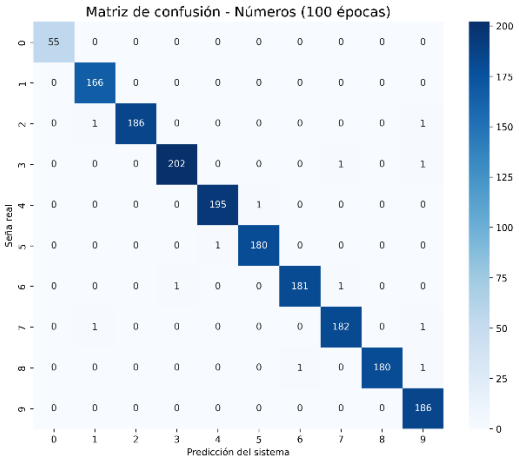
Pérdida	La pérdida de entrenamiento cae a 0.008. Es la última etapa donde la validación se mantiene estable antes de degradarse.
Precisión-confianza	Certeza absoluta. El modelo no duda al interpretar el número de dedos extendidos frente a la cámara.
F1-confianza	Equilibrio global del 99.3%. Todas las métricas armónicas llegan a su pico máximo.
Precisión-recall	Margen perfecto. No existen omisiones de detección (falsos negativos).





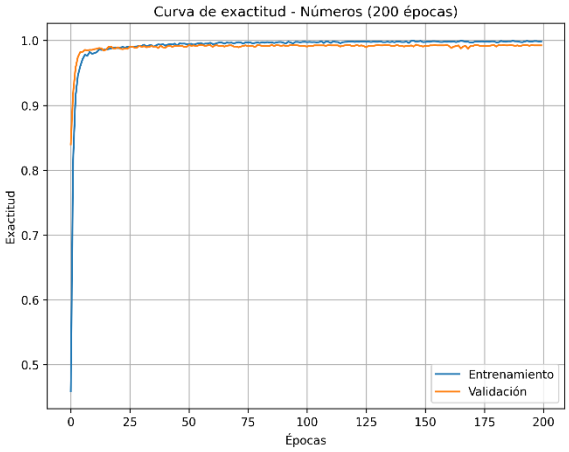
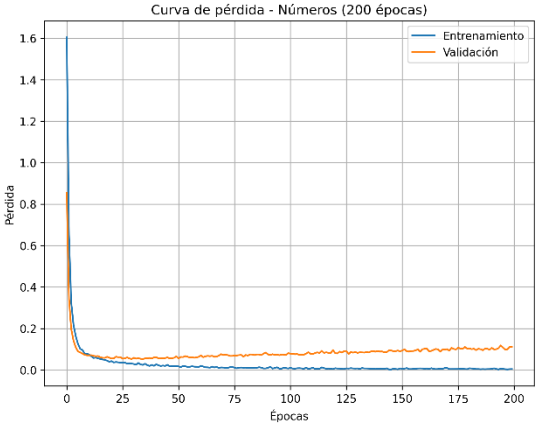




Matriz de confusión	Limpieza total. Todos los números registran puntuaciones F1 entre el 99% y el 100%.	

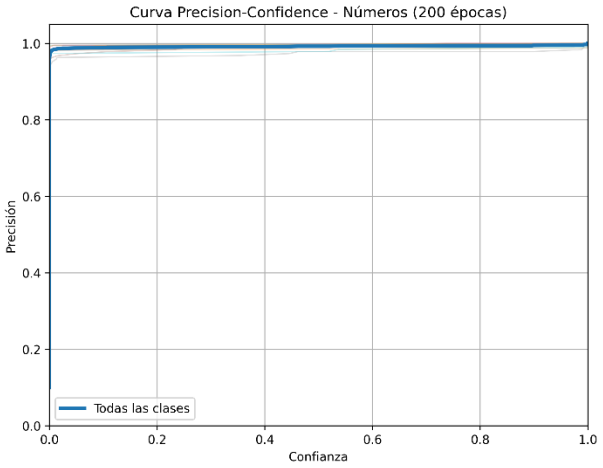
Análisis de Entrenamiento (Números): 200 Épocas (Efecto de Sobreajuste / Overfitting)

Tabla E10. *Rendimiento del modelo a 200 épocas*

Métrica	Análisis del resultado	Gráfica
Exactitud	En lugar de mejorar, la exactitud cae levemente a 99.2%. Síntoma claro de que el entrenamiento extra fue perjudicial.	
Pérdida	Evidencia de Overfitting: La pérdida de validación se dispara hasta 0.112, confirmando que la red memorizó los datos y perdió capacidad de generalizar.	

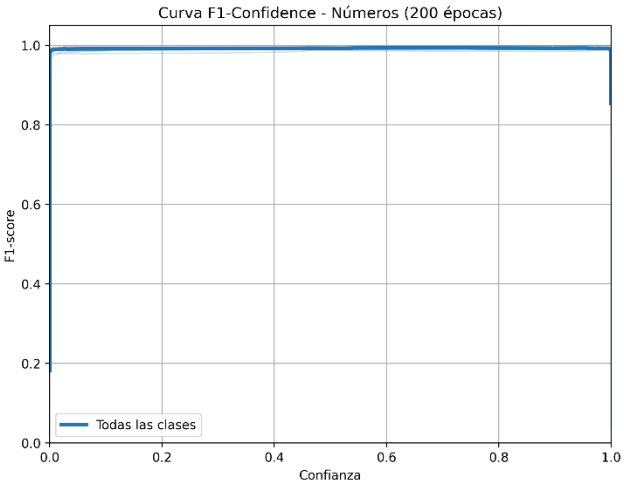
Precisión-confianza

Se mantienen las formas geométricas de las iteraciones anteriores, pero respaldadas por un modelo sobreentrenado.



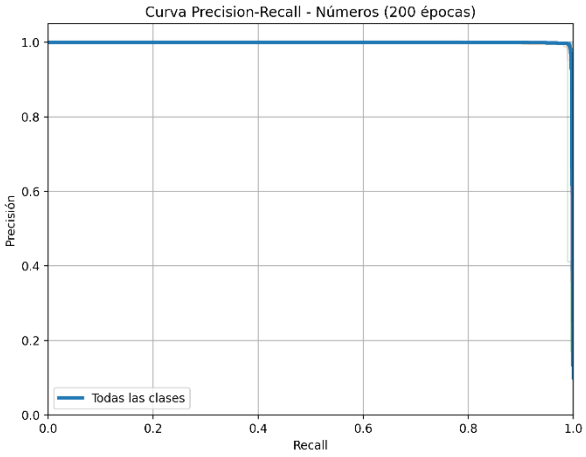
F1-confianza

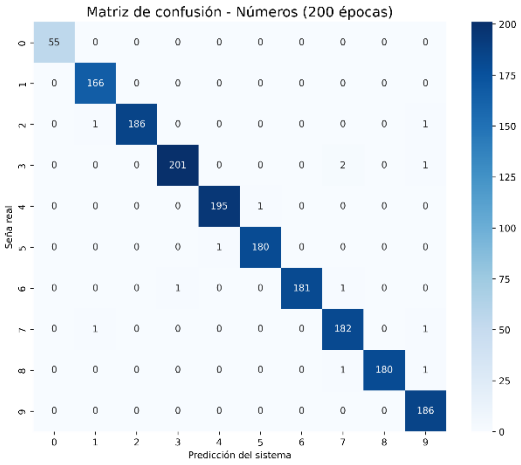
Leve retroceso al 99.2%. Refleja que seguir entrenando degradó la calidad general del algoritmo.



Precisión-recall

Las curvas se mantienen altas, pero el margen de pérdida (*loss*) previo anula cualquier beneficio.



Matriz de confusión	Similar a las 100 épocas. Se justifica técnicamente detener el modelo en la iteración 100 para preservar su utilidad en la vida real.	

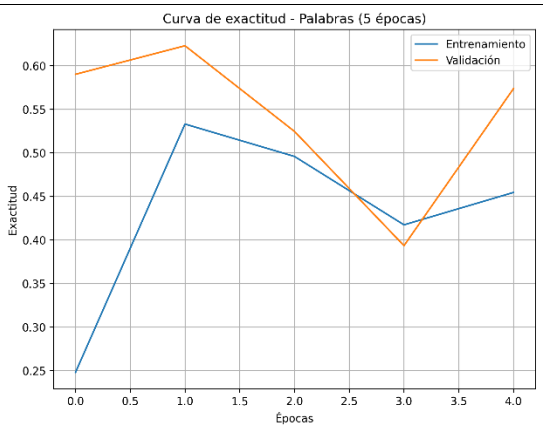
El modelo presentó una mejora progresiva hasta las 100 épocas, donde alcanzó una exactitud de 99,30 % y registró 12 errores. Con 200 épocas la exactitud disminuyó ligeramente a 99,25 % y la pérdida de validación aumentó. Por esta razón, se seleccionó el entrenamiento de 100 épocas.

Resultados del modelo de palabras

En este apartado se presentan los resultados del modelo de reconocimiento de las palabras ADIÓS, GRACIAS, HOLA, NADA, NO y SÍ. Al tratarse de señas que incluyen movimiento, el modelo necesitó más épocas para mejorar la identificación de las secuencias.

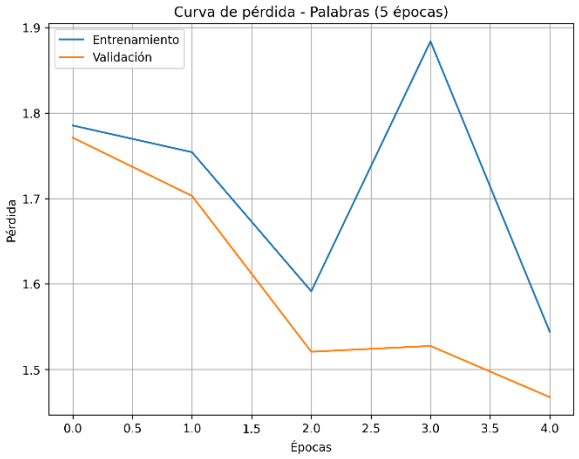
Análisis de Entrenamiento (Palabras): 5 Épocas

Tabla E10. *Rendimiento del modelo a 5 épocas*

Métrica	Análisis del resultado	Gráfica
Exactitud	Apenas alcanza un 57.3% global. Al procesar secuencias de movimiento (gestos dinámicos), el modelo requiere mucho más tiempo para comprender el contexto temporal.	

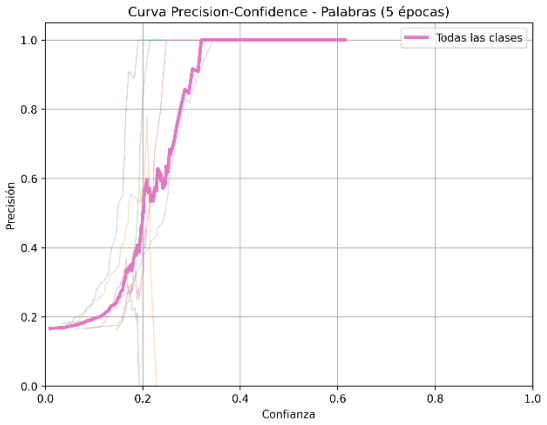
Pérdida

Nivel de error críticamente alto en entrenamiento (1.54) y validación (1.46). La red neuronal aún no encuentra los patrones espaciales.



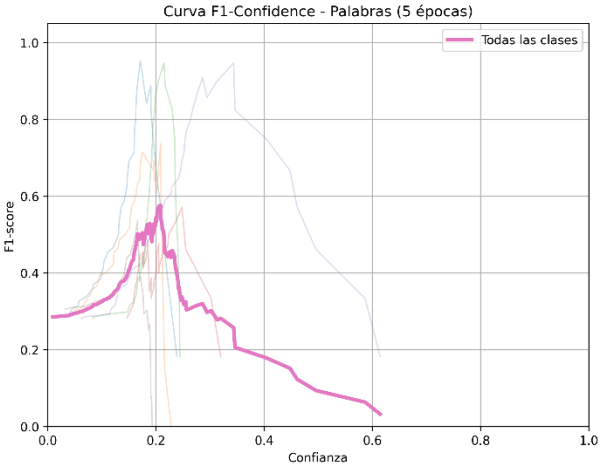
Precisión-confianza

Las curvas son altamente erráticas. El modelo emite predicciones con niveles de certidumbre muy bajos y llenos de falsos positivos.



