

UNIVERSIDAD TÉCNICA DEL NORTE



FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS CARRERA DE SOFTWARE

DESARROLLO DE UNA SOLUCIÓN TECNOLÓGICA DE 'AUTENTICACIÓN Y AUTORIZACIÓN' PARA LA
PLATAFORMA MÓVIL DE LA UNIVERSIDAD TÉCNICA DEL NORTE APLICANDO LAS
RECOMENDACIONES DE SEGURIDAD DEL OWASP TOP TEN

Trabajo de grado previo a la obtención del título de Ingeniero de Software presentado ante la
ilustre Universidad Técnica del Norte

AUTOR:

Erick Sebastian Carlosama Tabango

DIRECTOR:

Ing. Marco Remigio Pusdá Chulde, Ph. D.

Ibarra- Ecuador

2025



UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

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			
CÉDULA DE IDENTIDAD:	1004733695		
APELLIDOS Y NOMBRES:	CARLOSAMA TABANGO ERICK SEBASTIAN		
DIRECCIÓN:	IBARRA, LA MAGDALENA – VIA A ZULETA		
EMAIL:	escarlosamat@utn.edu.ec / sebastiantabango@gmail.com		
TELÉFONO FIJO:		TELÉFONO MÓVIL:	098 846 1840

DATOS DE LA OBRA	
TÍTULO:	DESARROLLO DE UNA SOLUCIÓN TECNOLÓGICA DE 'AUTENTICACIÓN Y AUTORIZACIÓN' PARA LA PLATAFORMA MÓVIL DE LA UNIVERSIDAD TÉCNICA DEL NORTE APLICANDO LAS RECOMENDACIONES DE SEGURIDAD DEL OWASP TOP TEN.
AUTOR (ES):	CARLOSAMA TABANGO ERICK SEBASTIAN
FECHA DE APROBACIÓN: DD/MM/AAAA	03/09/2025
PROGRAMA:	☒ PREGRADO ☐ POSGRADO
TÍTULO POR EL QUE OPTA:	INGENIERO DE SOFTWARE
DIRECTOR:	Ing. PUSDÁ CHULDE MARCO REMIGIO, Ph. D.

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 04 días del mes de septiembre de 2025

EL AUTOR:



Erick Sebastian Carlosama Tabango

CI: 1004733695



UNIVERSIDAD TÉCNICA DEL NORTE BIBLIOTECA UNIVERSITARIA

CERTIFICACIÓN DEL DIRECTOR

Certifico que el trabajo de grado “DESARROLLO DE UNA SOLUCIÓN TECNOLÓGICA DE ‘AUTENTICACIÓN Y AUTORIZACIÓN’ PARA LA PLATAFORMA MÓVIL DE LA UNIVERSIDAD TÉCNICA DEL NORTE APLICANDO LAS RECOMENDACIONES DE SEGURIDAD DEL OWASP TOP TEN”, ha sido desarrollado en su totalidad por el señor Erick Sebastian Carlosama Tabango, portador de la cédula de identidad número 1004733695.

MARCO
REMIGIO
PUSDA CHULDE

Firmado digitalmente por
MARCO REMIGIO PUSDA
CHULDE
Fecha: 2025.09.04 10:14:06
-05'00'

Ing. PUSDÁ Chulde Marco Remigio, Ph. D.

Director de Trabajo de Grado

DEDICATORIA

Este trabajo está dedicado con todo mi corazón a quienes han sido el pilar fundamental de mi vida.

A mis padres, Juan y Esthela, porque en su ejemplo encontré la fuerza para luchar y en su amor incondicional hallé el refugio en cada momento de dificultad. Gracias por enseñarme que la verdadera grandeza está en la humildad, el esfuerzo y la constancia; este triunfo es también el fruto de sus sacrificios y sueños.

A mis hermanos, que con su compañía y apoyo constante me recordaron siempre el valor de la unidad familiar y la importancia de caminar juntos en cada etapa de la vida.

A Evelyn, mi compañera de vida, por caminar a mi lado con paciencia y ternura, por sostenerme en los momentos de cansancio y recordarme que toda meta se alcanza mejor cuando se comparte.

A mis amigos, quienes con su amistad sincera y su compañía en los buenos y malos momentos hicieron más ligero este recorrido, demostrando que la verdadera amistad también forma parte de los grandes logros.

Más que un logro personal, este es un homenaje a todos quienes caminaron conmigo; a ustedes, con gratitud y amor eterno, les dedico este fruto.

AGRADECIMIENTO

Con la satisfacción de haber llegado a la meta, reconozco con gratitud a quienes, con su apoyo y enseñanza, dejaron una huella imborrable en mi camino y me impulsaron a seguir adelante con firmeza.

En primer lugar, a mis padres, Juan y Esthela, por su amor incondicional, por ser mi mayor fuente de inspiración y apoyo, y por enseñarme con su ejemplo que el esfuerzo, la constancia y la perseverancia hacen posible alcanzar los sueños. A mis hermanos, por su compañía, comprensión y respaldo en cada etapa de mi vida. A mi pareja, Evelyn, por su paciencia, motivación y por brindarme la fortaleza necesaria en los momentos más desafiantes.

De manera especial, al Ing. Marco Remigio Pusd Chulde, Ph.D., director de este trabajo, y al Ing. Jos Antonio Quia Mera, Ph.D., asesor, por su gua constante, su confianza y las valiosas recomendaciones que orientaron y fortalecieron este proyecto; as como al resto de docentes universitarios, quienes con su dedicacin y entrega contribuyeron de manera significativa a mi formacin, despertando en m el compromiso por el aprendizaje y el deseo de seguir creciendo profesionalmente.

A mis amigos y compaeros, quienes estuvieron presentes durante todo este trayecto, compartiendo alegras, esfuerzos y aprendizajes, y cuyo apoyo y compaa hicieron de este camino una experiencia ms llevadera y enriquecedora.

Finalmente, agradezco a todas aquellas personas que, de manera directa o indirecta, hicieron posible la culminacin de este trabajo. Este logro es tambin fruto del esfuerzo compartido.

Que este sea un recordatorio de que nunca debemos dejar de soar, creer y crear soluciones que mejoren el mundo.

Tabla de Contenidos

INTRODUCCIÓN	1
PLANTEAMIENTO DEL PROBLEMA	1
OBJETIVOS.....	1
<i>Objetivo General.....</i>	<i>1</i>
<i>OBJETIVOS ESPECÍFICOS</i>	<i>2</i>
ALCANCE	2
METODOLOGÍA	3
JUSTIFICACIÓN	4
CONTEXTO DE LA INVESTIGACIÓN	4
I. CAPÍTULO 1: MARCO TEÓRICO.....	6
1.1. SISTEMA INFORMÁTICO INTEGRADO UNIVERSITARIO	6
1.1.1. <i>Importancia del SIU en la gestión universitaria.....</i>	<i>6</i>
1.2. PLATAFORMAS MÓVILES.....	7
1.2.1. <i>Tendencias actuales en el desarrollo de aplicaciones móviles.....</i>	<i>7</i>
1.2.2. <i>Aplicaciones móviles multiplataformas.....</i>	<i>8</i>
1.3. HERRAMIENTAS Y ENTORNO DE DESARROLLO	8
1.3.1. <i>Herramientas de control de versiones.....</i>	<i>8</i>
1.3.2. <i>Entorno de desarrollo integrado (IDE).....</i>	<i>9</i>
1.3.3. <i>Visual Studio Code</i>	<i>9</i>
1.3.4. <i>Bases de datos Oracle</i>	<i>10</i>
1.3.5. <i>Tecnología para el desarrollo.....</i>	<i>10</i>
1.4. METODOLOGÍA DE DESARROLLO DE SOFTWARE.....	13
1.4.1. <i>Metodología de Desarrollo Ágil SCRUM.....</i>	<i>14</i>
1.4.2. <i>Fases de SCRUM</i>	<i>14</i>
1.5. ARQUITECTURA DE MICROSERVICIOS	17
1.5.1. <i>Ventaja de los microservicios</i>	<i>17</i>
1.5.2. <i>Patrones de diseño en microservicios</i>	<i>18</i>
1.6. SEGURIDAD INFORMÁTICA Y SEGURIDAD DE LA INFORMACIÓN	18
1.6.1. <i>Principios fundamentales de seguridad de la información.....</i>	<i>19</i>
1.6.2. <i>Amenazas y vulnerabilidades comunes.....</i>	<i>19</i>
1.7. SEGURIDAD EN SISTEMAS MICROSERVICIOS.....	20
1.7.1. <i>Autenticación y autorización en sistemas microservicios.....</i>	<i>20</i>
1.7.2. <i>Modelos de autorización</i>	<i>21</i>
1.7.3. <i>Autorización basada en roles (RBAC)</i>	<i>22</i>
1.7.4. <i>Tecnologías y protocolos de autenticación y autorización.....</i>	<i>22</i>

1.8.	RECOMENDACIONES OWASP TOP TEN.....	24
1.8.1.	Directrices de OWASP Top Ten 2021.....	24
1.9.	RESULTADOS A PARTIR DEL ESTUDIO	26
II.	CAPÍTULO 2: DESARROLLO.....	28
2.1.	DESARROLLO DE FASE II: PRE-JUEGO	28
2.1.1.	Equipo SCRUM.....	28
2.1.2.	Elicitación de Requerimientos	29
2.2.	DESARROLLO DE FASE II: JUEGO	37
2.2.1.	Sprint 0.	37
2.2.2.	Sprint 1.	46
2.2.3.	Sprint 2.	53
2.2.4.	Sprint 3	60
2.2.5.	Sprint 4	69
2.2.6.	Sprint 5	75
2.3.	DESARROLLO DE FASE III: POST-JUEGO.....	86
III.	CAPÍTULO 3: VALIDACION DE RESULTADOS	87
3.1.	DEFINICIÓN DE CRITERIOS DE EVALUACIÓN DE SEGURIDAD.....	87
3.2.	APLICACIÓN DE HERRAMIENTAS KALI POR CATEGORÍA OWASP	88
3.2.1.	A01 – Broken Access Control	88
3.2.2.	A02 – Cryptographic Failures	92
3.2.3.	A03 – Injection	95
3.2.4.	A04 – Injection Desing.....	98
3.2.5.	A05 – Security Misconfiguration	100
3.2.6.	A06 - Vulnerable and outdated componets	103
3.2.7.	A07 – Identification and authentication failures.....	109
3.2.8.	A08 - Software and data integrity failures.	113
3.2.9.	A09 – Security logging and monitoring failures.....	116
3.2.10.	A10 – Server-side request failure	121
3.2.11.	Síntesis y representación gráfica de riesgos OWASP.....	123
3.3.	VALIDACIÓN DE PRINCIPIOS CIA EN LA PLATAFORMA UTN MÓVIL.	126
3.3.1.	Confidencialidad	126
3.3.2.	Integridad.....	126
3.3.3.	Disponibilidad	127
3.3.4.	Matriz de cumplimiento y análisis comparativo CIA.....	127
	CONCLUSIONES	129
	RECOMENDACIONES	130