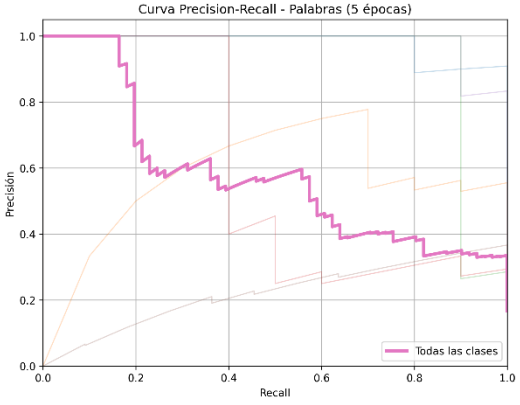
F1-confianza

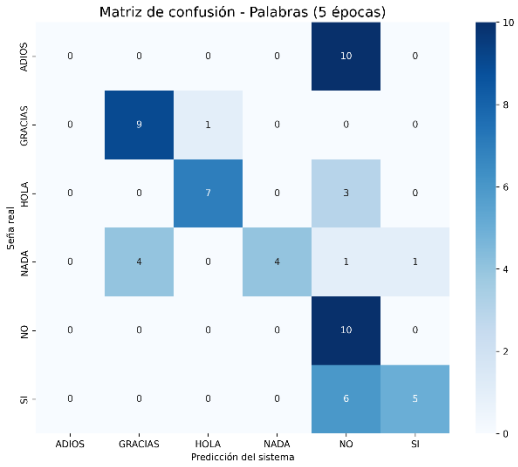
Desbalance severo. Gestos complejos como la palabra "ADIOS" registran un F1 de 0.00, lo que indica un fracaso total en su detección inicial.



Precisión-recall

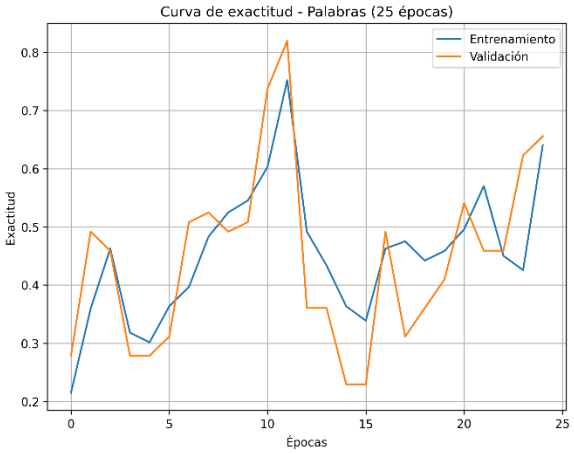
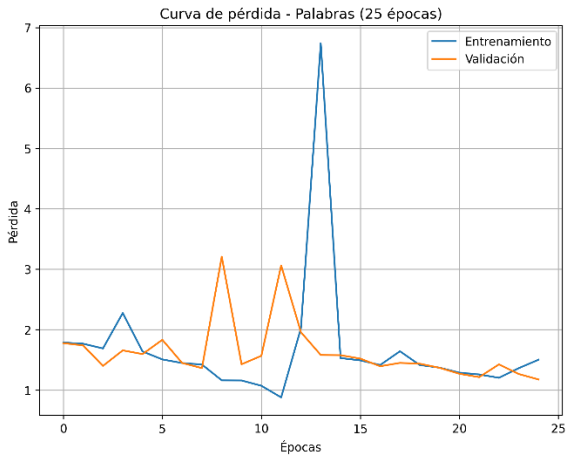
El área bajo la curva es deficiente, evidenciando que el sistema pierde gran parte de las muestras dinámicas en esta etapa temprana.



Matriz de confusión	Fuerte dispersión fuera de la diagonal principal. El modelo confunde groseramente las señas debido a la falta de madurez de los pesos neuronales.	

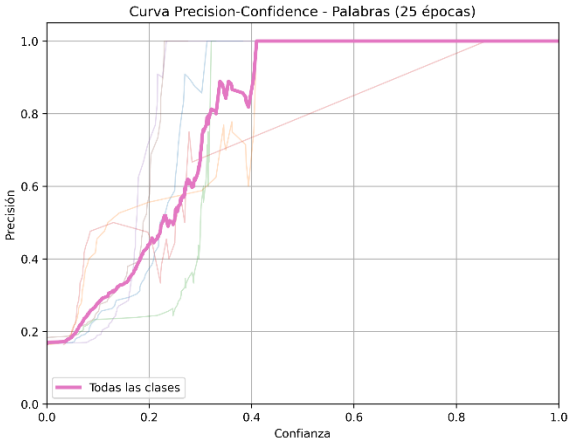
Análisis de Entrenamiento (Palabras): 25 Épocas

Tabla E11. *Rendimiento del modelo a 25 épocas*

Métrica	Análisis del resultado	Gráfica
Exactitud	Muestra un leve avance al 65.5%, pero sigue siendo insuficiente para aplicaciones reales de reconocimiento de lenguaje de señas.	
Pérdida	La pérdida general comienza a descender lentamente (1.17 en validación), mostrando una asimilación muy costosa matemáticamente.	

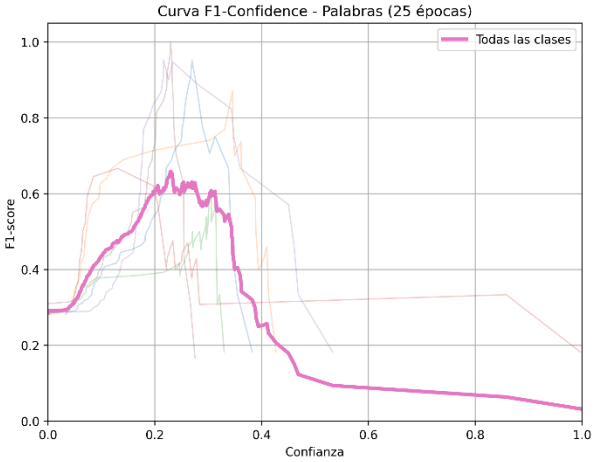
Precisión-confianza

Sigue presentando inestabilidad, especialmente en la identificación de gestos con recorridos cinemáticos similares.



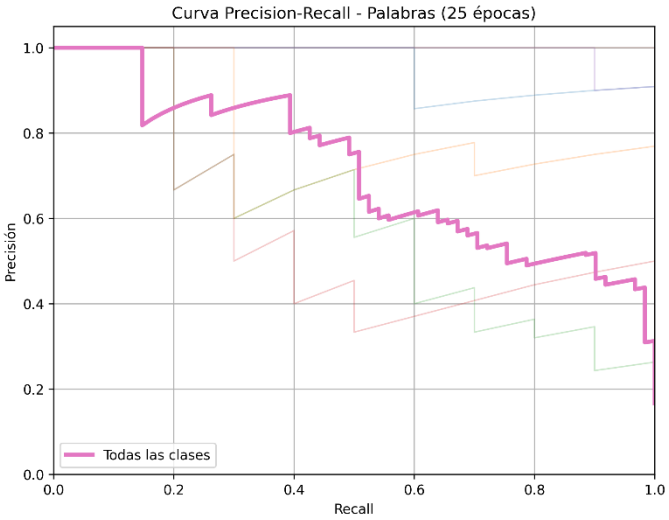
F1-confianza

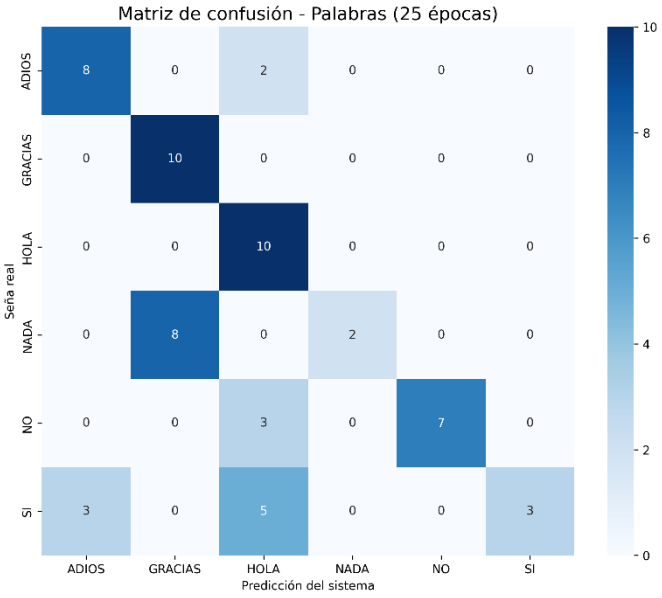
El equilibrio mejora ligeramente (62% global). Sin embargo, la palabra "NADA" sufre una caída drástica a un F1 de 0.33.



Precisión-recall

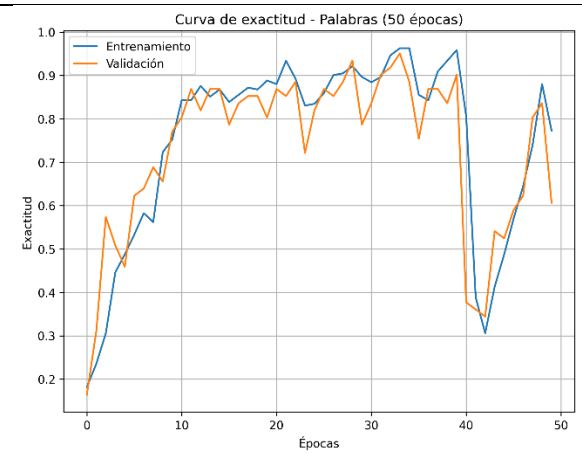
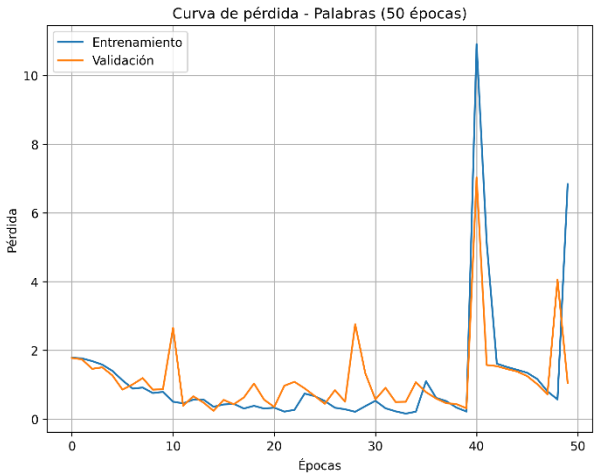
Refleja fluctuaciones negativas, confirmando que la red está sacrificando la sensibilidad general para intentar elevar la exactitud en ciertas palabras.



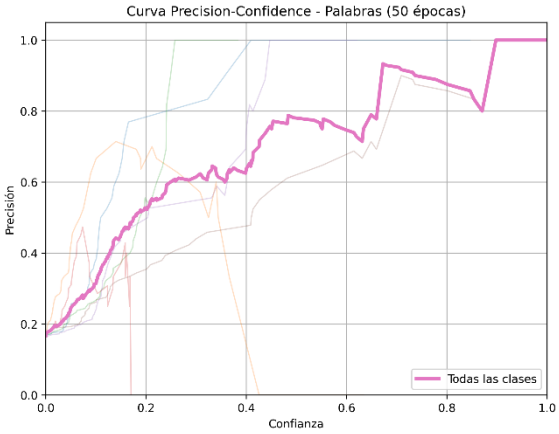
Matriz de confusión	Aunque la palabra "NO" mejora su detección (F1 de 0.82), la concentración de errores sigue siendo muy alta en el resto de las clases.	