REFERENCIAS	131
-------------------	-----

Índice de Figuras

Figura 1. Árbol de Problemas.....	1
Figura 2. Arquitectura Propuesta.....	3
Figura 3. Flujo Básico de Operaciones Git [25]	9
Figura 4. Comunicación cliente-servidor mediante REST API [38].....	11
Figura 5. Resultados del experimento de la comparación de tecnologías para desarrollo de APIs Rest	12
Figura 6. Arquitectura de Node.JS [43]	13
Figura 7. Proceso de SCRUM [59]	16
Figura 8. Arquitectura de microservicios [62].....	17
Figura 9. Triángulo CIA [77].....	19
Figura 10. Control de Acceso Basado en Roles [88].....	22
Figura 11. Diagrama de Fases de Desarrollo.....	28
Figura 12. Arquitectura Microservicios General para la plataforma UTN Móvil.....	38
Figura 13. Estructura del API Microservicio de Seguridad.....	39
Figura 14. Estructura del API Gateway.....	41
Figura 15. Estructura de carpetas por sub-aplicación.....	42
Figura 16. Estructura de capas para cada caso (Caso Login).....	43
Figura 17. Inyección de dependencias (Caso login)	44
Figura 18. Consolidación de dependencias (Módulo Auth).....	44
Figura 19. Gestión de rutas (Módulo Auth)	45
Figura 20. Diagrama de secuencia - Inicio de sesión	47
Figura 21. Pantalla de Inicio de Sesión.....	48
Figura 22. Diagrama de secuencia - Inicio de Sesión con envío de código OTP.....	49
Figura 23. Pantalla de ingreso de código OTP.....	50
Figura 24. Diagrama de secuencia - Redireccionamiento de solicitudes en API Gateway.....	51
Figura 25. Resultados de test unitario en API Gateway	52
Figura 26. Diagrama de secuencia - Comportamiento de token de acceso y refresco	55
Figura 27. Resultados de pruebas unitarias – Renovación automática de token.	56
Figura 28. Pantalla de login con mensaje de sesión caducada	57
Figura 29. Diagrama de secuencia - Gestión de múltiples sesiones	58
Figura 30. Pantalla 'Gestión de sesiones activas'	58
Figura 31. Estructura del token de acceso (Parcialmente oculta).....	62
Figura 32. Diagrama de secuencia - Acceso a módulo según permisos.....	62

Figura 33. Decorador Public.....	63
Figura 34. Modal mostrado cuando usuario no tiene permiso para x módulo	63
Figura 35. Archivo applications_constants	64
Figura 36. Pantalla 'Home' para aplicación UTN Móvil.....	65
Figura 37. Pantalla 'Acerca de' cuando el usuario inició sesión.....	65
Figura 38. Pantalla 'Home' de modo invitado.....	66
Figura 39. Pantalla 'Acerca de' para el modo invitado.....	67
Figura 40. Pantalla de 'Login' incluido botón de redireccionamiento para cambio de contraseña .	70
Figura 41. Diagrama de secuencia - Autenticación en UTN Móvil	71
Figura 42. Pantalla de 'Autenticación biométrica local'	72
Figura 43. Diagrama de secuencia - Verificación de conexión a internet	73
Figura 44. Pantalla de 'Conectividad'	73
Figura 45. Fragmento del archivo app.module.ts que registra dinámicamente los microservicios en el Gateway.....	77
Figura 46. Diagrama de secuencia - Control de versión.....	77
Figura 47. Pantalla 'Actualización de versión'	78
Figura 48. Fragmento de código para envío de datos adicionales de sesión desde Flutter	79
Figura 49. Pantalla de 'Gestión de sesiones'	79
Figura 50. Pantalla 'Home'	80
Figura 51. Composición de servicios en entorno Docker para pre-producción.....	81
Figura 52. Pantalla 'Formulario para parámetros generales'	82
Figura 53. Pantalla 'Listado de parámetros generales'.....	82
Figura 54. Pantalla 'Listado de sesiones activas'	83
Figura 55. Pantalla 'Historial de sesiones'	83
Figura 56. Pantalla 'Gestión de Funciones'	84
Figura 57. Pantalla 'Asignación de funciones'	84
Figura 58. Fragmento de Token generado para usuario sin permiso.....	89
Figura 59. Pantalla de Burp Suite donde receptamos la solicitud y agregamos el token (Prueba 01 de A01)	89
Figura 60. Pantalla de historial de solicitudes de Burp Suite (Prueba 01 de A01)	90
Figura 61. Fragmento de token generado para usuario sin rol Administrativo	91
Figura 62. Pantalla de Burp Suite donde receptamos la solicitud y agregamos el token (Prueba 02 de A01)	91
Figura 63. Pantalla de historial de solicitudes de Burp Suite (Prueba 02 de A01)	92

Figura 64. Configuración dinámica de la estrategia JWT con clave obtenida desde parámetros del sistema	93
Figura 65. Generación del token de acceso firmado con algoritmo simétrico y expiración configurada.	93
Figura 66. Generación del refresh token firmado con algoritmo asimétrico y expiración controlada.	94
Figura 67. Decodificación de contraseña durante el inicio de sesión.	94
Figura 68. Cifrado automático de valores sensibles antes de guardar parámetros.....	94
Figura 69. Servicio de cifrado y descifrado.	95
Figura 70. Comando de SQLMap ejecutado sobre el endpoint GET /parametros/:id para evaluar inyección en URL.	96
Figura 71. Comando de SQLMap ejecutado sobre el endpoint GET /parametros/app/version para evaluar inyección en URL.	97
Figura 72. Resultado de la prueba manual con payloads maliciosos enviada mediante Postman al endpoint protegido PUT /parametros/...	98
Figura 73. Arquitectura general de la plataforma UTN Móvil, compuesta por frontend, microservicios desacoplados, API Gateway y base de datos centralizada.....	99
Figura 74. Resultado del escaneo de puertos realizado con Nmap para la IP pública del servidor del Gateway.....	101
Figura 75. Resultado del escaneo de puertos, donde el puerto 443 aparece como filtrado.....	101
Figura 76. Resultado del escaneo con Nikto	101
Figura 77. Implementación del middleware de cabeceras de seguridad en NestJS.....	102
Figura 78. Configuración de las variables de entorno en el API Gateway, donde se cargan las claves y URLs para los servicios	102
Figura 79. Configuración de la conexión a la base de datos en un microservicio, donde las credenciales se obtienen de las variables de entorno para proteger la información sensible.....	103
Figura 80. Informe inicial del análisis de dependencias con npm audit.	104
Figura 81. Resolución de vulnerabilidades con npm audit fix.....	104
Figura 82. Resultado del análisis posterior con npm audit.	105
Figura 83. Ciclo de soporte de Redis versión 7.2, indicando el fin de soporte activo y la vigencia del soporte de seguridad hasta 2026 [96].	106
Figura 84. Tabla de soporte oficial de Oracle Database, donde se evidencia que la versión 19c continúa dentro del ciclo de mantenimiento [97].	107

Figura 85. Salida del comando flutter pub outdated que muestra las dependencias desactualizadas y sus versiones más recientes disponibles.....	108
Figura 86. Información de la imagen base de Node.js utilizada en el contenedor Docker.	109
Figura 87. Información de la imagen base de Redis utilizada en el contenedor Docker.....	109
Figura 88. Validación de credenciales mediante autenticación de Microsoft.	110
Figura 89. Respuesta si el usuario tiene MFA activado.....	110
Figura 90. Método para crear una sesión	111
Figura 91. Método para validar el límite de sesiones por usuario.....	112
Figura 92. Método para enviar un código OTP generado.	112
Figura 93. Método para validar un código OTP.....	113
Figura 94. Fragmento de código que muestra la generación de un token de acceso firmado.	114
Figura 95. Fragmento de código que muestra la verificación de un token de acceso.	114
Figura 96. Fragmento de código que muestra la generación de un refresh token firmado.	114
Figura 97. Fragmento de código que muestra la verificación de un refresh token utilizando la clave pública y el algoritmo asíncrono.....	115
Figura 98. Código que valida los permisos de un usuario antes de permitirle acceder a una ruta.	115
Figura 99. Código que gestiona el cierre de sesiones.	116
Figura 100. Fragmento de código que muestra el LoggerInterceptor para registrar información de cada solicitud HTTP.	117
Figura 101. Fragmento de código que muestra el CustomExceptionHandler para la captura y registro de excepciones.....	118
Figura 102. Fragmento de código que muestra la CustomValidationPipe para la validación de datos entrantes.....	119
Figura 103. Fragmento de código que muestra el registro de errores en el ProxyService.	120
Figura 104. Fragmento de configuración de Winston para el registro de logs en diferentes entornos.	121
Figura 105. Fragmento de código que muestra la redirección dinámica mediante createProxyMiddleware.....	122
Figura 106. Fragmento de código que reescritura de rutas para redireccionamiento transparente en el Gateway.....	122
Figura 107. Mapa de Calor de Riesgos – Evaluación OWASP UTN Móvil.....	123
Figura 108. Distribución porcentual del cumplimiento de principios CIA en la plataforma UTN Móvil.	128

Índice de Tablas

Tabla 1. Trabajos Relacionados	4
Tabla 2. Comparativa entre modelos de autorización aplicables en microservicios.....	21
Tabla 3. Comparativa de mecanismos de autenticación/autorización para microservicios	22
Tabla 4. Alineación de OWASP Top Ten 2021 con mecanismos implementados en UTN Móvil	25
Tabla 5. Equipo SCRUM del proyecto.....	29
Tabla 6. HU-SEC-01: Inicio de sesión	30
Tabla 7. HU-SEC-02: Cambio de contraseña de Microsoft	30
Tabla 8. HU-SEC-03: Refresco automático del token de acceso.....	30
Tabla 9. HU-SEC-04: Manejo de tokens expirados o inválidos	31
Tabla 10. HU-SEC-05: Control de múltiples sesiones por usuario.....	32
Tabla 11. HU-SEC-06: Acceso a aplicaciones según permisos.....	32
Tabla 12. HU-SEC-07: Autenticación biométrica	33
Tabla 13. HU-SEC-08: Configuración de parámetros generales desde fronted administrativo.....	33
Tabla 14. HU-SEC-09: Verificación de conexión a internet.....	33
Tabla 15. HU-SEC-10: Acceso como invitado sin autenticación	34
Tabla 16. HU-SEC-11: Carga de apps en el home	34
Tabla 17. HU-SEC-12: Envío de código OTP al correo institucional (MFA Habilitado).....	35
Tabla 18. HU-SEC-13: Api Gateway para redireccionar solicitudes	35
Tabla 19. HU-SEC-14: Control de versión	36
Tabla 20. Product Backlog del proyecto.....	36
Tabla 21. Sprint Backlog - Sprint 1	46
Tabla 22. Resumen de pruebas unitarias ejecutadas a API Gateway.....	51
Tabla 23. Revisión de criterios de aceptación involucrados en Sprint 1	52
Tabla 24. Sprint Backlog - Sprint 2	54
Tabla 25. Resumen de pruebas unitarias a TokenService	56
Tabla 26. Revisión de criterios de aceptación involucrados Sprint 2	59
Tabla 27. Sprint Backlog - Sprint 3	60
Tabla 28. Revisión de criterios de aceptación involucrados en Sprint 3	67
Tabla 29. Sprint Backlog - Sprint 4	69
Tabla 30. Revisión de criterios de aceptación en Sprint 4	74
Tabla 31. Sprint Backlog - Sprint 5	75
Tabla 32. Revisión de criterios de aceptación en Sprint 5	85
Tabla 33. Evaluación de seguridad basada en OWASP Ton Ten y herramienta Kali Linux.....	87

Tabla 34. Argumentación técnica para la clasificación de vulnerabilidades OWASP según impacto y probabilidad	123
Tabla 35. Resumen del cumplimiento general de la evaluación al módulo de seguridad de la plataforma UTN Móvil.....	125
Tabla 36. Matriz de cumplimiento de principios CIA en la plataforma UTN Móvil.....	127

Resumen

El presente trabajo describe el diseño, desarrollo e implementación del módulo de autenticación y autorización de la plataforma UTN Móvil, con enfoque en seguridad, escalabilidad y experiencia de usuario. Bajo una arquitectura basada en microservicios desarrollados en NestJS y una aplicación móvil Flutter con arquitectura limpia, se garantizó una solución modular, mantenible y adaptable a futuro. Se implementaron mecanismos avanzados de autenticación como el inicio de sesión con credenciales institucionales, autenticación multifactor (MFA) mediante código OTP, manejo seguro de tokens JWT (acceso y refresh), control de sesiones simultáneas y validación dinámica de permisos por usuario.

El sistema cuenta con un API Gateway que centraliza las solicitudes, valida tokens, aplica políticas de autorización y enruta dinámicamente hacia los microservicios. Se desarrolló un panel administrativo para la gestión de parámetros de seguridad, y se integró pruebas unitarias y de integración que verifican el cumplimiento de los requisitos definidos por historias de usuario bajo metodología SCRUM.