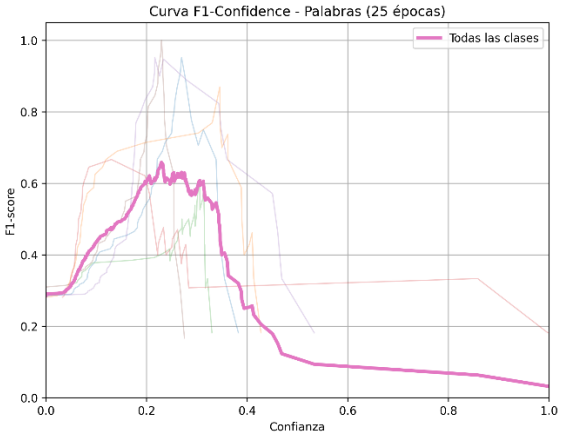
Análisis de Entrenamiento (Palabras): 50 Épocas (Inestabilidad y Caída)

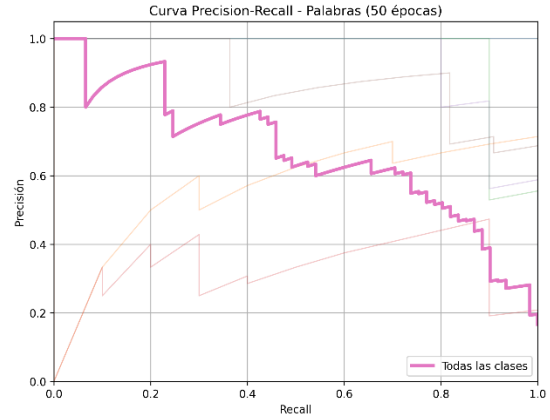
Tabla E12. *Rendimiento del modelo a 50 épocas*

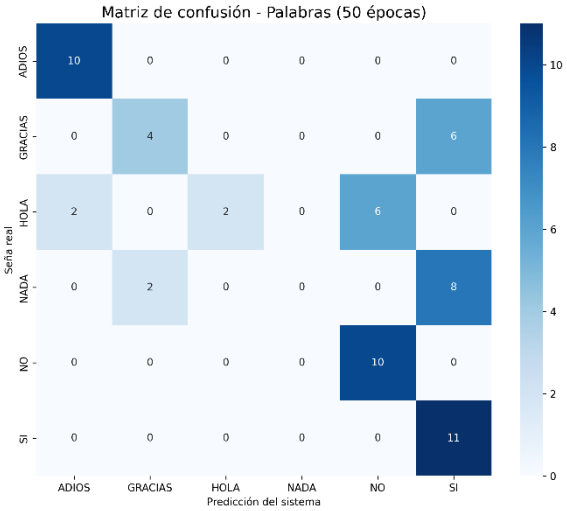
Métrica	Análisis del resultado	Gráfica
Exactitud	Sufre un retroceso al 60.6%. Esta inestabilidad es típica cuando el algoritmo intenta procesar la dimensión temporal del video y ajusta mal los pesos neuronales.	
Pérdida	La pérdida de entrenamiento sufre un pico anómalo, evidenciando un "tropiezo" matemático del modelo al intentar generalizar el *dataset* de gestos dinámicos.	

Precisión-confianza	Curvas colapsadas que demuestran pérdida de fiabilidad en los resultados.
F1-confianza	Descenso crítico al 52% global. La palabra "NADA" vuelve a fracasar completamente (F1 de 0.00) y "HOLA" cae al 0.33.
Precisión-recall	Deterioro severo en el área operativa, marcando el punto de peor desempeño de la red antes de su recuperación estructural.
Matriz de confusión	Alta dispersión de falsos negativos. La red se muestra incapaz de resolver el componente dinámico con apenas 50 iteraciones.



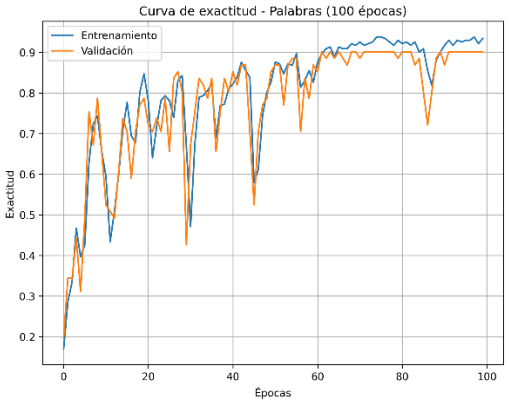
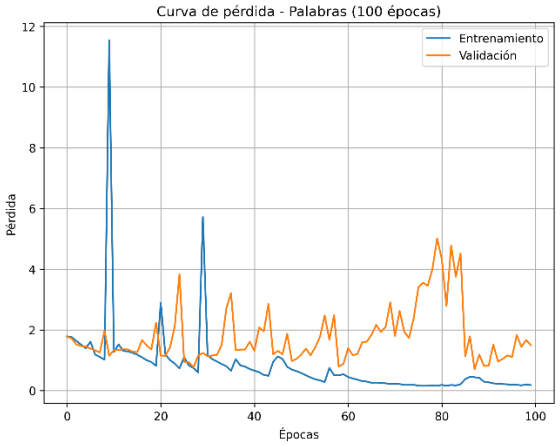
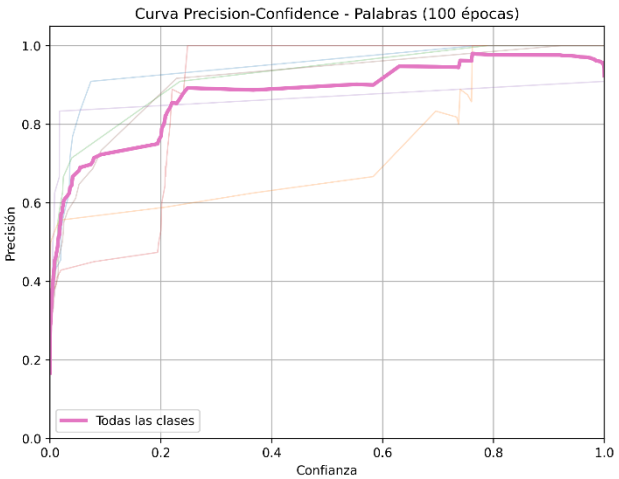


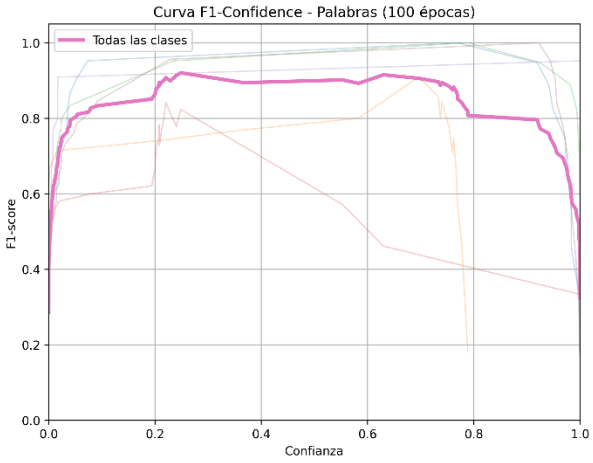
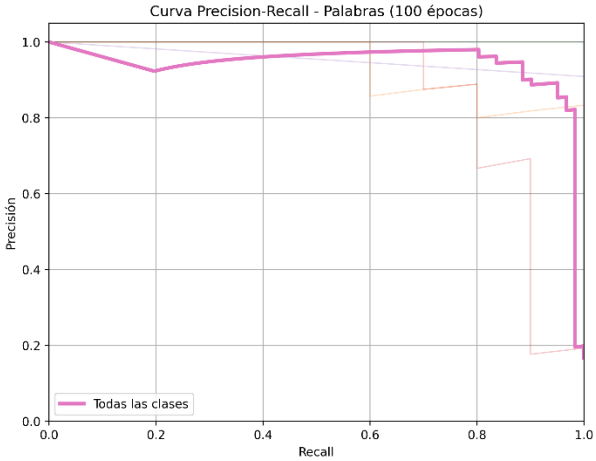
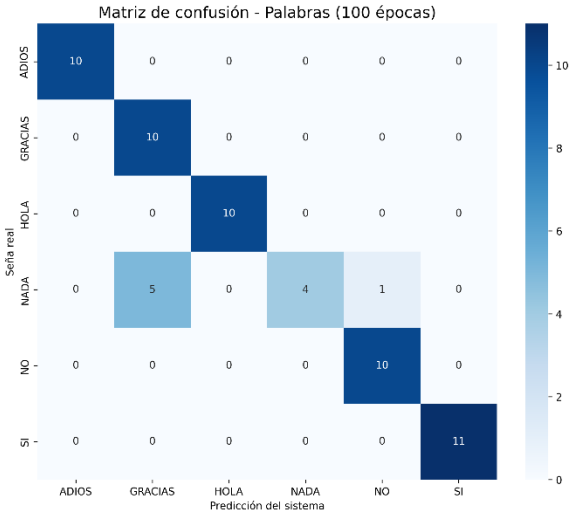




Análisis de Entrenamiento (Palabras): 100 Épocas (El Punto de Quiebre)

Tabla E13. *Rendimiento del modelo a 100 épocas*

Métrica	Análisis del resultado	Gráfica
Exactitud	Salto evolutivo monumental, alcanzando el 90.1%. El modelo finalmente logra estructurar matemáticamente el movimiento de las palabras.	
Pérdida	La pérdida de entrenamiento cae por fin a 0.18, aunque la validación (1.49) denota que el sistema aún necesita más entrenamiento para ser consistente.	
Precisión-confianza	Las fluctuaciones agresivas desaparecen. El modelo gana seguridad en la mayoría de sus umbrales de predicción.	

F1-confianza	Equilibrio robusto del 89% global. Palabras como "ADIOS" y "HOLA" logran puntuaciones perfectas (1.00).	
Precisión-recall	Recuperación total de la curva, logrando una clasificación mucho más limpia y balanceada.	
Matriz de confusión	La diagonal principal se define claramente. Solo la palabra "NADA" persiste con un rendimiento bajo (F1 de 0.57), justificando extender las épocas.	

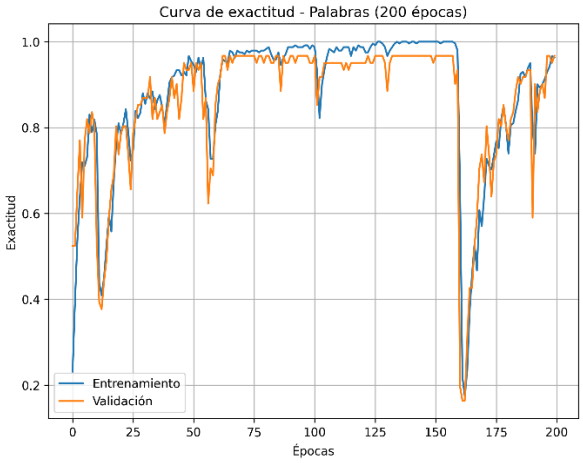
Análisis de Entrenamiento (Palabras): 200 Épocas (Convergencia Óptima)

Tabla E14. *Rendimiento del modelo a 200 épocas*

Métrica	Análisis del resultado	Gráfica

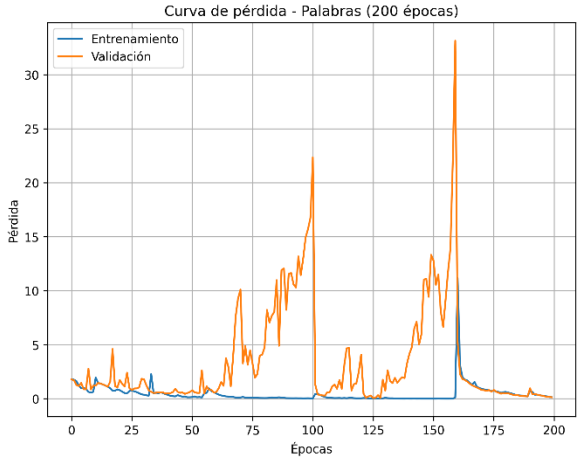
Exactitud

Alcanza su máxima eficiencia (96.7%). A diferencia de las señas estáticas (números/letras), los gestos dinámicos requieren obligatoriamente 200 ciclos para estabilizarse.



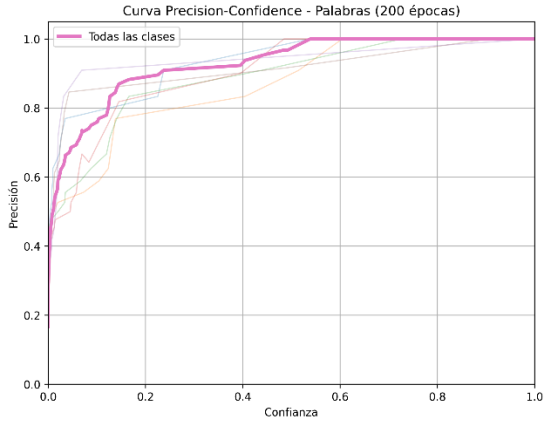
Pérdida

Estabilidad perfecta. Ambas pérdidas (entrenamiento 0.14, validación 0.13) convergen sin mostrar el mínimo indicio de sobreajuste (*overfitting*).



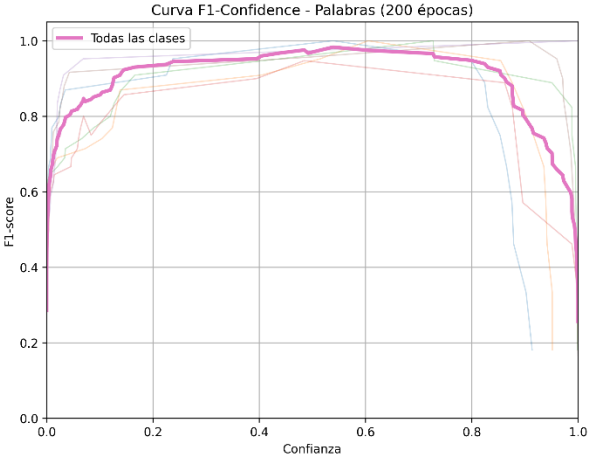
Precisión-confianza

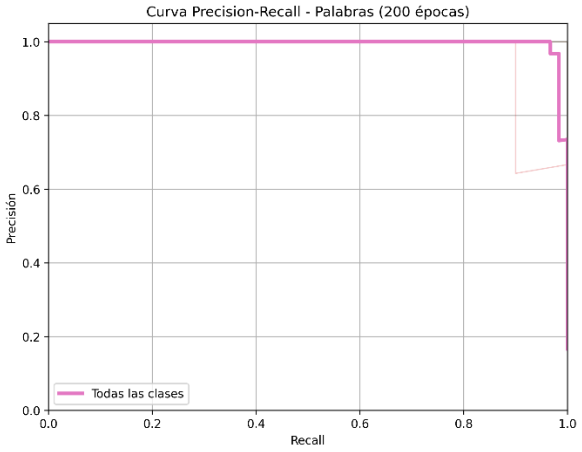
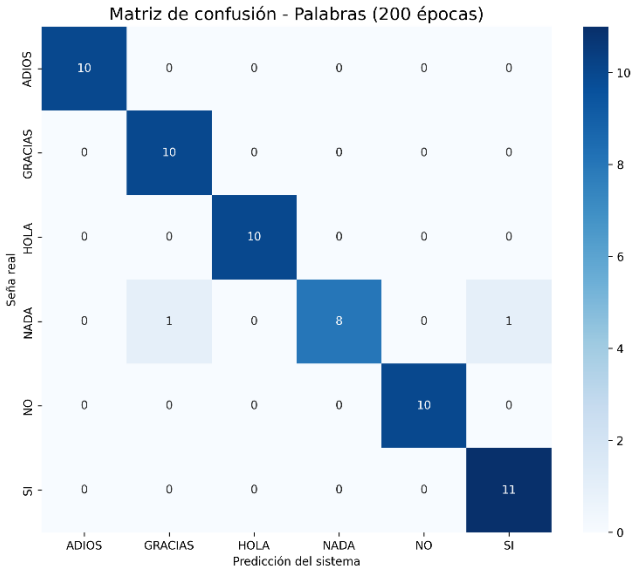
Confianza sólida en la predicción de secuencias complejas. El algoritmo es apto para entornos de producción.



F1-confianza

El sistema logra un espectacular 97% de balance. La rebelde palabra "NADA" finalmente es aprendida (F1 de 0.89).



Precisión-recall	Margen operativo maximizado, asegurando que el modelo distingue trayectorias de movimiento en tiempo real sin equivocarse.	
Matriz de confusión	Se confirma que 200 épocas es el hiperparámetro ideal para el dataset de palabras.	

Este modelo presentó los cambios más evidentes. La exactitud aumentó de 57,38 % con 5 épocas a 96,72 % con 200 épocas, mientras que el valor F1 pasó de 53,75 % a 96,61 %. La matriz de confusión también mostró una disminución de 26 a 2 errores. Por lo tanto, se seleccionó el modelo entrenado con 200 épocas.

Anexo F: Consentimiento Informado a Padres de Familia

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA ENCIENCIAS APLICADAS

CARRERA DE TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA: Dispositivo lúdico para reconocimiento de léxico dactilológico para niños de 6 a 8 años

AUTOR: Chamorro Lima Alfonsina Sheraseth

CONSENTIMIENTO INFORMADO Y AUTORIZACIÓN GENERAL DE USO DE IMAGEN PARA MENORES DE EDAD

1. INFORMACIÓN SOBRE EL ESTUDIO Por medio de este documento se le invita a participar en la investigación arriba mencionada. El objetivo de este estudio es evaluar el funcionamiento de un dispositivo lúdico interactivo diseñado para el reconocimiento de letras y palabras en lengua de señas. La participación del menor consistirá en interactuar con el dispositivo realizando gestos manuales dactilológicos frente a un sensor/cámara para que el sistema reconozca sus movimientos. El tiempo estimado de participación es de aproximadamente 15 a 30 minutos.

2. VOLUNTARIEDAD Y RETIRO La participación de su hijo(a) o representado(a) es completamente voluntaria. Usted tiene el derecho de negar su participación o de retirar al menor en cualquier momento del estudio, sin que esto le genere ninguna consecuencia negativa ni penalización.

3. CONFIDENCIALIDAD Y MANEJO DE DATOS Toda la información recolectada será estrictamente confidencial. Los datos personales del menor serán anonimizados utilizando códigos o pseudónimos. Los resultados se presentarán de forma agregada o anónima en el reporte final de la tesis de Ingeniería en Telecomunicaciones y en posibles publicaciones académicas derivadas.

4. AUTORIZACIÓN DE USO DE IMAGEN (Fotografía / Video) Para fines exclusivos del registro de la investigación y como evidencia científica en el cuerpo de la tesis o su defensa, se solicita su autorización para capturar la imagen del menor. Las imágenes o videos recolectados serán almacenados de forma segura y serán destruidos una vez finalizada y aprobada la tesis. *(En la tabla de firmas, por favor indique el NÚMERO de la opción que usted autoriza):*

- **Opción 1:** SÍ autorizo la toma de fotografías/videos para análisis interno Y SÍ autorizo que aparezcan en el documento de tesis final, **siempre y cuando se difumine, cubra o pixele el rostro** del menor para proteger su identidad.
- **Opción 2:** SÍ autorizo la toma de fotografías/videos **SOLO** para el análisis interno del algoritmo de la investigación (NO aparecerán en el documento final de la tesis).
- **Opción 3:** NO autorizo el uso de la imagen (rostro o cuerpo) del menor bajo ninguna circunstancia.

5. BENEFICIOS Y RIESGOS La participación no representa ningún riesgo físico ni psicológico para el menor. No se ofrece ninguna compensación económica por participar. El beneficio principal es contribuir al desarrollo del conocimiento científico, la tecnología inclusiva y la educación en el área de las telecomunicaciones aplicadas.

6. DECLARACIÓN DE CONSENTIMIENTO Y FIRMAS Los suscritos, en calidad de representantes legales de los menores indicados en la tabla inferior, declaramos que hemos leído (o se nos ha leído) y comprendido en su totalidad este documento. Se nos han aclarado todas las dudas de manera satisfactoria. Por lo tanto, otorgamos de manera libre y consciente nuestro consentimiento para que nuestros representados participen en esta investigación y aceptamos las condiciones de uso de imagen seleccionadas.

Lugar y fecha: Ibarra, 15 de junio de 2026.

Nombres y Apellidos del Representado	Nombres y Apellidos del Representante Legal	Cédula del Representante	Opción de Imagen (Escribir 1, 2 o 3)	Firma del Representante
	Elvia Amaguaña	105010916-2	1	
	Maria Ibaza	100328585 4	1	
	Ana Lucia Pasco	1003343827	1	
	Jesica Batallas.	1004774541-8	1	Jesica Batallas
	Cinthya Chinchuza	100165041-0	1	

Firma de la responsable del Proyecto

Autor: Alfonsina Sheraseth Chamorro Lima

C.I.:100408800-9

Anexo G: Evidencias detalladas de la validación con los niños participantes
Objetivo.

- Presentar los resultados individuales por categoría y seña que respaldan la validación expuesta en el apartado 4.4 del Capítulo IV.

Los resultados de este anexo se obtuvieron de la tabla registro_practicas de la base de datos SQLite. Antes del análisis se excluyeron los 22 registros correspondientes al usuario PRUEBA, por lo que las tablas contienen únicamente las actividades realizadas por los cinco niños participantes.

Para cada seña se presenta el número de actividades, los registros con estado EXITO, los saltos, el porcentaje de actividades completadas y el tiempo promedio. Este porcentaje no corresponde a la exactitud de los modelos durante el entrenamiento, sino al resultado final de las actividades realizadas con el sistema integrado.

G.1 Resultados detallados por participante y seña

Las siguientes tablas conservan el detalle de las señas practicadas por cada participante. Las etiquetas se presentan tal como fueron almacenadas en la base de datos.

G.1.1 Estudiante 1

Estudiante 1 corresponde a un participante oyente de 6 años. La Tabla G.1 presenta los resultados agrupados por seña.

Tabla G.1
Resultados detallados de Estudiante 1 por categoría y seña

Categoría	Seña	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y Vocales	A	13	12	1	92,31%	23,95 s
	E	9	9	0	100,00%	18,99 s
	I	7	7	0	100,00%	7,81 s
	O	7	7	0	100,00%	8,44 s
	U	7	7	0	100,00%	7,27 s
Subtotal	**—**	**43**	**42**	**1**	**97,67%**	**15,05 s**
Números	0	2	2	0	100,00%	7,05 s
	1	7	7	0	100,00%	33,14 s
	2	5	4	1	80,00%	12,90 s

	3	5	5	0	100,00%	9,50 s
	4	5	5	0	100,00%	6,34 s
	5	5	5	0	100,00%	6,26 s
	6	2	2	0	100,00%	5,65 s
	7	2	2	0	100,00%	11,70 s
	8	2	2	0	100,00%	12,35 s
	9	2	2	0	100,00%	14,50 s
Subtotal	**—**	**37**	**36**	**1**	**97,30%**	**13,77 s**
Gestos Dinámicos	ADIOS	2	1	1	50,00%	29,50 s
	HOLA	1	0	1	0,00%	17,10 s
	NO	3	2	1	66,67%	21,03 s
	SI	3	3	0	100,00%	18,70 s
Subtotal	**—**	**9**	**6**	**3**	**66,67%**	**21,70 s**
TOTAL	**—**	**89**	**84**	**5**	**94,38%**	**15,19 s**

Nota. Resultados obtenidos de los registros válidos almacenados en SQLite. Las etiquetas de las señas se conservan como aparecen en la base de datos. Elaboración propia, 2026.

El Estudiante 1 completó todas las actividades correspondientes a las vocales A, E, I, O y U. En números se registró un salto en el número 2, mientras que en los gestos dinámicos las señas ADIOS, HOLA y NO presentaron los porcentajes más bajos. La seña SI fue completada en las tres actividades registradas.

G.1.2 Estudiante 2

Estudiante 2 corresponde a un participante oyente de 8 años. La Tabla G.2 presenta los resultados agrupados por seña.