También se verificó el cumplimiento de las recomendaciones del estándar OWASP Top Ten 2021 utilizando herramientas de pentesting y análisis estático para detectar vulnerabilidades. Se aplicó un análisis de los principios de seguridad de la información (Confidencialidad, Integridad y Disponibilidad), generando una matriz de los tres principios en donde se destaca la solidez del diseño frente a amenazas comunes y fallos de implementación.

La evaluación del cumplimiento de las recomendaciones de OWASP Top Ten concluye que el sistema cumple con criterios técnicos, normativos y de seguridad, lo que demuestra que es apto para su despliegue institucional y su posterior evolución.

Palabras clave: UTN Móvil, plataforma móvil, autenticación, autorización, seguridad informática, microservicios, NestJS, Flutter, JWT, autenticación multifactor (MFA), API Gateway, OWASP Top ten, SCRUM.

Abstract

This work presents the design, development, and implementation of the authentication and authorization module for the UTN Móvil platform, focusing on security, scalability, and user experience. Based on a microservices architecture developed in NestJS and a mobile application built with Flutter using a clean architecture approach, the solution ensures modularity, maintainability, and adaptability for future needs. Advanced authentication mechanisms were implemented, including login with institutional credentials, multi-factor authentication (MFA) via OTP, secure management of JWT tokens (access and refresh), control of concurrent sessions, and dynamic permission validation per user.

The system includes an API Gateway that centralizes requests, validates tokens, enforces authorization policies, and dynamically routes traffic to the corresponding microservices. An administrative panel was developed to manage key security parameters, and both unit and integration tests were conducted to verify compliance with user stories defined under the SCRUM methodology.

Compliance with the OWASP Top Ten 2021 standard recommendations was also verified using pentesting tools and static analysis to detect vulnerabilities. An analysis of the information security principles (Confidentiality, Integrity, and Availability) was applied, generating a matrix of the three principles that highlights the robustness of the design against common threats and implementation failures.

The evaluation of compliance with the OWASP Top Ten recommendations concludes that the system meets technical, regulatory, and security criteria, demonstrating its suitability for institutional deployment and subsequent evolution.

Keywords: UTN Móvil, mobile platform, authentication, authorization, cybersecurity, microservices, NestJS, Flutter, JWT, multi-factor authentication (MFA), API Gateway, OWASP Top ten, SCRUM

INTRODUCCIÓN

Planteamiento del Problema

En el contexto actual, donde un gran porcentaje de la población y su interacción con los dispositivos móviles han redefinido el consumo de servicios digitales. La atención de los jóvenes se centra en contenidos accesibles a través de smartphones y tabletas, lo que subraya la importancia de desarrollar plataformas adaptativas y seguras [1]. Con la creciente de las amenazas cibernéticas, como las configuraciones inadecuadas y vulnerabilidades de los softwares señaladas por Symantec [2], se requiere que los módulos de seguridad combinen una autenticación efectiva y una gestión de permisos ajustados a normativas y prácticas recomendadas por estándares. Las recomendaciones de OWASP son efectivas para contrarrestar los problemas de seguridad de las aplicaciones web, centrándose en los 10 riesgos más importantes [3]. Bogantes [4], en su trabajo, nos menciona lo importante de considerar siempre los tres pilares de la seguridad informática: confidencialidad, integridad y disponibilidad. Este enfoque pretende mitigar riesgos actuales y anticiparse a futuras amenazas, reforzando la confianza en los sistemas de información de la universidad en un ambiente digital predominantemente móvil. En Figura 1 se detalla el problema a cubrirse con el presente trabajo.

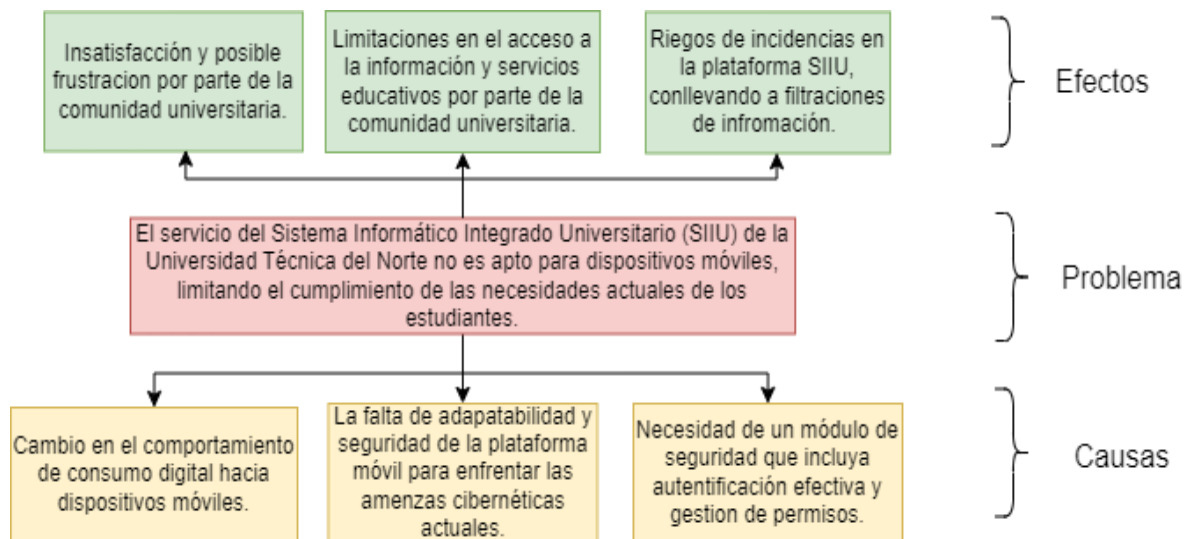


Figura 1. Árbol de Problemas

Objetivos

Objetivo General

Desarrollar una solución tecnológica de autenticación y autorización para la plataforma móvil de la Universidad Técnica del Norte (UTN), que garantice la seguridad en el acceso y gestión de permisos mediante la aplicación de las recomendaciones del estándar OWASP Top Ten.

Objetivos Específicos

1. Redactar un marco teórico sobre el diseño y desarrollo de un módulo de autenticación y autorización, con énfasis en la gestión de permisos en sistemas basados en microservicios.
2. Desarrollar la solución tecnológica que integre mecanismos de autenticación y autorización para la plataforma móvil de la UTN, conforme a las recomendaciones de OWASP Top Ten.
3. Evaluar la solución tecnológica utilizando las recomendaciones del OWASP Top Ten, asegurando el cumplimiento de los principios fundamentales de la seguridad informática.

Alcance

El alcance del presente trabajo se enfoca en la investigación, el diseño y el desarrollo de una solución tecnológica de autenticación y autorización para la plataforma móvil de la UTN, la cual está disponible tanto para el módulo de portafolio docente como para el portafolio estudiante. Para ello, se exploró la organización actual de las medidas de autenticación y autorización en las bases de datos Oracle utilizadas por la institución, con el fin de entender su funcionamiento y utilizar esta información como apoyo en el desarrollo.

Posteriormente, se procedió a analizar, diseñar y desarrollar la solución tecnológica de autenticación y autorización. La arquitectura bajo la cual se estructura la solución tecnológica es la de microservicios. La selección de esta arquitectura se debe a sus beneficios claves, tales como el desarrollo de un sistema altamente modular que facilita el mantenimiento, la capacidad de desplegar servicios rápidamente, y la creación de unidades de código altamente cohesivas que son más fáciles de razonar y administrar de forma aislada [5]. En Figura 2 muestra cómo está definida la solución tecnológica.

El desarrollo del backend se basó en un experimento comparativo entre 4 tecnologías elegidas: Oracle APEX, Spring Boot, Node.js y Python, seleccionadas por su rendimiento en el backend. En cuanto a la base de datos, se aprovechó los recursos de la institución, es decir, se utilizó una base de datos Oracle. El frontend administrativo se desarrolló utilizando Angular, lo que permitió una mejor integración con el backend y proporcionó una interfaz de usuario moderna, escalable y de alto rendimiento.

Finalmente, se garantizó que la solución tecnológica esté preparada para los riesgos de seguridad informática emergentes en aplicaciones web y móviles, abordando de manera eficaz las recomendaciones del OWASP Top Ten.

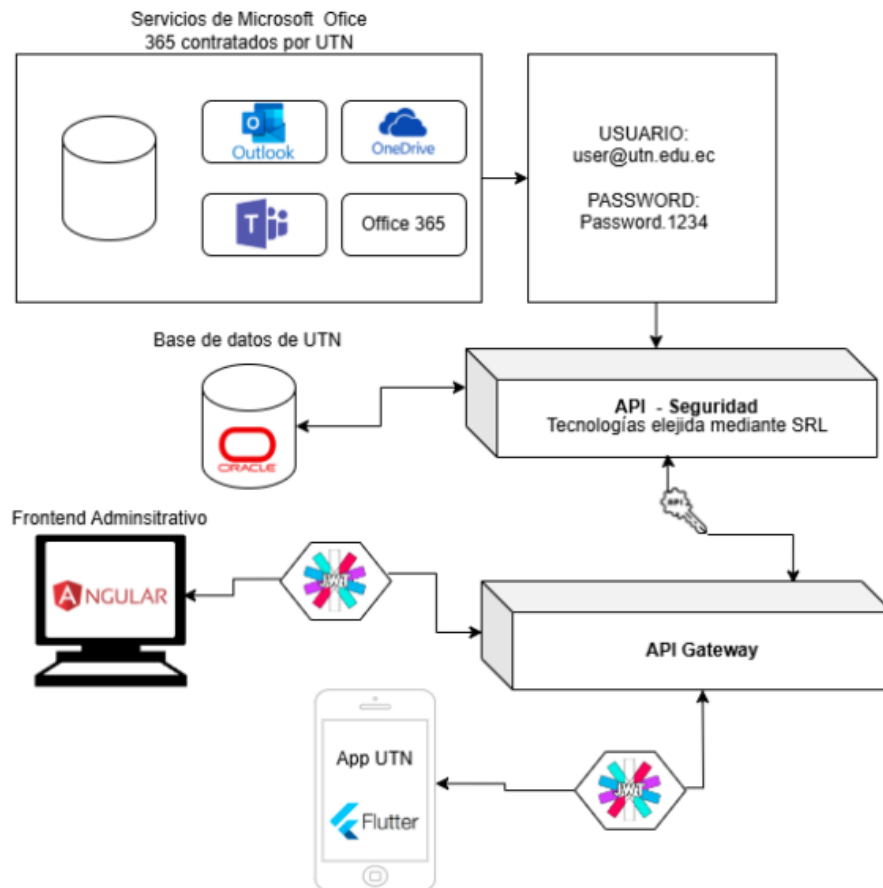


Figura 2. Arquitectura Propuesta

Metodología

El presente trabajo de investigación es de tipo aplicado, debido a su aplicación inmediata. En cuanto a la técnica de investigación, es de tipo documental para el desarrollo del marco teórico. Se utilizó las bases científicas disponibles en la biblioteca virtual de la institución, y como gestor bibliográfico el software Mendeley.

Las siguientes actividades se relacionan estrechamente con los objetivos:

- Actividad 1: Se llevó a cabo una investigación literaria académica técnica relacionada al desarrollo módulos de autenticación y autorización en arquitectura de microservicios. Basándose en la información recolectada se describe la teoría y los fundamentos principales.
- Actividad 2: Se exploró las medidas actuales de autenticación y autorización de los usuarios en la base de datos de la UTN, con el fin de utilizar esta información para el desarrollo de la solución tecnológica. Se diseñó la arquitectura de la solución, teniendo en cuenta las interacciones y el funcionamiento de los módulos a desarrollarse. Se adoptó la metodología SCRUM para garantizar un proceso ágil y eficiente. Una vez establecida la arquitectura, se precedió con el desarrollo de la solución.

- Actividad 3: Se evaluó la solución tecnológica mediante el cumplimiento de las recomendaciones de OWASP Top Ten 2021

Justificación

El proyecto se alinea estrechamente con los principios de los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas, específicamente con el objetivo 9 de “construir infraestructuras resilientes, promover la industrialización inclusiva y sostenible, y fomentar la innovación” [6].