Tabla G.2
Resultados detallados de Estudiante 2 por categoría y seña

Categoría	Seña	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y Vocales	A	10	9	1	90,00%	7,99 s
	B	1	1	0	100,00%	6,60 s
	C	1	1	0	100,00%	6,60 s
	CH	1	1	0	100,00%	8,50 s
	D	2	2	0	100,00%	8,65 s
	E	6	5	1	83,33%	7,10 s
	F	1	1	0	100,00%	7,30 s
	G	1	1	0	100,00%	8,80 s
	H	1	1	0	100,00%	9,60 s
	I	3	3	0	100,00%	7,73 s
	J	2	2	0	100,00%	7,05 s
	K	1	1	0	100,00%	9,20 s
	L	1	1	0	100,00%	6,50 s
	M	1	1	0	100,00%	5,70 s

	N	1	1	0	100,00%	5,80 s
	O	3	3	0	100,00%	6,57 s
	P	1	1	0	100,00%	9,10 s
	Q	1	1	0	100,00%	9,40 s
	R	1	1	0	100,00%	6,10 s
	S	1	1	0	100,00%	7,50 s
	T	2	2	0	100,00%	14,70 s
	U	2	2	0	100,00%	8,00 s
	V	1	1	0	100,00%	6,50 s
	W	1	1	0	100,00%	6,60 s
	X	1	1	0	100,00%	7,30 s
	Y	1	1	0	100,00%	7,00 s
	Z	4	3	1	75,00%	8,55 s
Subtotal	**—**	**52**	**49**	**3**	**94,23%**	**7,89 s**
Números	0	4	3	1	75,00%	7,72 s
	1	5	4	1	80,00%	6,20 s
	2	7	5	2	71,43%	7,46 s
	3	5	3	2	60,00%	6,36 s
	4	4	3	1	75,00%	6,47 s
	5	4	3	1	75,00%	7,47 s
	6	1	1	0	100,00%	7,10 s
	7	1	1	0	100,00%	7,20 s
	8	1	1	0	100,00%	6,50 s
	9	1	1	0	100,00%	7,00 s
Subtotal	**—**	**33**	**25**	**8**	**75,76%**	**6,95 s**
Gestos Dinámicos	ADIOS	6	5	1	83,33%	16,03 s
	HOLA	5	3	2	60,00%	12,40 s
	NO	8	4	4	50,00%	12,30 s
	SI	8	7	1	87,50%	14,34 s
Subtotal	**—**	**27**	**19**	**8**	**70,37%**	**13,75 s**
TOTAL	**—**	**112**	**93**	**19**	**83,04%**	**9,03 s**

Nota. Resultados obtenidos de los registros válidos almacenados en SQLite. Las etiquetas de las señas se conservan como aparecen en la base de datos. Elaboración propia, 2026.

El Estudiante 2 presentó resultados altos en la mayoría de las letras. Los saltos se concentraron principalmente en los números del 0 al 5 y en los gestos dinámicos. Las señas HOLA y NO alcanzaron los porcentajes más bajos dentro de esta última categoría.

G.1.3 Estudiante 3

Estudiante 3 corresponde a un participante no oyente de 7 años. La Tabla G.3 presenta los resultados agrupados por seña.

Tabla G.3
Resultados detallados de Estudiante 3 por categoría y seña

Categoría	Seña	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y Vocales	A	2	2	0	100,00%	6,35 s
	B	3	3	0	100,00%	5,77 s
	C	3	3	0	100,00%	11,40 s
	CH	4	4	0	100,00%	8,20 s
	D	4	4	0	100,00%	6,12 s
	E	3	3	0	100,00%	7,17 s
	F	3	3	0	100,00%	6,27 s
	G	3	3	0	100,00%	6,50 s
	H	3	3	0	100,00%	7,17 s
	I	2	2	0	100,00%	5,95 s
	J	1	1	0	100,00%	8,10 s
	K	2	2	0	100,00%	6,50 s
	L	3	3	0	100,00%	6,50 s
	M	2	2	0	100,00%	5,55 s
	N	1	1	0	100,00%	5,50 s
	O	2	2	0	100,00%	6,25 s
	P	2	1	1	50,00%	6,50 s
	Q	1	1	0	100,00%	10,20 s
	R	1	1	0	100,00%	6,70 s
	S	2	2	0	100,00%	6,95 s
	T	2	2	0	100,00%	8,35 s
	U	1	1	0	100,00%	6,40 s
	V	2	2	0	100,00%	6,05 s
	W	1	1	0	100,00%	6,30 s
	X	2	2	0	100,00%	6,80 s
	Y	2	2	0	100,00%	6,30 s
	Z	1	1	0	100,00%	9,40 s
Subtotal	**—**	**58**	**57**	**1**	**98,28%**	**6,99 s**
Números	0	1	1	0	100,00%	5,70 s
	1	1	1	0	100,00%	5,60 s
	2	1	1	0	100,00%	6,50 s
	3	1	1	0	100,00%	6,90 s
	4	1	1	0	100,00%	6,30 s
	5	1	1	0	100,00%	6,40 s
	6	2	2	0	100,00%	5,65 s
	7	2	2	0	100,00%	6,30 s
	8	2	2	0	100,00%	5,65 s
	9	2	2	0	100,00%	6,60 s
Subtotal	**—**	**14**	**14**	**0**	**100,00%**	**6,13 s**
Gestos Dinámicos	ADIOS	2	2	0	100,00%	16,60 s
	NO	3	3	0	100,00%	12,70 s
	SI	3	3	0	100,00%	16,87 s
Subtotal	**—**	**80**	**79**	**1**	**98,75%**	**7,66 s**
TOTAL	**—**	**112**	**93**	**19**	**83,04%**	**9,03 s**

Nota. Resultados obtenidos de los registros válidos almacenados en SQLite. Las etiquetas de las señas se conservan como aparecen en la base de datos. Elaboración propia, 2026.

El Estudiante 3 completó todas las actividades de números y gestos dinámicos registradas. En el abecedario se presentó un solo salto en la letra P. Los demás signos practicados fueron finalizados con estado EXITO.

G.1.4 Estudiante 4

Estudiante 4 corresponde a un participante no oyente de 8 años. La Tabla G.4 presenta los resultados agrupados por seña.

Tabla G.4
Resultados detallados de Estudiante 4 por categoría y seña

Categoría	Seña	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y Vocales	A	2	2	0	100,00%	7,50 s
	B	1	1	0	100,00%	5,60 s
	CH	1	1	0	100,00%	9,10 s
	E	2	2	0	100,00%	7,95 s
	F	1	1	0	100,00%	6,30 s
	G	1	1	0	100,00%	9,10 s
	H	1	1	0	100,00%	6,00 s
	I	2	2	0	100,00%	5,85 s
	J	1	1	0	100,00%	7,20 s
	K	1	0	1	0,00%	0,20 s
	L	1	1	0	100,00%	5,60 s
	N	2	1	1	50,00%	11,25 s
	O	2	2	0	100,00%	6,05 s
	P	1	0	1	0,00%	13,30 s
	Q	1	0	1	0,00%	5,10 s
	R	1	0	1	0,00%	15,00 s
	S	1	1	0	100,00%	7,10 s
	T	1	1	0	100,00%	6,40 s
	U	2	2	0	100,00%	5,90 s
	V	1	1	0	100,00%	14,10 s
	W	1	1	0	100,00%	5,60 s
	X	1	1	0	100,00%	8,20 s
	Y	1	1	0	100,00%	6,00 s
	Z	2	1	1	50,00%	13,20 s
Subtotal	**—**	**31**	**25**	**6**	**80,65%**	**7,91 s**
Números	0	2	2	0	100,00%	5,70 s
	1	3	3	0	100,00%	6,70 s
	2	3	3	0	100,00%	6,60 s
	3	3	3	0	100,00%	6,23 s

	4	3	3	0	100,00%	6,53 s
	5	3	3	0	100,00%	6,73 s
	6	2	2	0	100,00%	5,95 s
	7	2	2	0	100,00%	6,75 s
	8	2	2	0	100,00%	6,15 s
	9	2	2	0	100,00%	6,00 s
Subtotal	**—**	**25**	**25**	**0**	**100,00%**	**6,38 s**
	ADIOS	2	1	1	50,00%	16,45 s
Gestos Dinámicos	NO	3	3	0	100,00%	15,27 s
	SI	3	3	0	100,00%	12,33 s
Subtotal	**—**	**8**	**7**	**1**	**87,50%**	**14,46 s**
TOTAL	**—**	**64**	**57**	**7**	**89,06%**	**8,13 s**

Nota. Resultados obtenidos de los registros válidos almacenados en SQLite. Las etiquetas de las señas se conservan como aparecen en la base de datos. Elaboración propia, 2026.

El Estudiante 4 completó todas las actividades de números. En el abecedario se registraron saltos en las letras K, N, P, Q, R y Z. En los gestos dinámicos, ADIOS presentó un éxito y un salto, mientras que NO y SI fueron completados en todas las actividades realizadas.

G.1.5 Estudiante 5

Estudiante 5 corresponde a un participante no oyente de 8 años. La Tabla G.5 presenta los resultados agrupados por seña.

Tabla G.5
Resultados detallados de Estudiante 5 por categoría y seña

Categoría	Seña	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
	A	3	3	0	100,00%	9,50 s
	B	1	1	0	100,00%	7,90 s
	F	3	3	0	100,00%	9,43 s
Abecedario y	I	3	3	0	100,00%	6,10 s
Vocales	K	1	1	0	100,00%	8,90 s
	L	1	1	0	100,00%	6,10 s
	O	3	3	0	100,00%	6,97 s
	S	3	3	0	100,00%	6,60 s
Subtotal	**—**	**18**	**18**	**0**	**100,00%**	**7,71 s**
	0	3	3	0	100,00%	6,70 s
	1	3	3	0	100,00%	7,43 s
	2	3	3	0	100,00%	6,43 s
	3	3	3	0	100,00%	5,77 s
	4	3	3	0	100,00%	5,67 s
Números	5	3	3	0	100,00%	7,17 s
	6	3	3	0	100,00%	5,60 s
	7	3	3	0	100,00%	7,27 s
	8	3	3	0	100,00%	5,93 s
	9	3	3	0	100,00%	6,80 s

Subtotal	**—**	**30**	**30**	**0**	**100,00%**	**6,48 s**
Gestos Dinámicos	ADIOS	3	2	1	66,67%	14,87 s
	NO	3	3	0	100,00%	14,27 s
	SI	3	3	0	100,00%	14,90 s
Subtotal	**—**	**9**	**8**	**1**	**88,89%**	**14,68 s**
TOTAL	**—**	**57**	**56**	**1**	**98,25%**	**8,16 s**

Nota. Resultados obtenidos de los registros válidos almacenados en SQLite. Las etiquetas de las señas se conservan como aparecen en la base de datos. Elaboración propia, 2026.

El Estudiante 5 completó todas las actividades registradas de abecedario y números. El único salto se presentó en el gesto dinámico ADIOS. Las señas NO y SI fueron completadas en las tres actividades realizadas para cada una.

G.2 Matriz consolidada por seña y grupo evaluado

Para facilitar la revisión de los resultados, la matriz consolidada se dividió según la condición de los participantes. La Tabla G.6 presenta las actividades realizadas por el grupo oyente, mientras que la Tabla G.7 muestra los resultados del grupo no oyente. En ambas tablas se conserva la categoría, la seña solicitada, la cantidad de actividades, los éxitos, los saltos, el porcentaje de actividades completadas y el tiempo promedio.

G.2.1 Resultados del grupo oyente

El grupo oyente estuvo conformado por dos participantes de 6 años. La Tabla G.6 reúne los resultados de sus 201 actividades válidas y permite identificar las señas que presentaron una mayor cantidad de saltos.

Tabla G.6

Matriz consolidada por seña del grupo oyente

Categoría	Seña	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y Vocales	A	23	21	2	91,30%	17,01 s
	B	1	1	0	100,00%	6,60 s
	C	1	1	0	100,00%	6,60 s
	CH	1	1	0	100,00%	8,50 s
	D	2	2	0	100,00%	8,65 s
	E	15	14	1	93,33%	14,23 s
	F	1	1	0	100,00%	7,30 s
	G	1	1	0	100,00%	8,80 s
	H	1	1	0	100,00%	9,60 s
	I	10	10	0	100,00%	7,79 s
	J	2	2	0	100,00%	7,05 s
	K	1	1	0	100,00%	9,20 s
	L	1	1	0	100,00%	6,50 s
	M	1	1	0	100,00%	5,70 s
	N	1	1	0	100,00%	5,80 s
	O	10	10	0	100,00%	7,88 s
	P	1	1	0	100,00%	9,10 s

	Q	1	1	0	100,00%	9,40 s
	R	1	1	0	100,00%	6,10 s
	S	1	1	0	100,00%	7,50 s
	T	2	2	0	100,00%	14,70 s
	U	9	9	0	100,00%	7,43 s
	V	1	1	0	100,00%	6,50 s
	W	1	1	0	100,00%	6,60 s
	X	1	1	0	100,00%	7,30 s
	Y	1	1	0	100,00%	7,00 s
	Z	4	3	1	75,00%	8,55 s
Subtotal	**—**	**95**	**91**	**4**	**95,79%**	**11,13 s**
Números	0	6	5	1	83,33%	7,50 s
	1	12	11	1	91,67%	21,92 s
	2	12	9	3	75,00%	9,72 s
	3	10	8	2	80,00%	7,93 s
	4	9	8	1	88,89%	6,40 s
	5	9	8	1	88,89%	6,80 s
	6	3	3	0	100,00%	6,13 s
	7	3	3	0	100,00%	10,20 s
	8	3	3	0	100,00%	10,40 s
	9	3	3	0	100,00%	12,00 s
Subtotal	**—**	**70**	**61**	**9**	**87,14%**	**10,56 s**
Gestos Dinámicos	ADIOS	8	6	2	75,00%	19,40 s
	HOLA	6	3	3	50,00%	13,18 s
	NO	11	6	5	54,55%	14,68 s
	SI	11	10	1	90,91%	15,53 s
Subtotal	**—**	**36**	**25**	**11**	**69,44%**	**15,74 s**
TOTAL	**—**	**201**	**177**	**24**	**88,06%**	**11,76 s**

Nota. Resultados obtenidos de las actividades válidas realizadas por los participantes oyentes. El porcentaje representa las actividades finalizadas con estado EXITO respecto al total registrado para cada seña. Elaboración propia, 2026.