La propuesta aporta el desarrollo de una infraestructura de seguridad y adaptativa para la plataforma móvil. El resultado de esto aporta conocimientos en seguridad informática y desarrollo de soluciones tecnológicas eficientes, pues se aplicará técnicas y teorías como JSON Web Token (JWT) para la gestión de sesiones, autenticación y autorización de usuarios, y recomendaciones proporcionadas por OWASP. Este conocimiento promueve significativamente, en el campo de la ingeniería en software, y el desarrollo de soluciones tecnológicas seguras, innovadoras y confiables.

El desarrollo de la solución tecnológica proporciona una experiencia directa de diseño de arquitectura, desarrollo y la gestión de esta en entornos reales. Los conocimientos brindados en esta propuesta son beneficiosos para futuros profesionales de la tecnología e ingeniería, pues los prepara para afrontar problemas similares

Contexto de la investigación

Tabla 1. Trabajos Relacionados

INVESTIGACIÓN	APORTE
Contexto: Local Creación de una Aplicación Backend del Módulo de Transferencia (Cuentas y Montos), con Microservicios GRAPHQL y REST en Contenedores Docker, para Fomentar el Levantamiento de Arquitectura DevOps.	Guitarra desarrolló una aplicación backend para un módulo de transferencias utilizando microservicios GraphQL y REST, desplegados en contenedores Docker, para implementar una arquitectura DevOps, mejorando así la eficiencia de despliegue y operación bajo la normativa ISO/IEC 25010 [7]. El trabajo está estrechamente relacionado proporcionando una base técnica para el despliegue eficiente y seguro de servicio, la diferencia radica en el desarrollo de una solución tecnológica de seguridad y gestión de permisos para una plataforma móvil, integrando prácticas de OWASP para abordar un espectro más amplio de riesgos de seguridad
Contexto: Nacional Desarrollo de un Sistema Web y un Módulo de Seguridad basado en la norma ISO 27000, para mitigar y controlar la integridad de la información.	Escobar y Alarcón desarrollaron un sistema web y un módulo de seguridad basado en la norma ISO 27000, utilizando la metodología espiral para proteger y gestionar la integridad de la información en un contexto financiero, implementando controles y procedimientos conforme a los estándares ISO 27001 [8]. El presente trabajo aborda la seguridad y gestión de permisos en una

	plataforma móvil bajo una arquitectura de microservicios, utilizando las recomendaciones de OWASP.
Contexto: Internacional Designing a Node.js Project Architecture for RESTful Microservices.	Chaplia y Klym diseñaron una arquitectura de proyecto para microservicios RESTful en Node.js, enfocándose en la modularidad y minimización de dependencias a través de la implementación de un modelo de capas y componentes basado en las mejores prácticas [9]. El presente trabajo de igual forma se centra en la arquitectura de proyectos en Node.js para microservicios, pero también aborda la seguridad y gestión de permisos en una plataforma móvil, utilizando las directrices de OWASP para proporcionar una solución de seguridad más integral.
Contexto: Internacional Authentication and Authorization in Microservice-Based Applications	Sänger y Abeck desarrolló una solución de SaaS para la autenticación y autorización en aplicaciones basadas en microservicios, utilizando los productos de integración MuleSoft y la solución IAMaaS de Okta [10]. La contribución con el trabajo es directa, con un enfoque practico para integrar la autenticación y autorización en microservicios.
Contexto: Internacional Autenticación de microservicios basados en OAuth	Rodríguez desarrolló un entorno de microservicios asegurado con OAuth, aplicando recomendaciones de seguridad del documento RFC-6819 para mejorar la autenticación y la autorización [11]. La diferencia con lo existente es que el trabajo presente extiende la autorización con JWT y una gestión de permisos, aplicando las directrices de OWASP Top Ten.

I. CAPÍTULO 1: MARCO TEÓRICO

1.1. Sistema Informático Integrado Universitario

La Universidad Técnica del Norte (UTN) ha implementado el Sistema Informático Integrado Universitario (SIIU), el cual está definido en el artículo 1 del Reglamento de Portafolio Universitario como una colección de recursos. La intención de este software en la comunidad universitaria es evidenciar, potenciar la reflexión, evaluar el rendimiento y fomentar mejora continua de cada una de las prácticas académicas [12].

El sistema está diseñado para asegurar que todos los actores universitarios, es decir, que incluye a rectorado, vicerrectorados, direcciones departamentales, facultades, programas de postgrado, docentes y estudiantes, mantengan actualizados sus portafolios a través de una plataforma web integrada, la cual utiliza repositorios electrónicos para almacenar y gestionar la información necesaria, esto descrito en los artículos 2 y 3 [12].

1.1.1. Importancia del SIIU en la gestión universitaria

El SIIU destaca su importancia fundamental para la gestión universitaria centralizando y optimizando los diversos procesos administrativos y académicos. El SIIU también promueve la transparencia y la toma de decisiones informales. Lo siguiente son aspectos que destacan a la importancia del SIIU en la gestión universitaria [12]:

- *Eficiencia Administrativa*

Varios procesos administrativos rutinarios han sido automatizados, como la gestión de las matrículas, la administración de recursos humanos y el seguimiento. Gracias a la implementación de procesos automatizados se ha logrado una notable disminución en el tiempo y esfuerzo invertido en estas tareas, lo que permite al personal del sector administrativo enfocarse en actividades estratégicas de mayor valor agregado. [13].

- *Mejora en la Toma de Decisiones*

La centralización de datos en el SIIU ha proporcionado a los administradores de la universidad acceso a información precisa y actualizada en tiempo real. Gracias a la centralización se ha facilitado la toma de decisiones basadas en datos sólidos para la planificación estratégica y gestión de la institución.

- *Transparencia y Rendición de Cuentas*

El SIU ha facilitado el seguimiento y la evaluación de las actividades académicas y administrativas, y ha fomentado la transparencia y la rendición de cuentas. El SIU genera, analiza y supervisa el cumplimiento de los objetivos institucionales, lo que ha fortalecido la confianza universitaria y otras partes interesadas en la integridad y eficiencia de la institución [13].

- *Adaptabilidad*

La capacidad de adaptarse rápidamente a nuevas demandas y desafíos en un entorno educativo de constante cambio ha sido de mucha importancia. El SIU ha permitido a la institución de manera efectiva adaptarse a estos cambios, esto se ha logrado gracias a la implementación de tecnologías, actualización de currículos e implementación de nuevas políticas educativas [13].

1.2. Plataformas Móviles

Con el radical avance tecnológico en la parte móvil, a partir del hito importante correspondiente al lanzamiento del iPhone por Apple en 2007, los usuarios han redefinido las expectativas de los usuarios sobre lo que un dispositivo puede ofrecer. Posterior a este evento se ha establecido nuevos estándares para la industria como el caso de Google que surgió con su sistema operativo Android de licencia abierta, fomentando un ecosistema abierto y masivo de aplicaciones disponibles. A partir de esto se ha establecido un enfoque primordial en la creación de valor en la industria del desarrollo de aplicaciones móviles [14].

Una aplicación móvil está diseñada para ejecutarse en smartphones, tablets y otros dispositivos móviles, lo que permite al usuario realizar diversas tareas y actividades [15]. Existen dos tipos de aplicaciones, por un lado, las aplicaciones nativas que son desarrolladas específicamente para una sola plataforma, Android y iOS, utilizando lenguajes específicos de programación nativa. Por otro lado, están las aplicaciones híbridas o multiplataformas, que como su nombre lo indica, combina el desarrollo web con la capacidad de adaptarse al dispositivo de una aplicación nativa [16].

1.2.1. Tendencias actuales en el desarrollo de aplicaciones móviles.

Actualmente el desarrollo de aplicaciones móviles está siendo profundamente influenciado por la necesidad de crear soluciones de gran impacto en diversos sectores. Eleni A. y Paris F. (2023) en su estudio nos revelan como las aplicaciones móviles han emergido como herramientas cruciales, y destacan la importancia de diseñar aplicaciones que incorporen principios de diseño universal, útiles para una amplia gama de usuarios y que fomenten una sociedad más inclusiva y equitativa [17]. Para Zhang, Peng y Chen (2021) el desarrollo de aplicaciones móviles ha mostrado un avance significativo, pues estas herramientas se han integrado en la vida diaria de los ciudadanos pues ofrecen servicios en diversos sectores como transporte, salud y vida cotidiana [18].

1.2.2. Aplicaciones móviles multiplataformas

El desarrollo de aplicaciones nativas para múltiples plataformas ha sido todo un reto debido al consumo considerable de recursos, y debido a que se tenía que duplicar el código para diferentes plataformas lo cual implicaba también un alto aumento de costos del proyecto [19]. Con el fin de responder a esta problemática han surgido diversas herramientas que han permitido el desarrollo una sola vez la aplicación e implementarla en múltiples plataformas. Este tipo de aplicación es denominada aplicaciones multiplataformas, en otras palabras, es una aplicación dirigida a más de una plataforma móvil con el fin de cubrir un gran número de usuarios, y así, aumentar el impacto en el mercado e ingresos [20].

- *Flutter para desarrollo de aplicaciones multiplataformas*

Flutter, creada por Google, es una tecnología híbrida que utiliza el lenguaje Dart y se distingue por su habilidad para desarrollar aplicaciones nativas para Android e iOS de manera rápida y eficiente, gracias a su motor gráfico 2D Skia [16]. Utiliza componentes visuales divididos en módulos principales, Material Design para Android y Cupertino para iOS. Flutter compila todo el código a nativo, e independiza las versiones de Android e iOS. La arquitectura bajo la cual trabaja está basada en componentes estado-dependencias, parecido a React [21].

I. Dart

Dart es un lenguaje de propósito general desarrollada por Google utilizada en una amplia variedad de aplicaciones. Dart puede ser transpilado a JavaScript y frameworks como AngularDart. Este lenguaje es el utilizado para aplicaciones móviles en Flutter, pero también puede ser adecuado para aplicaciones de consola [16], [21].

1.3. Herramientas y Entorno de Desarrollo

1.3.1. Herramientas de control de versiones

En la computación e ingeniería, el control de versiones es crucial debido a la relevancia de esta forma profesional y la capacidad para ayudar a mostrar el trabajo y colaboración eficaz en trabajo colaborativos. Además, facilita la retroalimentación y el seguimiento de problemas ya que ayuda a desarrollar competencias esenciales [22].

Durante el desarrollo de software, el control de versiones resulta ser una parte crucial debido a que contiene todo el historial de desarrollo del proyecto, convirtiéndola en una rica fuente de información [23]. . Utilizar estas herramientas resultan ser eficiente en la gestión del trabajo de equipos en proyectos, y a lo largo del tiempo han surgido diversas herramientas destacadas como Concurrent

Version System (CVS), Revision Control System (RCS), Perforce Helix, BitKeeper, Subversion, , Git y Mercurial [24].

- *Git*

Creado por Linus Torvalds en el 2005, Git ha sido el VCS más utilizado por desarrolladores independientes y empresas en la industria del software, con un amplio apoyo por parte de la comunidad de código abierto [24].

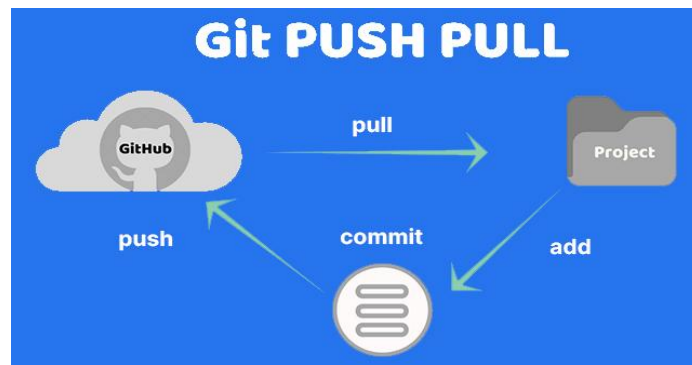


Figura 3. Flujo Básico de Operaciones Git [25]

Git facilita la colaboración desde cualquier ubicación registrando cada versión de un archivo en el repositorio, y permitiéndoles editar sus propias copias de archivos. En *Figura 3* se muestra un flujo básico de las operaciones commit, push y pull de Git. Cada vez que los desarrolladores realizan un commit y push o pull, toman una instancia del archivo para poder registrarlas y gestionar las diferencias [26]. Entre los proveedores más populares para el alojamiento de repositorios Git está GitHub, Bitbucket y GitLab [24].

1.3.2. Entorno de desarrollo integrado (IDE)

Chernestsov, Ivanov y Shevtson (2024) definen un entorno de desarrollo integrado como un conjunto de herramientas de software que pueden apoyar todas las etapas del desarrollo de software, ya que generalmente incluyen un editor de texto, un compilador (o intérprete), una herramienta de automatización de ensamblaje y un depurador [27]. Entre los IDEs populares están Visual Studio, IntelliJ y Eclipse, estos son capaces de soportar varios lenguajes de programación a través de una arquitectura de plugins. La ventaja de esta arquitectura sobre el IDE, es que permite a los desarrolladores adicionar extensiones adicionales específicas a cada lenguaje, e incluso permitirles adicionar funcionalidades como resaltar la sintaxis, reportar errores y advertencias, y el autocompletado, no solo para ahorrar tiempo, sino para verificar la corrección de código rápidamente permitiendo mejorar la productividad [28], [29].

1.3.3. Visual Studio Code

En los últimos años, Visual Studio Code ha tenido un crecimiento en su popularidad [30], esto debido a su alta flexibilidad, gran compatibilidad con diversos lenguajes de programación y de scripts como JavaScript, Typescript, Node.js, HTML, CSS, Python y Java [31], fuerte comunidad y sistema de extensiones. Una característica que gusta a los desarrolladores es que pueden crear sus propias extensiones y usarlas en combinación con la de otros, logran una gran comunidad de código abierto [32].

1.3.4. Bases de datos Oracle

Oracle Database, o también conocido como Oracle DB, es un sistema de gestión de bases de datos relacional (RDBMS) convergente y multimodelo de alta eficiencia y rendimiento, ofrece diversas versiones de bases de datos, incluyendo Oracle Database, bases de datos en memoria, NoSQL y MySQL [33]. Oracle DB es uno de los motores de bases de datos relacionales más confiables y utilizados en todo el mundo, ya que permite a los usuarios acceder directamente a los objetos de datos mediante el lenguaje de consulta estructurado (SQL). Su arquitectura es escalable, y es utilizada por empresas a nivel global gracias a su componente de red propia que facilita las comunicaciones entre redes [34].

1.3.5. Tecnología para el desarrollo.

- *API RESTful*

La arquitectura REST fue propuesta en el año 2000, la cual consiste en una arquitectura de un sistema hipermedia distribuido. La principal abstracción de información es el recurso que puede ser un documento, imagen, un servicio temporal, un conjunto de otros recursos, e incluso un objeto no virtual [35]. Desde la aparición de REST, ha dado lugar a un aumento en el uso de aplicaciones web que utilizan este tipo de arquitecturas en comparación a los servicios tradicionales como SOAP [36].

API RESTful facilita y hace intuitiva la llamada a recursos, reduciendo la complejidad del servicio. La mayoría de las aplicaciones web desarrolladas con arquitectura basada en microservicios utilizan API RESTful para entregar mensajes [35]. Los desarrolladores de aplicaciones web suelen preferir implementar API REST y JSON como formato de intercambio principal [37]. En conclusión, las APIs REST, también conocidos como API Web, implican solicitudes a recursos almacenados en un servidor y respuestas por parte del servidor con la información solicitada como se puede observar en *Figura 4*, y el protocolo utilizado para transportar datos es HTTP (Hypertext Transfer Protocol) [38]. Los datos se exponen mediante puntos finales de lectura como GET, y de escritura como POST, PUT y DELETE [37], [39].

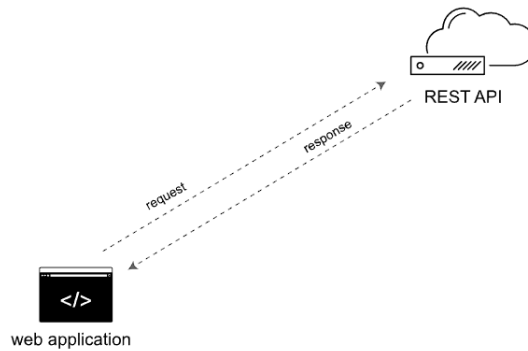


Figura 4. Comunicación cliente-servidor mediante REST API [38]

- *Tecnología de desarrollo backend*

La elección de la tecnología de desarrollo fue llevada a cabo tras una evaluación rigurosa documentada en una Revisión Sistemática de Literatura (SRL). Los hallazgos demostraron una combinación de 4 tecnologías (lenguaje de programación + framework) para realizar un experimento comparativo en el desarrollo de API Rest. Estas 4 combinaciones tecnológicas fueron:

- I. SQL con APEX
- II. TypeScript con NestJs
- III. Java con Spring Boot
- IV. Python con Flask

El experimento consistía en realizar una serie de consultas a la base de datos (Oracle), aumentando progresivamente a complejidad desde 1 a 4 niveles de profundidad, y considerando el volumen de datos que pueden arrojar en determinado tiempo. Los resultados obtenidos se pueden observar en **Figura 5***Error! Reference source not found.*.

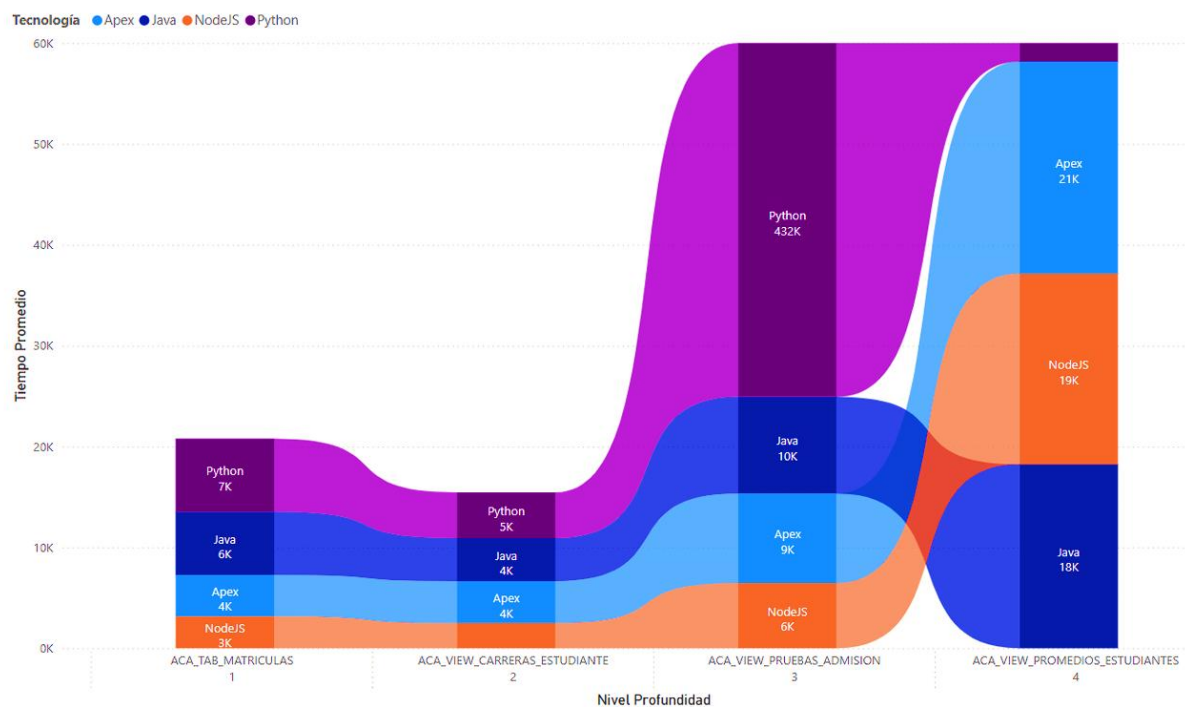


Figura 5. Resultados del experimento de la comparación de tecnologías para desarrollo de APIs Rest

Los resultados demostraron que TypeScript con NestJs tuvo un rendimiento superior, en términos de velocidad de ejecución mientras se aumentaba progresivamente la complejidad. Con estos resultados se posicionó a NestJs como tecnología para el desarrollo del proyecto UTN Móvil.

- *Node.JS*

En los últimos años, Javascript ha pasado a ser uno de los lenguajes de mayor popularidad para ser implementado en aplicaciones web del lado del servidor [40]. El factor determinante de esta tendencia es debido a Node.JS, que consiste en un entorno de tiempo de ejecución multiplataforma del lado del servidor, o comúnmente denominado backend, que ejecuta código Javascript [41]. Node.JS lleva un rico exosistema de bibliotecas y herramientas como el Node Package Manager (NPM) que se ha convertido en un recurso indispensable. NPM puede integrarse fácilmente para agilizar el desarrollo y mejorar la productividad, todo esto es gracias al acceso a una amplia gama de paquetes de código abierto [42].

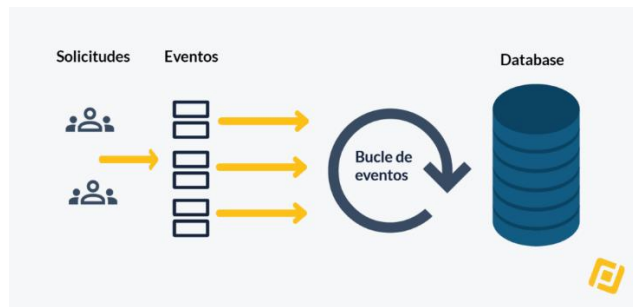


Figura 6. Arquitectura de Node.JS [43]

La arquitectura de Node.JS está basada en un bucle de eventos de un solo hilo, así como se observa en *Figura 6*, esto le permite manejar múltiples clientes concurrentes de manera eficiente, superando a otras plataformas del lado del servidor como Java o PHP [44].

- *NestJS*

Con el crecimiento exponencial de NodeJS como entorno de ejecución de aplicaciones backend, ha surgido la necesidad de contar con un framework que aproveche al máximo el rendimiento. NestJS es una solución avanzada desarrollada con NodeJS y TypeScript centrada en la creación de aplicaciones escalables y altamente escalables. NestJS presenta una arquitectura limpia modular inspirada en Angular que se basa en la reutilización de servicios y módulos, lo que facilita la adopción del patrón y los principios SOLID [45]. NestJS se caracteriza por un sistema de inyección de dependencias que permiten el desacoplamiento de componentes de forma elegante, generando código limpio, comprobable y de fácil mantenimiento. La modularidad y separación de responsabilidades que tiene el framework NestJS se asocia con la precisión de beneficios sustanciales sobre las alternativas más tradicionales, particularmente en aquellas situaciones donde la simplicidad, la claridad y el rendimiento son cruciales [46].

1.4. Metodología de Desarrollo de Software

El desarrollo de software es un trabajo cooperativo, las tareas, distribuidas entre distintos equipos, pueden ser gestionadas y ordenadas en base a distintos criterios. La necesidad de coordinación entre el desarrollo de estas tareas es imprescindible pues estas son desarrolladas de forma paralela, y en ocasiones hay ciertas tareas que no pueden iniciarse sin que otra haya finalizado [47]. Existen varios enfoques tradicionales de desarrollo de software como en cascada, en espiral, evolutivo, entre otros [48]. Estos enfoques resultan especialmente útiles cuando el desarrollo de software es extenso y complejo, ya que contribuyen a eliminar métodos informales y obsoletos de desarrollo, con el objetivo principal de entregar un software de calidad dentro de un plazo establecido y cumpliendo con los requisitos del usuario [47], [49].

El problema con respecto a usar un enfoque tradicional es que suele ser un proceso muy pesado, por lo que se enfocan en abordar problemas específicos e importantes en el mantenimiento y cambios en función de los requerimientos del usuario. En función a esto surgieron métodos de desarrollo livianos que aceleran el desarrollo y responden efectivamente a los cambios solicitudes durante el desarrollo del software [47]. Generalmente los directores del proyecto suelen seleccionar el enfoque para desarrollar el software, a través de su experiencia ya que no existe una base lógica para una selección del todo correcta [50].