En el abecedario, los saltos se concentraron en las letras A, E y Z. Dentro de los números, el 2 presentó tres saltos, seguido del 3 con dos. En los gestos dinámicos, HOLA y NO registraron los porcentajes más bajos, con 50,00 % y 54,55 %, respectivamente, mientras que SI alcanzó 90,91%.

G.2.2 Resultados del grupo no oyente

El grupo no oyente estuvo conformado por tres participantes de 7 a 8 años. La Tabla G.7 presenta los resultados de las 201 actividades válidas registradas para este grupo.

Tabla G.7
Matriz consolidada por seña del grupo no oyente

Categoría	Seña	Actividades	Éxitos	Saltos	Completadas	Tiempo promedio
Abecedario y Vocales	A	7	7	0	100,00%	8,03 s
	B	5	5	0	100,00%	6,16 s
	C	3	3	0	100,00%	11,40 s
	CH	5	5	0	100,00%	8,38 s
	D	4	4	0	100,00%	6,12 s
	E	5	5	0	100,00%	7,48 s
	F	7	7	0	100,00%	7,63 s
	G	4	4	0	100,00%	7,15 s

	H	4	4	0	100,00%	6,88 s
	I	7	7	0	100,00%	5,99 s
	J	2	2	0	100,00%	7,65 s
	K	4	3	1	75,00%	5,53 s
	L	5	5	0	100,00%	6,24 s
	M	2	2	0	100,00%	5,55 s
	N	3	2	1	66,67%	9,33 s
	O	7	7	0	100,00%	6,50 s
	P	3	1	2	33,33%	8,77 s
	Q	2	1	1	50,00%	7,65 s
	R	2	1	1	50,00%	10,85 s
	S	6	6	0	100,00%	6,80 s
	T	3	3	0	100,00%	7,70 s
	U	3	3	0	100,00%	6,07 s
	V	3	3	0	100,00%	8,73 s
	W	2	2	0	100,00%	5,95 s
	X	3	3	0	100,00%	7,27 s
	Y	3	3	0	100,00%	6,20 s
	Z	3	2	1	66,67%	11,93 s
Subtotal	**—**	**107**	**100**	**7**	**93,46%**	**7,38 s**
Números	0	6	6	0	100,00%	6,20 s
	1	7	7	0	100,00%	6,86 s
	2	7	7	0	100,00%	6,51 s
	3	7	7	0	100,00%	6,13 s
	4	7	7	0	100,00%	6,13 s
	5	7	7	0	100,00%	6,87 s
	6	7	7	0	100,00%	5,71 s
	7	7	7	0	100,00%	6,84 s
	8	7	7	0	100,00%	5,91 s
	9	7	7	0	100,00%	6,51 s
Subtotal	**—**	**69**	**69**	**0**	**100,00%**	**6,37 s**
Gestos Dinámicos	ADIOS	7	5	2	71,43%	15,81 s
	NO	9	9	0	100,00%	14,08 s
	SI	9	9	0	100,00%	14,70 s
Subtotal	**—**	**25**	**23**	**2**	**92,00%**	**14,79 s**
TOTAL	**—**	**201**	**192**	**9**	**95,52%**	**7,95 s**

Nota. Resultados obtenidos de las actividades válidas realizadas por los participantes no oyentes. El porcentaje representa las actividades finalizadas con estado EXITO respecto al total registrado para cada seña. La seña HOLA no aparece porque no se registraron actividades válidas de esta clase para el grupo. Elaboración propia, 2026.

El grupo no oyente completó todas las actividades correspondientes a los números. En el abecedario se registraron saltos en K, N, P, Q, R y Z, con el porcentaje más bajo en la letra P. En los gestos dinámicos, ADIOS presentó dos saltos, mientras que NO y SI fueron completados en todas las actividades registradas.

G.3 Comparación detallada entre los grupos

Los resultados también se organizaron según la condición de los participantes. El grupo oyente estuvo conformado por dos niños y el grupo no oyente por tres; sin embargo, cada grupo acumuló 201 actividades válidas. La Tabla G.8 presenta la comparación por categoría.

Tabla G.8

Comparación de resultados entre participantes oyentes y no oyentes

Categoría	Actividad	Éxitos	Saltos	Completadas	Tiempo	Actividad	Éxitos	Saltos	Completadas	Tiempo
	OYENTE					NO OYENTE				
Abecedario y Vocales	95	91	4	95,79%	11,13 s	107	100	7	93,46%	7,38 s
Números	70	61	9	87,14%	10,56 s	69	69	0	100,00%	6,37 s
Gestos Dinámicos	36	25	11	69,44%	15,74 s	25	23	2	92,00%	14,79 s
TOTAL	**201**	**177**	**24**	**88,06%**	**11,76 s**	**201**	**192**	**9**	**95,52%**	**7,95 s**

Nota. Elaboración propia, 2026.

En el resultado total, el grupo no oyente completó 192 de las 201 actividades, equivalente al 95,52 %, con un tiempo promedio de 7,95 segundos. El grupo oyente completó 177 actividades, correspondiente al 88,06 %, y registró un tiempo promedio de 11,76 segundos. La diferencia entre los grupos fue de 7,46 puntos porcentuales y 3,81 segundos en el tiempo promedio.

Por categoría, el grupo no oyente completó todas las actividades de números y alcanzó 92,00 % en los gestos dinámicos. En el grupo oyente estos valores fueron 87,14 % y 69,44 %, respectivamente. En el abecedario, el grupo oyente presentó un porcentaje ligeramente mayor, con 95,79 % frente a 93,46 %; no obstante, el grupo no oyente realizó estas actividades en un menor tiempo promedio: 7,38 segundos frente a 11,13 segundos.

Durante las sesiones se observó que los participantes no oyentes tenían mayor familiaridad previa con el abecedario dactilológico y con la formación de las señas. Esta experiencia puede ayudar a comprender el menor tiempo registrado, mientras que los participantes oyentes necesitaron más tiempo para observar y reproducir las posiciones de la mano.

Anexo H: Encuesta de Funcionamiento
Anexo H.1: Formato de encuesta de funcionamiento

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA ENCIENCIAS APLICADAS

CARRERA DE TELECOMUNICACIONES

ENCUESTA

TÍTULO:

Instrumento de Validación Operativa y Calidad del Sistema Basado en la Norma ISO/IEC 25010.

OBJETIVO:

Evaluar la idoneidad funcional, la usabilidad y la eficiencia de desempeño del dispositivo lúdico mediante la experiencia directa del docente tras las pruebas de funcionamiento en entorno real. La evaluación se fundamenta en los atributos del modelo de calidad de software ISO/IEC 25010, permitiendo validar el cumplimiento de los requerimientos técnicos y pedagógicos del sistema.

PROPÓSITO E INSTRUCCIONES:

Estimado/a docente: Tras haber finalizado las pruebas de interacción con el prototipo funcional, este instrumento busca recopilar su evaluación técnica y pedagógica sobre el comportamiento del sistema. Sus respuestas son vitales para validar la viabilidad del proyecto como herramienta de apoyo real en el aula. Por favor, responda con total sinceridad, evaluando del 1 al 5 cuando se le solicite, o desarrollando su respuesta en las preguntas abiertas. Los datos serán tratados con absoluta confidencialidad para fines académicos.

Bloque 1: Utilidad de la herramienta en el aprendizaje

- **Pregunta 1:** Del 1 al 5, ¿qué tan útil le parece la función de "Signos a Reforzar" para identificar automáticamente qué señas necesitan más práctica teórica?

- **Pregunta 2:** Basado en su experiencia como educadora, ¿qué funcionalidad adicional le agregaría al sistema, o qué detalle cambiaría para que la herramienta sea perfecta en su día a día?

Bloque 2: Facilidad de uso y motivación del niño

- **Pregunta 3:** Del 1 al 5, ¿qué tan intuitivo y fácil de manejar le resultó el panel de control desde su teléfono móvil?

- **Pregunta 4:** ¿Considera que los colores (verde para aciertos, rojo suave para fallos) y las imágenes de apoyo motivan al niño o le generan algún grado de frustración?

- **Pregunta 5:** ¿Cómo calificaría el nivel de motivación e interés del niño al interactuar con el sistema en comparación con los métodos de enseñanza tradicionales?

Bloque 3: Velocidad y respuesta del equipo

- **Pregunta 6:** ¿Considera que las opciones de "Pausar", "Saltar Signo" y "Finalizar Sesión" responden con la rapidez necesaria para mantener el control del aula?

- **Pregunta 7:** ¿El tiempo que la Inteligencia Artificial espera a que el niño forme la seña antes de marcar un error le parece adecuado para su nivel de desarrollo motriz?

Bloque 4: Funcionamiento físico y seguridad en el aula

- **Pregunta 8:** Si la escuela no contara con acceso a internet, ¿considera una ventaja significativa que el sistema funcione mediante una red Wi-Fi local autónoma?

- **Pregunta 9:** ¿Considera que la disposición física del equipo (cámara, placa, cables y pantalla) es segura y cómoda para los niños?



Estudiante
Srta. Alfonsina Chamorro



Director
MSc. Jaime Michilena

Anexo H.2: Encuesta de funcionamiento realizada a docente

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA ENCIENCIAS APLICADAS

CARRERA DE TELECOMUNICACIONES

ENCUESTA

- **Pregunta 1:** Del 1 al 5, ¿qué tan intuitivo y fácil de manejar le resultó el panel de control desde su teléfono móvil?

  5

- **Pregunta 2:** ¿Considera que las opciones de "Pausar", "Saltar Signo" y "Finalizar Sesión" responden con la rapidez necesaria para mantener el control del aula?

  Si totalmente.

- **Pregunta 3:** Del 1 al 5, ¿qué tan útil le parece la función de "Signos a Reforzar" para identificar automáticamente qué señas necesitan más práctica teórica?

  2

- **Pregunta 4:** ¿Cómo calificaria el nivel de motivación e interés del niño al interactuar con el sistema en comparación con los métodos de enseñanza tradicionales?

  Muy Motivo.

- **Pregunta 5:** ¿El tiempo que la Inteligencia Artificial espera a que el niño forme la seña antes de marcar un error le parece adecuado para su nivel de desarrollo motriz?

  Si

- **Pregunta 6:** ¿Considera que los colores (verde para aciertos, rojo suave para fallos) y las imágenes de apoyo motivan al niño o le generan algún grado de frustración? Si fuera mejor colores mas llamativos y fuertes ya q' ellos se motivan y tambien mejoran puede ser ratitos ☺☺

- **Pregunta 7:** ¿Considera que la disposición física del equipo (cámara, placa, cables y pantalla) es segura y cómoda?

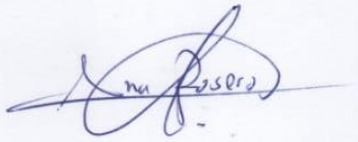
  Si 100%

- **Pregunta 8:** Si la escuela no contara con acceso a internet, ¿considera una ventaja significativa que el sistema funcione mediante una red Wi-Fi local autónoma?

  Si fuera de gran ayuda para los niños puedan mejorar sus conocimientos

- **Pregunta 9:** Basado en su experiencia como educadora, ¿qué funcionalidad adicional le agregaría al sistema, o qué detalle cambiaría para que sea una herramienta perfecta en su día a día?

Muy productiva ya q' se iría balanzando agregando un poco mas complejo, fuera ideal tener el acceso de incrementar mas temas con gráficos y ello puedan hacer su tarea y hacer por ejemplo una evaluación tecnológica para los niños sordos.