1.4.1. Metodología de Desarrollo Ágil SCRUM

Las metodologías de desarrollo ágil han sido elogiadas con frecuencia debido a sus resultados beneficios al defender los procesos iterativos, la adaptabilidad y un enfoque centrado en las partes interesadas [51]. La popularidad que se ha ganado es debido a sus beneficios como mejor cumplimiento de requisitos, menor tiempo de comercialización y mayor productividad [52]. Entre las metodologías que más son adoptadas por equipos de desarrollo de software son SCRUM, Extreme Programming, pero también existen otras alternativas preferidas, como Lean Software Development y Kanban [53].

La metodología SCRUM ha sido, en los últimos años, la más admirada por varios equipos de desarrollo de software con un 89% de equipos que han optado por el desarrollo ágil, y de igual forma en otras encuestas se afirmó que SCRUM tiene una mayor tasa de aceptación sobre otras metodologías ágiles de desarrollo [54].

SCRUM se fundamenta en tres pilares esenciales: transparencia, inspección y adaptación, los cuales permiten trabajar de forma iterativa para ajustar los procesos según sea necesario a lo largo de sus tres fases [55]. Los valores que respaldan esta metodología bajo entornos de experimentación son coraje, enfoque, compromiso, respeto y apertura. La estructura bajo la cual trabaja esta metodología es en Sprints, ciclos en donde el entregable es de alto valor y fomenta la retroalimentación continua, el equipo Scrum, compuesto por Scrum Master, desarrolladores y Product Owner, quienes trabajan en conjunto con los stakeholders para revisar y ajustar el entregable de cada sprint, que comúnmente se realiza en menos de un mes [56].

1.4.2. Fases de SCRUM

- *Fase I: Pre-Juego*

En la fase de pre-juego se define la visión del proyecto la cual puede irse perfeccionando posteriormente. Aquí se prepara el Product Backlog, se estima los tiempos y costos, fechas de entrega

y sprints previstos. Además, durante esta etapa, se evalúa los riesgos, la validación de herramientas de desarrollo y recursos, y el equipo de trabajo [57].

i. **Artefactos de SCRUM**

Scrum utiliza tres artefactos clave que representan valor: Backlog del producto, Backlog del sprint e Incremento del producto [58].

- a. **Product Backlog:** Consiste en una lista dinámica y continuamente actualizada de funcionalidades, mejoras, correcciones y tareas necesarias para desarrollar el producto. El propietario es quien prioriza y organiza estos elementos.
- b. **Sprint Backlog:** Es un conjunto específico de tareas derivadas del Product Backlog para realizarse durante un sprint determinado. Se actualiza durante la planificación del sprint y orienta el trabajo inmediato de desarrollo y diseño.
- c. **Incremento:** Es el resultado concreto y entregable del trabajo realizado en cada sprint, acumulando mejoras de todos los sprints anteriores.

ii. **Roles de SCRUM**

Scrum define tres roles clave [57]:

- a. **Product Owner:** Representa al cliente, siendo responsable del producto. Define, prioriza y adapta el Product Backlog según necesidades cambiantes del negocio. Establece cronogramas, gestiona financiamiento y negocia con el equipo Scrum para satisfacer intereses de todos los involucrados. Debe comprender claramente aspectos de negocios, ingeniería y marketing, además de tener sólidas habilidades de comunicación.
- b. **Scrum Master:** Facilita y supervisa la aplicación adecuada de las prácticas Scrum. Es quien protege al equipo de interferencias externas, asegurando un entorno óptimo para el trabajo. Dirige reuniones diarias (Daily Scrum) y una retrospectiva al finalizar cada sprint para analizar experiencias y mejorar continuamente [55].
- c. **Equipo Scrum:** Grupo autoorganizado sin roles específicos asignados. Distribuyen las tareas según sus intereses y habilidades. Son responsables del diseño, desarrollo, pruebas y documentación del producto, entregando resultados funcionales al término de cada sprint [55].

- **Fase II: Juego**

Es la fase principal de desarrollo en Scrum, donde se trabaja en iteraciones breves llamadas sprints, que duran entre una y cuatro semanas según la complejidad del proyecto [57].

i. Eventos de SCRUM

Scrum trabaja mediante sprints, periodos cortos con tiempo definido en los que se desarrolla software funcional según las prioridades del Product Backlog. Cada sprint incluye estas actividades clave [57]:

- Sprint Planning:** Se realiza en dos partes. Primero, el Product Owner y el Scrum Master revisan y seleccionan tareas prioritarias del backlog, estableciendo objetivos claros para el equipo. Luego, el equipo define cómo ejecutar esas tareas y se compromete a realizarlas dentro del sprint.
- Daily Scrum:** Consiste en una reunión diaria breve (10-15 minutos), donde cada miembro responde tres preguntas esenciales: ¿qué hizo ayer?, ¿qué hará hoy? y ¿qué dificultades enfrenta?. El objetivo de la reunión es permitir monitorear los avances y resolver obstáculos rápidamente.
- Sprint Development:** El equipo realiza actividades como diseño, desarrollo y pruebas, implementando las tareas según la prioridad asignada por el Product Owner.
- Sprint Review:** Al finalizar cada sprint, se revisa y evalúa el producto desarrollado con el Product Owner para verificar que cumpla con las necesidades. Aquí se obtiene retroalimentación valiosa que puede modificar el rumbo del desarrollo.
- Sprint Retrospective:** Aquí el equipo analiza lo que funcionó bien y lo que debe mejorar en su proceso de trabajo, definiendo acciones concretas para optimizar el siguiente sprint.

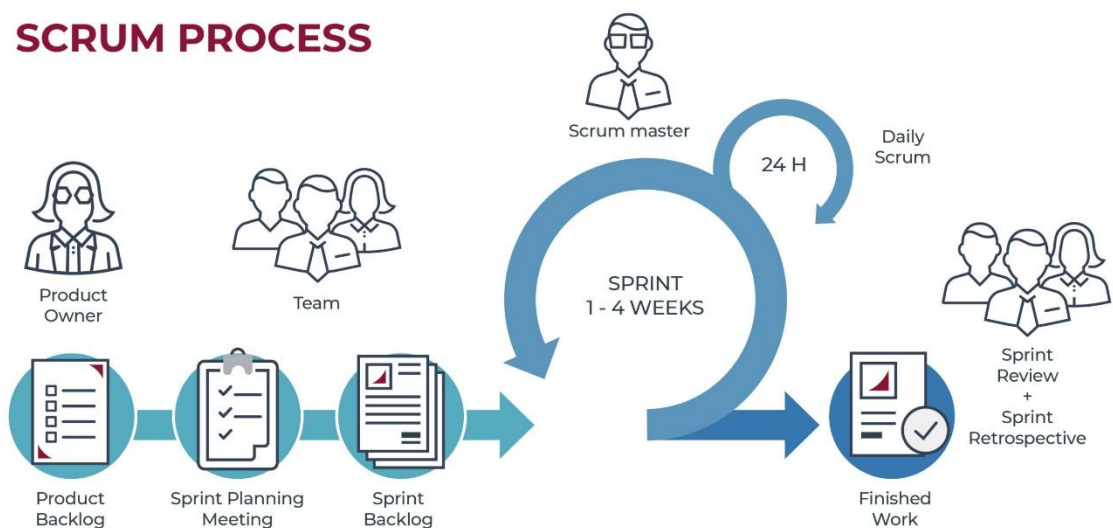


Figura 7. Proceso de SCRUM [59]

De forma conjunta, estos eventos, roles y artefactos, mencionados en el punto anterior, son el núcleo de SCRUM como se puede observar en *Figura 7*.

- *Fase III: Post-Juego*

Esta es la etapa de cierre del proyecto Scrum. Después de desarrollar las funcionalidades deseadas, se realiza la entrega final del producto. Se considera finalizada esta fase cuando se cumplen los objetivos definidos en la fase inicial (Pre-Juego). Aquí se realizan pruebas finales de integración, y se elaboran material de apoyo como manuales de usuario y materiales capacitación para la entrega definitiva.

1.5. Arquitectura de Microservicios

La arquitectura basada en microservicios son un tipo de sistemas de software basados en una arquitectura distribuida, la cual está compuesta por diversos servicios que trabajan de manera independiente y autónoma [60], como se muestra en *Figura 8*. En la actualidad, muchos sistemas están divididos en servicios acoplados para mejorar la mantenibilidad, escalabilidad y tolerancia a fallos [61].

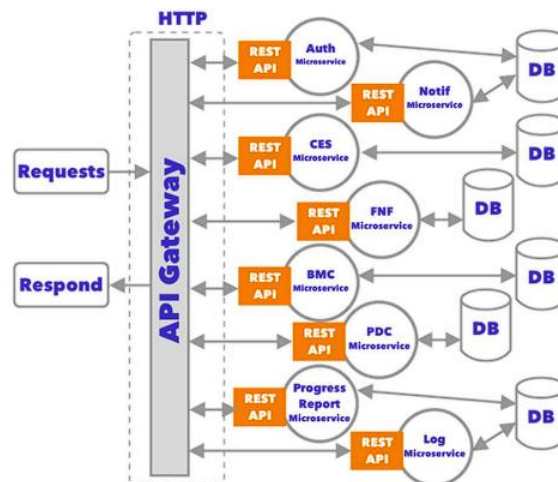


Figura 8. Arquitectura de microservicios [62]

Los desarrolladores deben formarse profesionalmente en una amplia pila de tecnologías para infraestructuras, desarrollo, operaciones de software y respectivos dominios comerciales [63]. Pero pese a que la curva de aprendizaje sea extensa, muchos de los desarrolladores no son especialistas, pero tienen competencias que complementan el dominio de las tecnologías y una perspectiva integral de roles [64].

1.5.1. Ventaja de los microservicios

El diseño de esta arquitectura esta intencionalmente orientado para ejecutar funciones altamente independientes, orientada a servicios e implementación mapeada [65], [66]. Los microservicios ejecutan procesos propios conectados entre sí, utilizando protocolos de comunicación ligera como API REST [67]. Y, además, el grado de centralización es limitado, con la ventaja de que pueden usar distintos lenguajes de programación y tecnologías de almacenamiento de datos [62].

1.5.2. Patrones de diseño en microservicios

Aplicar patrones de diseño desempeña un papel importante en la calidad al diseñar software. Esta aplicación es una solución reutilizable y comprobada por el mundo real, afectando directamente en los atributos de calidad y el desarrollo de sistemas de microservicios [68].

Desde la aparición de este enfoque de arquitectura microservicios, han aparecido diversos patrones orientados a varios aspectos, entre algunos, destacados por su relevancia, podemos encontrar los siguientes patrones: CQRS, BFF, Strangler, Circuit Breaker, Database per Microservice, API Gateway, Consumer-Driven Contract Tracing y External Configuration [69], [70].

- *API Gateway*

API Gateway utiliza puertas de enlace (API Gateways) como un proxy inverso, el cual estaría encargado de redirigir las llamadas a los servicios apropiados, devolviendo el resultado consiguiente. Algunas otras características complementarias brindadas por este patrón son autenticación, limitación de tasas y análisis sobre el uso del servicio. Este patrón contiene un solo defecto que consiste en que, si el Gateway deja de funcionar, todo el sistema dejará de funcionar hasta que se solventa el problema [71].

1.6. Seguridad informática y seguridad de la información

En las ciencias de la computación, la seguridad informática, o ciberseguridad, es una de las pautas muy importantes en el plan de estudio, puesto que se ha convertido en una disciplina central de la dependencia mundial de la tecnología de la información [72]. La seguridad informática comprende la seguridad de varios tipos diferentes de elementos, e informes han mostrado un aumento de problemas recurrentes de manera frecuente sin un enfoque claro de cómo resolverlos [73]. Las estrategias y perspectivas de seguridad informática difieren entre país, utilizando enfoques como el de protegerse contra las amenazas del ciberespacio, o en el que las propiedades de seguridad deben salvaguardarse y garantizarse [74]. La Unión Internacional de Telecomunicaciones (UIT) define la seguridad informática como el conjunto de herramientas, directrices y acciones empleadas para proteger el entorno cibernético [75].

Por otro, la seguridad de la información se relaciona fuertemente con las medidas preventivas enfocadas a salvaguardar y proteger la información, adoptando los principios de confidencialidad, disponibilidad e integridad [75], [76]. La relación de estos principios se muestra en *Figura 9*.

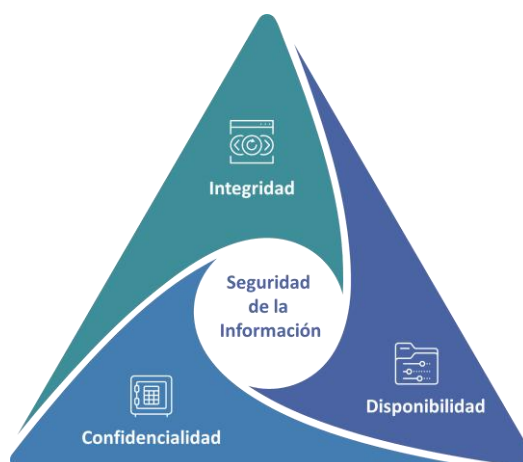


Figura 9. *Triángulo CIA* [77]

1.6.1. Principios fundamentales de seguridad de la información.

La seguridad de la información se asegura mediante la implementación de medidas que protegen y definen los datos, garantizando su disponibilidad, integridad, autenticación, confidencialidad y no repudio [78], [79]. Con el fin de conllevar estos principios, el Instituto Nacional de Normas y Tecnología (NIST) confirma que debemos garantizar los principios de la triada de la CIA con el objetivo de asegurar la información [78]. A continuación, se explica cada uno de estos principios según [79]:

- *Confidencialidad*

Consiste en el acceso restringido a cualquier tipo de usuario sin autorización, es decir que no podrá acceder a los datos o información mientras no tenga los permisos establecidos.

- *Integridad*

Este principio consiste en el almacenamiento o transferencia de esta sin ningún tipo de modificación, y mucho menos destruido. A este principio se le extiende también el no repudio y autenticidad.

- *Disponibilidad*

La finalidad de este principio es que, si el usuario autorizado requiera los datos o información, estará disponible sin ningún inconveniente.

1.6.2. Amenazas y vulnerabilidades comunes

Tras el desarrollo de una Revisión Sistemática de la Literatura (SRL) por parte de [80] enumeramos las amenazas y vulnerabilidades de seguridad cibernética que se identificaron tras un estudio de mapeo sistemático:

- Denegación de servicios (DoS)
- Malware
- Phishing
- Ataques de inyección SQL
- Secuestros de sesiones y ataques de intermediario
- Reutilización de credenciales
- Secuencia de comandos entre sitios (XSS)

El mismo trabajo nos menciona que las víctimas claves en su mayor parte son del grupo organizativo y posterior los usuarios comunes. Y de igual forma hay que mencionar que los resultados de la RQ5 han mostrado que el ciberespacio y el entorno de la nube son los principales objetivos de la ciberdelincuencia [80].

1.7. Seguridad en sistemas microservicios.

La adopción de sistemas basados en microservicios ha ofrecido varias ventajas tanto a desarrolladores como a empresas, pero también ha conllevado a nuevos retos, el principal de ellos es la seguridad debido a la prestación de atención a cada uno de los servicios desarrollados[81] [82]. El incremento de ataque a estos tipos de sistemas es debido a su granularidad y heterogeneidad, lo que conlleva a mecanismos totalmente robustos de autenticación y autorización, con el fin de proteger la comunicación entre servicios [83].

1.7.1. Autenticación y autorización en sistemas microservicios.

Independientemente de arquitectura implementada al sistema desarrollado, la autenticación y autorización son relevantes para la seguridad, considerándoles elementos claves en sistemas basados en microservicios [81], [84]. Es vital que, durante la fase de diseño y utilización de patrones de arquitecturas, se implemente la autenticación y autorización en los sistemas [85]. Podemos definir estos dos aspectos claves de la siguiente forma [81], [84]:

- *Autenticación*

La autenticación es el proceso en el que se determina si el usuario es quien dice ser, es decir comprobar si la identidad del usuario.

- *Autorización*

La autorización consiste en el proceso permitir, es decir dar permiso, al usuario para realizar acciones específicas o acceder a recursos determinados.

1.7.2. Modelos de autorización

Los modelos de control de acceso (autorización) proporcionan una estructura y permisos a usuarios, es decir que actúa como un marco que establece el control sobre los recursos del sistema por parte del usuario [86]. Para sistemas simples la autenticación es suficiente, pero en nuestro caso, al ser un sistema basado en microservicios, podemos utilizar modelos de autorización como los siguientes [87]:

- Uso de roles
 - I. Control de Acceso Basado en Roles (RBAC)
- Uso de roles y permisos
 - I. Control de Acceso Basado en Roles (RBAC) + Permisos
- Uso de atributos para el control de acceso
 - I. Control de Acceso Basado en Atributos (ABAC)
- En función de sus relaciones
 - I. Control de Acceso basado en Relaciones (ReBAC)

Tabla 2. Comparativa entre modelos de autorización aplicables en microservicios

CRITERIO	RBAC	ABAC	PBAC / ReBAC
Tipo de control	Basado en roles	Basado en atributos	Basado en relaciones y permisos
Escalabilidad	Media	Alta	Alta
Flexibilidad	Limitada	Alta	Alta
Complejidad de implementación	Baja	Alta	Media
Adecuado para microservicios	Sí	Sí (con control granular)	Sí (ideal para relaciones jerárquicas)
Ejemplo de uso	Usuario con rol <i>docente</i>	Usuario con <i>rol + localización</i>	Usuario con permisos sobre recurso compartido

Después de un comparativa como está expuesto en Tabla 2, se optará por RBAC extendido con asignación dinámica de permisos ya que permite escalar y controlar el acceso en una plataforma móvil institucional sin requerir un motor de políticas complejo como en ABAC puro.

1.7.3. Autorización basada en roles (RBAC)

Este modelo consiste en agrupar a los usuarios en función de sus responsabilidades. El enfoque ofrecido por este modelo es sencillo y manejable para administrar el acceso en función a la responsabilidad, o rol, del usuario [87]. Este enfoque también ha llegado a ser el preferido debido a la sobrecarga mínima de permisos asignados por usuarios, los cuales se analizan y actualizan con frecuencia. La idea de RBAC gira en torno a la creación de roles, a los que también se le puede incorporar cierto conjunto de permisos [86], como se muestra en *Figura 10*.

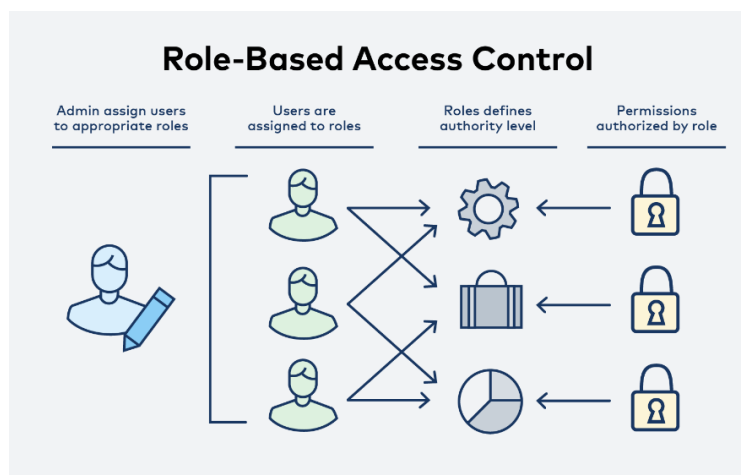


Figura 10. Control de Acceso Basado en Roles [88]

Un rol está definido como una colección de operaciones que pueden ser ejecutadas sobre ciertos recursos del sistema [87]. En un rol se especifican las tareas asignadas a los diferentes usuarios, definiendo de esta manera quienes pueden acceder a ciertos recursos del sistema [89].

1.7.4. Tecnologías y protocolos de autenticación y autorización.

Entre las principales soluciones tecnológicas que se pueden implementar para abordar los retos de la autenticación, y por consiguiente autorización, en sistemas basados en microservicios están OAuth 2.0, JWT, API Gateway, OpenID Connect y la estrategia de Single Sing-on (SSO) [81].

Tabla 3. Comparativa de mecanismos de autenticación/autorización para microservicios

TECNOLOGÍA	TIPO	FORTALEZAS	LIMITACIONES	APLICABILIDAD EN PROYECTO
------------	------	------------	--------------	---------------------------

JWT (Json Web Token)	Token basado (stateless)	Ligero, seguro, fácil de usar en API Rest y portable.	No se puede revocar sin lista negra. Si se compromete es peligroso.	Si
OAuth 2.0	Delegación de acceso	Amplio estándar industrial, útil para apps de terceros y acceso granular.	Complejidad de configuración. Requiere servidor de autorización externo.	No
API Gateway	Proxy inverso + seguridad	Control centralizado de acceso, rutas protegidas y gestión de tokens.	Punto único de falla si no se asegura disponibilidad y redundancia.	Si
OpenID Connect	Identidad federada sobre OAuth	Añade identidad al protocolo OAuth Ideal para SSO moderno	Depende de proveedores externos o compleja integración propia.	No
SSO (Single Sing-On)	Inicio de sesión único	Mejora UX al permitir una sola autenticación para múltiples apps.	Implementación robusta y segura. Requiere servidores y protocolos adicionales.	No
Ejemplo de uso	Usuario con rol <i>docente</i>	Usuario con <i>rol + localización</i>	Usuario con permisos sobre	

- *OAuth 2.0*

OAuth 2.0 es el protocolo estándar de la industria para la autorización. Proporciona métodos específicos de autorización para aplicaciones web, de escritorio, móviles y dispositivos de sala de estar. Este protocolo permite a las aplicaciones obtener un acceso limitado a los recursos del usuario sin exponer sus credenciales. OAuth 2.0 gestiona las sesiones de los usuarios de manera segura y eficiente mediante el uso de tokens de acceso y actualización [90].

- *JSON Web Token (JWT)*

JSON Web Token es un estándar abierto (RFC 7519) que define un método compacto y autónomo para la transmisión segura de información entre partes mediante JSON. Comúnmente se firman para garantizar la integridad de los datos. Es información verificable y confiable gracias a su firma digital, que puede generarse mediante un secreto mediante el algoritmo HMAC, o con un par de claves pública/privada usando RSA o ECDSA. Esta tecnología es usada en situaciones de autorización como el inicio de sesión único (SSO), permitiendo el acceso a recursos y servicios a usuarios sin necesidad de autenticarse repetidamente [91]. Un JWT está compuesto por el encabezado, el payload y la firma, cada uno codificado en Base64Url y separados por puntos, creando una estructura compacta y fácil de transmitir en entornos HTML y HTTP [92].

1.8. Recomendaciones OWASP Top Ten

OWASP (Open Web Application Security Project) es una Fundación sin fines lucrativos, su objetivo es el de mejorar la seguridad del software. Fue fundada en diciembre de 2001, y constituida de forma oficial como organización benéfica en abril de 2004. Dentro de su programa se incluye proyectos de código abierto liberados por la comunidad, más de 250 capítulos locales, miles de miembros y capacitaciones líderes en la industria. La comunidad abierta de OWASP permite a las organizaciones desarrollar y mantener software seguro, brindando de forma gratuita sus proyectos, herramientas, documentos, foros y capítulos a quienes estén interesados en el tema de seguridad con respecto al software [93].