Anexo I: Manual de uso del sistema

MANUAL DE USUARIO

Sistema didáctico de reconocimiento de señas dactilológicas

1. Introducción

El presente manual de usuario describe el uso del sistema didáctico de reconocimiento de señas dactilológicas, desarrollado como una herramienta de apoyo para la práctica de letras, números y palabras seleccionadas mediante reconocimiento visual de gestos. El sistema está orientado al trabajo con niños de 6 a 8 años y permite que la docente configure sesiones de práctica, supervise resultados y consulte reportes de desempeño.

2. Objetivo

Orientar a la docente o usuario responsable en el uso correcto del prototipo, incluyendo el encendido del sistema, acceso a la interfaz del estudiante, uso del dashboard docente, ejecución de sesiones de práctica, consulta de resultados y generación de reportes.

3. Alcance

El manual cubre las funciones principales del sistema implementado: inicio del prototipo, selección de estudiante, selección de categoría de práctica, reconocimiento de señas, retroalimentación visual, consulta de resultados, revisión de historial y descarga de reportes.

4. Usuarios

Usuario	Descripción	Funciones principales
Estudiante	Niño o niña que realiza la práctica de señas frente al prototipo.	Observar la seña objetivo, realizar la seña y recibir retroalimentación visual.
Docente	Persona encargada de guiar la actividad y revisar los resultados.	Seleccionar estudiante, configurar práctica, consultar resultados y descargar reportes.

5. Requisitos previos de uso

Antes de utilizar el sistema, se debe verificar lo siguiente:

Requisito	Descripción
Alimentación eléctrica	El prototipo debe estar conectado a una fuente de alimentación de 5 V–4 A.
Cámara	La cámara IMX219 debe estar conectada correctamente a la Jetson Nano.
Pantalla	La pantalla debe estar conectada mediante HDMI y USB.

Red local	El prototipo y el dispositivo de la docente deben estar conectados a la misma red local.
Iluminación	El área de práctica debe contar con iluminación suficiente.
Ubicación del estudiante	El estudiante debe colocarse frente a la cámara para que la mano pueda ser detectada correctamente.

6. **Precauciones de uso**
   - Utilizar el prototipo bajo supervisión de la docente o responsable.
   - Evitar manipular cables internos mientras el sistema se encuentra encendido.
   - No cubrir la cámara durante la sesión de práctica.
   - Evitar golpes o movimientos bruscos sobre la carcasa.
   - Mantener el prototipo en una superficie estable.
   - Apagar el sistema correctamente al finalizar su uso.

7. **Encendido del prototipo**

Para iniciar el sistema, siga los siguientes pasos:

- Coloque el prototipo sobre una superficie estable.
- Verifique que la cámara, pantalla y cable de red estén conectados.
- Conecte la fuente de alimentación del prototipo.
- Espere a que la Jetson Nano inicie el sistema.
- Verifique que la pantalla muestre la interfaz principal.

Resultado esperado:
El prototipo debe iniciar correctamente y mostrar la interfaz visual del estudiante.



Acceso al dashboard docente

El dashboard docente es el panel web local que permite a la docente ingresar al sistema, seleccionar al estudiante, configurar la categoría de práctica, consultar resultados y generar reportes.

Procedimiento

1. Conecte el teléfono, tableta o computadora de la docente a la red local del sistema.
2. Abra el navegador web del dispositivo.

3. En la barra de direcciones, escriba la dirección IP local del sistema.
4. Presione la tecla **Enter** o el botón de búsqueda del navegador.
5. Espere a que se cargue la pantalla de inicio de sesión del dashboard.

Ejemplo de dirección local:

http://192.168.X.X:5000

La dirección IP puede variar según la configuración de la red local utilizada durante la sesión.

Resultado esperado:
Debe mostrarse la pantalla de inicio de sesión del dashboard docente.



Inicio de sesión del docente

Una vez cargada la pantalla de inicio de sesión, la docente debe ingresar sus credenciales de acceso. Estas credenciales permiten proteger el panel docente y evitar que usuarios no autorizados consulten o modifiquen la información registrada.

Para iniciar sesión, siga los siguientes pasos:

1. Ubique el campo **Usuario**.
2. Escriba el nombre de usuario asignado.
3. Ubique el campo **Contraseña**.
4. Escriba la contraseña correspondiente.
5. Presione el botón **Ingresar**.
6. Espere a que el sistema cargue la pantalla principal del dashboard.

Ejemplo de credenciales de prueba:

Campo	Ejemplo
Usuario	docente_demo
Contraseña	********

Nota: Las credenciales mostradas en este manual son únicamente referenciales. Las credenciales reales deben ser proporcionadas por el administrador o responsable del sistema.

Resultado esperado:
El sistema debe permitir el acceso al dashboard y mostrar la pantalla principal del panel docente.



Pantalla principal del dashboard

Luego de ingresar al sistema, se muestra el dashboard principal. Desde esta pantalla, la docente puede gestionar estudiantes, seleccionar actividades, iniciar prácticas, controlar la sesión, revisar reportes y apagar el sistema.

La pantalla principal está organizada en varias secciones: gestión de estudiantes, selección de abecedario, selección de números, selección de gestos dinámicos y controles generales de sesión



Registro y gestión de estudiantes

La sección **Estudiantes Registrados** permite administrar los niños que utilizarán el sistema durante las sesiones de práctica. Desde esta sección, la docente puede agregar un nuevo estudiante, seleccionar el estudiante activo, editar datos o eliminar registros.

Agregar nuevo estudiante

Para agregar un estudiante, realice los siguientes pasos:

1. Ubique la sección **Estudiantes Registrados**.
2. Escriba el nombre del estudiante en el campo **Nombre**.
3. Escriba la edad del estudiante en el campo **Edad**.
4. Presione el botón + **Agregar**.
5. Verifique que el estudiante aparezca en la lista.



Resultado esperado:
El nuevo estudiante debe mostrarse en la lista de estudiantes registrados.



Editar o eliminar estudiante

Los botones de edición y eliminación permiten modificar o retirar estudiantes registrados.

Botón	Función	Icono
Seleccionar	Activa al estudiante para iniciar una práctica.	✅
Editar	Permite modificar los datos del estudiante.	✏️
Eliminar	Permite borrar el estudiante de la lista.	🗑️

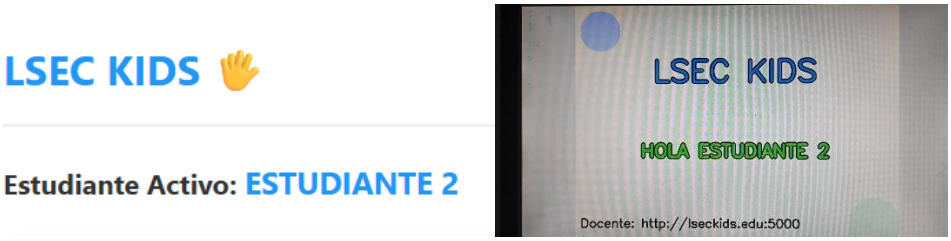
Seleccionar estudiante activo

Antes de iniciar una práctica, se debe seleccionar el estudiante que realizará la actividad.

Pasos básicos:

1. Ubique al estudiante en la lista.
2. Presione el botón de selección ✅ ubicado junto al nombre del estudiante.
3. Verifique que el nombre aparezca en el campo **Estudiante Activo**.

Resultado esperado:
El estudiante seleccionado debe aparecer como activo antes de iniciar la práctica.



Selección de letras para práctica del abecedario

La sección **Abecedario** permite seleccionar las letras que serán practicadas por el estudiante. La docente puede seleccionar letras individuales o utilizar las opciones **Todos** y **Ninguno**.



Pasos básicos:

1. Ubicar la sección **Abecedario**.
2. Seleccionar las letras que se desea practicar.
3. Usar **Todos** si se desea marcar todas las letras.

4. Usar **Ninguno** si se desea limpiar la selección.
5. Presionar **Iniciar Letras**.

Resultado esperado:
El sistema debe iniciar una práctica con las letras seleccionadas.



Selección de números

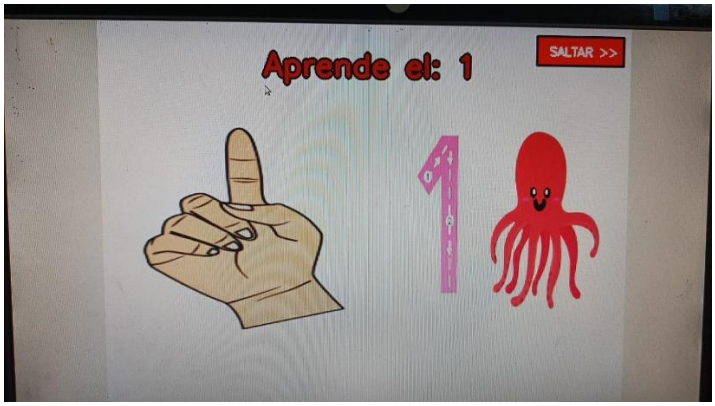
La sección **Números** permite seleccionar los números que serán practicados por el estudiante.



Pasos básicos:

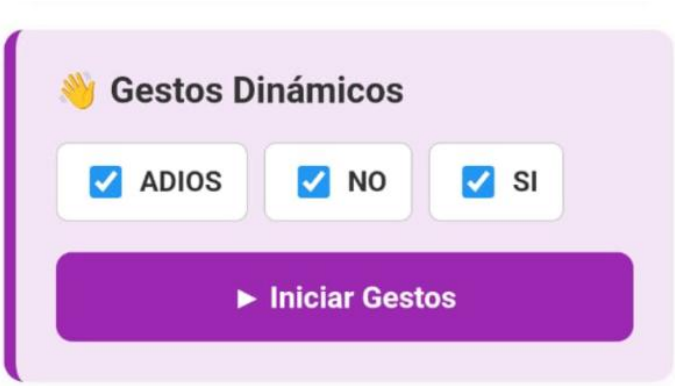
1. Ubicar la sección **Números**.
2. Seleccionar los números que se desea practicar.
3. Usar **Todos** para marcar todos los números.
4. Usar **Ninguno** para desmarcar la selección.
5. Presionar **Iniciar Números**.

Resultado esperado:
El sistema debe iniciar una práctica con los números seleccionados.



Selección de gestos dinámicos

La sección Gestos Dinámicos permite seleccionar palabras o gestos que requieren movimiento, como ADIÓS, NO o SÍ.



Pasos básicos:

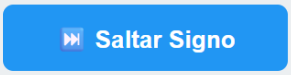
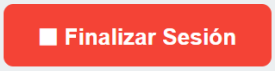
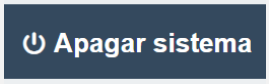
1. Ubicar la sección **Gestos Dinámicos**.
2. Seleccionar los gestos que se desea practicar.
3. Presionar el botón **Iniciar Gestos**.

Resultado esperado:
El sistema debe iniciar una práctica con los gestos dinámicos seleccionados.



Controles generales de sesión

En la parte inferior del dashboard se encuentran los controles generales de sesión. Estos permiten controlar la práctica una vez iniciada.

Botón	Función	Icono
Saltar Signo	Permite avanzar a la siguiente seña sin finalizar la sesión.	
Pausar / Reanudar	Permite detener temporalmente la práctica y continuarla después.	
Finalizar Sesión	Permite terminar la práctica activa.	
Ver Reportes de Estudiantes	Permite acceder a los resultados registrados.	
Apagar sistema	Permite apagar el prototipo de forma controlada.	

Pasos básicos:

1. Utilizar **Saltar Signo** si se desea pasar a otra seña.
2. Utilizar **Pausar / Reanudar** si se necesita detener temporalmente la práctica.
3. Presionar **Finalizar Sesión** al terminar la actividad.
4. Ingresar a **Ver Reportes de Estudiantes** para consultar resultados.
5. Utilizar **Apagar sistema** únicamente al finalizar el uso del prototipo.

Uso de la interfaz del estudiante

La interfaz del estudiante se muestra en la pantalla del prototipo. En ella se presenta la seña objetivo, una referencia visual y la retroalimentación generada por el sistema.

Pasos básicos:

1. El estudiante observa la seña objetivo en la pantalla.
2. La docente guía al estudiante si es necesario.
3. El estudiante realiza la seña frente a la cámara.
4. El sistema procesa la seña realizada.
5. La pantalla muestra una respuesta visual.

Resultado esperado en caso de que acierte la letra:
El estudiante debe recibir un incentivo visual después de realizar la seña.



Resultado esperado en caso de que no acierte la letra:
El estudiante debe recibir una retroalimentación visual clara después de realizar la seña.



Consulta de reportes de progreso

La opción Ver Reportes de Estudiantes permite a la docente revisar el avance de los estudiantes registrados en el sistema. Esta sección presenta una matriz de progreso donde se relacionan los estudiantes con las señas practicadas, utilizando indicadores visuales de color para facilitar la interpretación del desempeño.

Para ingresar a esta sección, la docente debe presionar el botón Ver Reportes de Estudiantes ubicado en la parte inferior del dashboard principal.



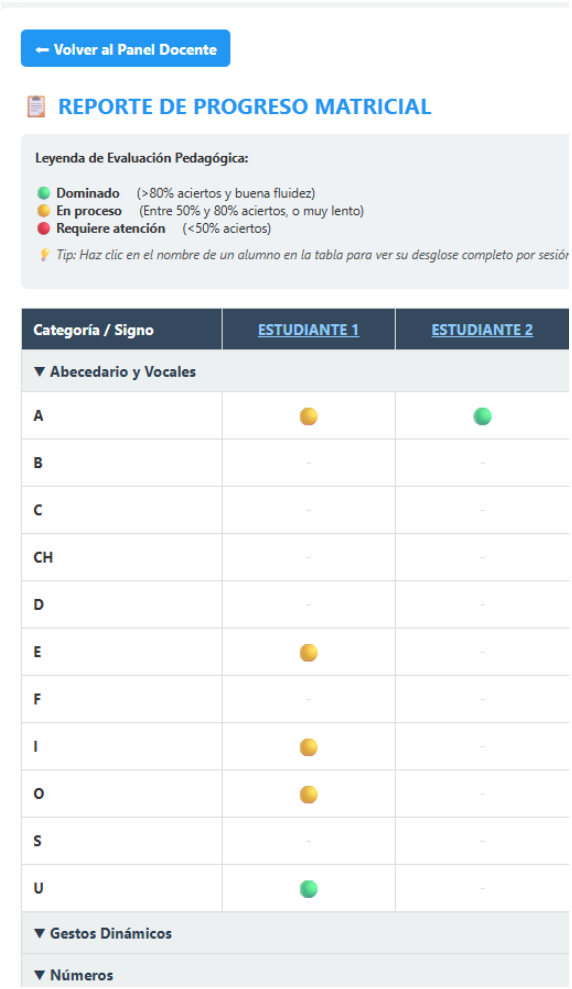
Reporte de progreso matricial

El **Reporte de Progreso Matricial** muestra el avance de los estudiantes por categoría y seña practicada. En la parte superior se presenta una leyenda pedagógica que permite interpretar el estado de aprendizaje mediante colores.

Indicador	Estado	Interpretación
Verde	Dominado	El estudiante presenta buen desempeño en la seña evaluada.
Amarillo	En proceso	El estudiante requiere más práctica o presenta lentitud en la ejecución.
Rojo	Requiere atención	El estudiante necesita refuerzo en la seña evaluada.

Para utilizar el reporte matricial, se deben seguir los siguientes pasos:

1. Ingresar a **Ver Reportes de Estudiantes**.
2. Revisar la leyenda de colores ubicada en la parte superior.
3. Identificar la categoría o seña en la columna izquierda.
4. Ubicar el nombre del estudiante en la parte superior de la tabla.
5. Observar el color mostrado en la intersección entre estudiante y seña.
6. Hacer clic sobre el nombre del estudiante o sobre el indicador para revisar el detalle correspondiente.

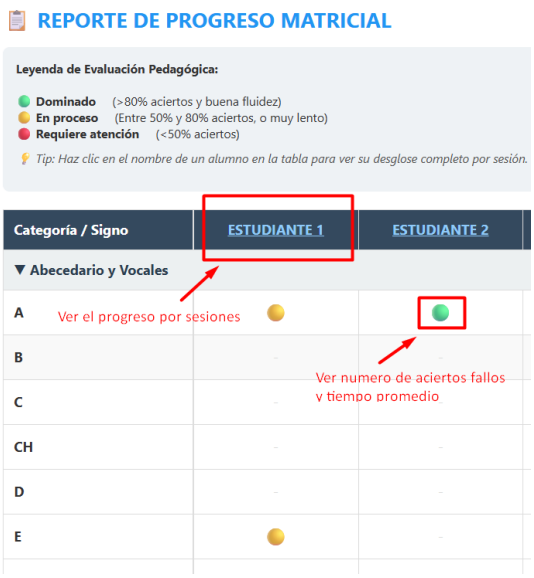


Resultado esperado:

La docente debe identificar visualmente qué señas están dominadas, cuáles están en proceso y cuáles requieren refuerzo.

Detalle del desempeño por signo

Al seleccionar un estudiante o una seña dentro del reporte matricial, el sistema muestra una ventana emergente con información resumida del desempeño. Esta ventana permite revisar el estado del estudiante en una seña específica, incluyendo aciertos, errores y tiempo promedio de respuesta.



Para revisar el detalle de una seña, se deben seguir los siguientes pasos:

1. Ingresar al reporte matricial.
2. Seleccionar el estudiante o signo que se desea revisar.
3. Leer el estado mostrado en la ventana emergente.
4. Revisar los aciertos, errores y tiempo promedio.
5. Presionar **Enviar a Signos a Reforzar** si la seña requiere más práctica.
6. Cerrar la ventana para regresar a la matriz.

Resultado esperado:
La docente debe conocer el desempeño específico del estudiante en una seña determinada y decidir si requiere refuerzo.



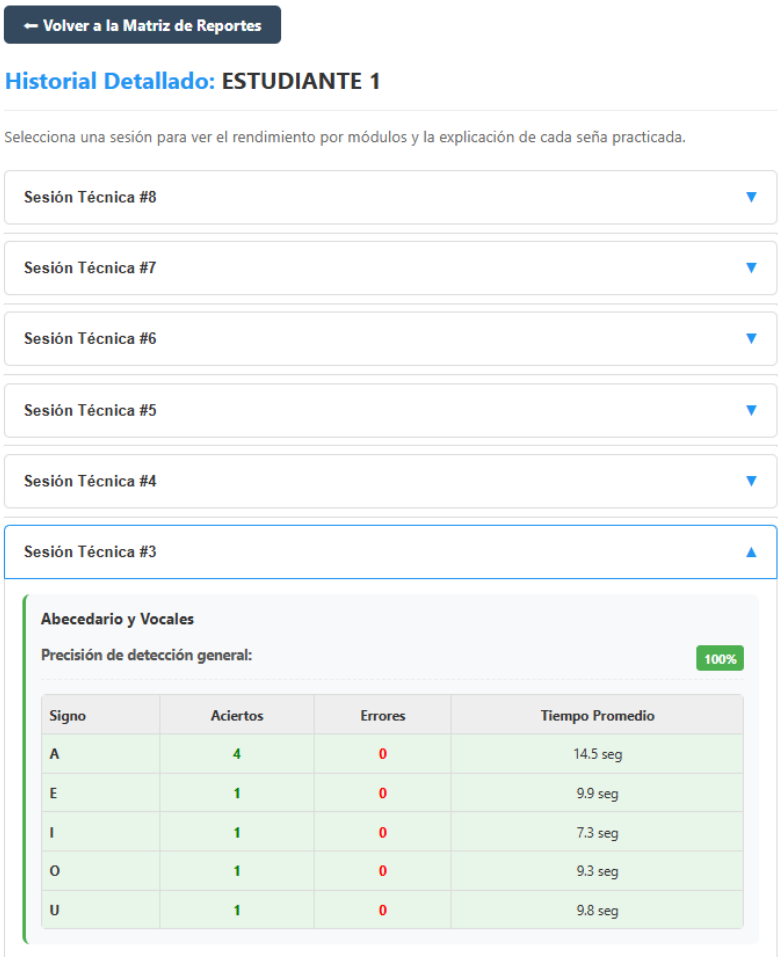
Historial detallado por sesiones

El sistema también permite revisar el historial detallado de un estudiante por sesión. Esta sección muestra las sesiones registradas y permite desplegar cada una para observar el rendimiento por categoría, seña, aciertos, errores y tiempo promedio.

Para revisar el historial detallado, se deben seguir los siguientes pasos:

1. Ingresar al reporte del estudiante.
2. Seleccionar la sesión que se desea revisar.
3. Desplegar la sesión presionando sobre la barra correspondiente.
4. Revisar la categoría practicada.
5. Observar la tabla de signos, aciertos, errores y tiempo promedio.
6. Presionar **Volver a la Matriz de Reportes** para regresar a la vista general.

Resultado esperado:
La docente debe visualizar el desempeño del estudiante por sesión y reconocer qué señas fueron practicadas, cuántos aciertos tuvo, si existieron errores y cuánto tiempo tardó en responder.

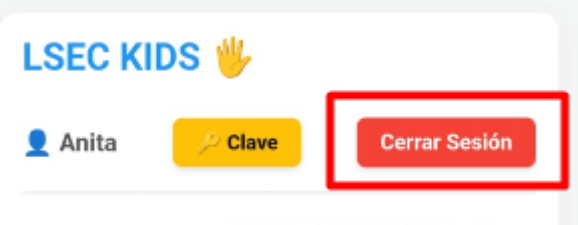


Cierre de sesión del dashboard

Al finalizar el uso del dashboard, la docente debe cerrar sesión para evitar accesos no autorizados.

Pasos básicos:

1. Ubicar el botón **Cerrar Sesión**.
2. Presionar el botón.
3. Verificar que el sistema cierre la sesión activa.



Resultado esperado:
El dashboard debe cerrar la sesión del usuario docente.

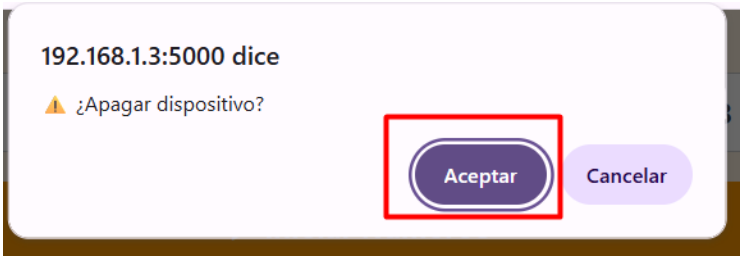
Apagado del sistema

Para finalizar el uso del sistema, se debe apagar el prototipo de forma segura.



Pasos básicos:

1. Finalizar la sesión activa desde el dashboard.
2. Verificar que los resultados se hayan guardado.
3. Presionar el botón **Apagar sistema**
4. Esperar a que el sistema termine el proceso de apagado.
5. Desconectar la fuente de alimentación únicamente después del apagado.



Resultado esperado:
El sistema debe apagarse sin pérdida de información.

Buenas prácticas de uso

- Verificar que el estudiante esté correctamente seleccionado antes de iniciar.
- Ubicar al niño frente a la cámara.
- Mantener buena iluminación.
- Evitar que otros objetos o personas interfieran en la captura.
- Seleccionar pocas señas al inicio si el estudiante está aprendiendo.
- Revisar los reportes al finalizar cada sesión
- Finalizar la sesión antes de apagar el sistema.

Problemas frecuentes y solución

Problema	Posible causa	Solución recomendada
El dashboard no abre	El dispositivo no está en la red local	Conectarse a la misma red del prototipo.
El estudiante no aparece como activo	No se seleccionó estudiante	Presionar el botón de selección junto al estudiante.
No se detecta la mano	Mala iluminación o mano fuera de cámara	Mejorar iluminación y ubicar la mano frente a la cámara.
La práctica no inicia	No se seleccionó ninguna seña	Marcar al menos una letra, número o gesto.
No aparecen reportes	No existen sesiones registradas	Realizar una práctica y finalizar la sesión correctamente.
El resultado no coincide	Gesto incompleto o mala posición	Repetir la seña con la mano visible.

Limitaciones del sistema

El sistema puede presentar dificultades de reconocimiento cuando existe poca iluminación, cuando la mano no se encuentra dentro del área de captura, cuando la seña se realiza de forma incompleta o cuando existe oclusión de los dedos. Por esta razón, se recomienda que la docente supervise la práctica y oriente al estudiante durante el uso del prototipo.