1.8.1. Directrices de OWASP Top Ten 2021

OWASP Top Ten consiste en un estándar de concientización para desarrolladores a nivel mundial, que tiene un enfoque en la seguridad de aplicaciones web. Este estándar es reconocido a nivel mundial

como el primer paso para una codificación más segura. Las organizaciones a nivel mundial adoptan estas directrices para minimizar los riesgos sus aplicaciones y fomentar una cultura de desarrollo de software seguro. Para el 2021, en el Top 10 se agregaron tres nuevas categorías, cuatro con cambios de nombre y alcance, y algunas establecidas de forma sólida, todo esto con el fin de servir como una guía para mejorar las prácticas de seguridad en el desarrollo de software. Estas diez directrices tomadas de [94] son descritas a continuación en Tabla 4 junto a su medida aplicada:

Tabla 4. Alineación de OWASP Top Ten 2021 con mecanismos implementados en UTN Móvil

CATEGORÍA OWASP 2021	MEDIDA APLICADA EN UTN MÓVIL
A01 – Broken Access Control	Implementación de control de acceso basado en roles (RBAC) dinámico, con validación de permisos embebida en los JWT.
A02 – Cryptographic Failures	Uso de HTTPS para todas las comunicaciones; tokens JWT firmados con HMAC (HS256) y RSA según el contexto
A03 – Injection	Validación estricta de entrada mediante class-validator en DTOs; uso seguro de consultas parametrizadas con TypeORM.
A04 – Insecure Design	Arquitectura modular con separación de responsabilidades; principios KISS y SOLID aplicados desde el diseño.
A05 – Security Misconfiguration	Uso de variables de entorno (.env), protección de rutas con <i>guards</i> personalizados en NestJS y políticas de CORS seguras.
A06 – Vulnerable and Outdated Components	Auditoría continua de dependencias mediante npm audit; actualización regular en entornos de staging y producción.

A07 – Identification and Authentication Failures	Autenticación robusta basada en JWT, con verificación MFA vía correo institucional y autenticación biométrica en apps móviles.
A08 – Software and Data Integrity Failures	Validación de integridad de datos con DTOs tipados y tokens firmados digitalmente; control de versiones en CI/CD.
A09 – Security Logging and Monitoring Failures	Registro detallado de sesiones activas, trazabilidad de acciones críticas y auditoría persistente en base de datos.
A10 – Server-Side Request Forgery (SSRF)	Eliminación de exposición directa a recursos internos; filtrado de rutas y acceso restringido a través de un gateway proxy.

1.9. Resultados a partir del estudio

Durante este capítulo se han identificado y fundamentado los pilares conceptuales que respalden las decisiones tecnológicas adoptadas para el diseño y desarrollo de la solución tecnológica de autenticación y autorización para la plataforma móvil de la Universidad Técnica del Norte (UTN).

La implementación del Sistema Informático Integrado Universitario (SIIU) ha demostrado su importancia estratégica en la gestión universitaria, permitiendo automatizar procesos críticos. Entender la importancia de la implementación fue esencial para definir las exigencias de seguridad que debían cumplir la solución tecnológica propuesta, sobre todo al considerar la sensibilidad de los datos gestionados por los portafolios docente y estudiantil.

El estudio de las plataformas móviles y sus tendencias en la actualidad permitió evidenciar la necesidad de adoptar tecnologías multiplataforma que garanticen eficiencia, mantenibilidad y alcance.

El framework Flutter fue seleccionado debido a su madurez, soporte por la comunidad, velocidad de desarrollo y capacidad de desplegar interfaces modernas para Android e iOS a partir de una sola base de código con el lenguaje de programación Dart. La estructura modular y el enfoque en componentes reutilizables de Flutter se ajustan perfectamente con la arquitectura limpia adoptada en esta investigación.

Para lo que corresponde al control de versión de código se identificó la relevancia de Git, así como Visual Studio Code como entorno de desarrollo integrado. La combinación de estas tecnologías permite la optimización de la productividad durante el desarrollo, facilitar la trazabilidad del avance del proyecto, la colaboración efectiva y la gestión eficiente de versiones y pruebas.

Por otro lado, destacamos la importancia de aplicar los pilares de confidencialidad, integridad y disponibilidad (triada CIA) para incorporar medidas preventivas basadas en estándares internacionales y normativas. De igual forma, el estudio de mecanismo de autenticación y autorización en arquitecturas microservicios permitió justificar de manera rigurosa el uso de tecnologías como JWT (JSON Web Token), autenticación multifactor (MFA), control de acceso basado en roles (RBAC) y validación biométrica. Por ejemplo, la elección de JWT se fundamenta por su facilidad de implementación en APIs REST y su compatibilidad con arquitecturas sin estado, mientras que la inclusión de MFA responde a las exigencias actuales de proteger los sistemas ante accesos no autorizados aun cuando las credenciales principales hayan sido comprometidas. El modelo RBAC extendido con asignación dinámica de permisos resulta más apropiado para un entorno institucional como el de la UTN, donde las jerarquías y funciones de usuario están claramente definidas, pero se requiere cierto grado de personalización por módulo o servicio.

Y finalmente, un eje transversal a todo el marco teórico ha sido la alineación con las recomendaciones de seguridad propuestas por OWASP Top Ten 2021, en donde cada uno de los riesgos más relevantes definidos por OWASP fue contrastado con las soluciones propuestas para el proyecto.

II. CAPÍTULO 2: DESARROLLO

En este capítulo se detalló el proceso de desarrollo de la solución tecnológica, siguiendo las fases establecidas por la metodología SCRUM. Estas fases permitieron organizar y gestionar el proyecto de forma iterativa e incremental, lo que facilitó la adaptación continua a los cambios y asegurando entregables funcionales en cada ciclo. A continuación, en Figura 11, se muestra la descripción de estas fases principales de SCRUM.

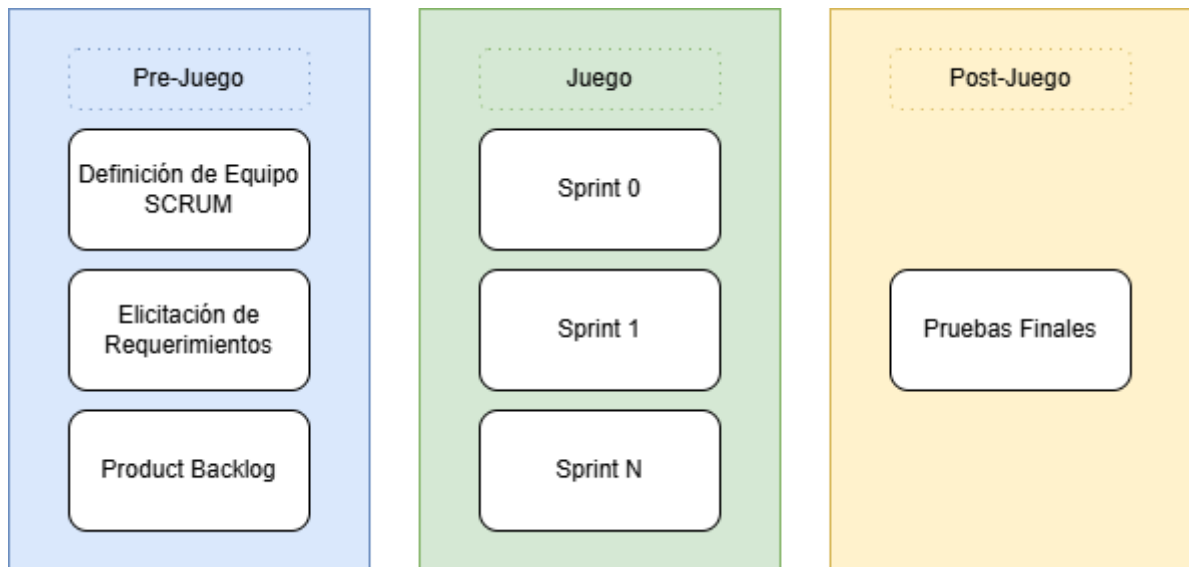


Figura 11. Diagrama de Fases de Desarrollo

- Pre-juego: Se definió el equipo SCRUM, se realizó la elicitación de requerimientos y se construyó el Product Backlog, lo cual resultó crucial para establecer las bases del proyecto, y permitió una planificación estructurada de los sprints.
- Juego: Se lleva a cabo el inicio del sprint 0 en donde se desarrolla los preparativos iniciales como la planificación, entornos de desarrollo y definición de arquitectura. Los siguientes sprints se centralizan en el desarrollo y entrega de incrementos de funcionalidades en base al ciclo de planificación, ejecución y revisión.
- Post-juego: Aquí se realizan las pruebas finales, se validan todos los entregables asegurándose que se cumpla los requisitos definidos anteriormente.

2.1. Desarrollo de Fase II: Pre-Juego

2.1.1. Equipo SCRUM

En la *Tabla 5* se describió la conformación y roles del equipo Scrum encargado del desarrollo e implementación del servicio de autenticación y autorización para la plataforma móvil de la Universidad Técnica del Norte.

Tabla 5. *Equipo SCRUM del proyecto*

NOMBRE	ROL	CARGO
José Antonio Quiña Mera	Product Owner	Director de proyecto UTN Móvil Docente UTN
Erick Sebastian Carlosama Tabango	Scrum Master Equipo de Desarrollo	Tesista
Dirección de Desarrollo Tecnológico e Informático (DDTI)	Stakeholder	Departamento de Tecnología de la Información de la Universidad Técnica del Norte
Marco Remigio Pusedá Chulde	Stakeholder	Director de Trabajo de Grado Docente UTN

2.1.2. Elicitación de Requerimientos

La obtención de los requisitos se realizó mediante historias de usuarios, conforme a los lineamientos establecidos por la metodología SCRUM. Para este propósito, se llevaron a cabo reuniones previas entre Scrum Master y la Dirección de Desarrollo Tecnológico e Informático (DDTI) de la Universidad Técnica del Norte. En estas sesiones se revisaron detalladamente todos los aspectos necesarios para asegurar una correcta implementación del módulo de autenticación y autorización, y demás ajustes en el entorno móvil.

A continuación, se detallan las historias de usuario que se obtuvieron durante estas sesiones:

Tabla 6. HU-SEC-01: Inicio de sesión

Código: HU-SEC-01	Título: Inicio de sesión	Prioridad: Alta
Descripción:	Como estudiante o docente registrado, quiero iniciar sesión en la aplicación móvil utilizando mis credenciales para acceder a mi perfil y gestionar mis datos académicos.	
Criterios de aceptación:	CA1: El servicio debe verificar que el usuario existe y que las credenciales son correctas.	
	CA2: Las contraseñas deben enviarse codificadas para asegurar el envío de todos los caracteres.	
	CA3: Al iniciar sesión se debe generar un token de acceso seguro con expiración controlada y un token de refresco con expiración prolongada y controlada.	

Tabla 7. HU-SEC-02: Cambio de contraseña de Microsoft

Código: HU-SEC-02	Título: Cambio de contraseña de Microsoft	Prioridad: Media
Descripción:	Como usuario autenticado con credenciales de Microsoft me gustaría ser redirigido a la plataforma de cambio de contraseña de Microsoft para poder cambiar mi contraseña de manera segura.	
Criterios de aceptación:	CA1: Debe haber un enlace en la aplicación que redirija a la página de cambio de contraseña de Microsoft.	
	CA2: La redirección debe abrirse en el navegador predeterminado del dispositivo.	

Tabla 8. HU-SEC-03: Refresco automático del token de acceso

Código: HU-SEC-03	Título: Refresco automático del token de acceso	Prioridad: Alta
Descripción:	Como usuario autenticado, quiero que mi token de acceso se renueve automáticamente sin necesidad de volver a iniciar sesión para evitar interrupciones.	
Criterios de aceptación:	CA1: El servicio de seguridad debe generar un nuevo token para reemplazar al que expiró si el usuario está usando la aplicación.	
	CA2: La renovación del token debe realizarse en segundo plano.	
	CA3: Si el token de refresco expira, el token de acceso no podrá volver a refrescarse y la sesión caducará, solicitando que el usuario nuevamente se autentique con sus credenciales.	

Tabla 9. HU-SEC-04: Manejo de tokens expirados o inválidos

Código: HU-SEC-04	Título: Manejo de tokens expirados o inválidos	Prioridad: Alta
Descripción:	Como usuario, quiero que la aplicación detecte si mi token de autenticación ha expirado o es inválido, y me redirija sin mostrar errores técnicos para asegurar una experiencia estable y segura.	
Criterios de aceptación:	CA1: La aplicación debe interceptar errores de tipo 401 o inválidos en todas las llamadas protegidas.	
	CA2: Si el token es inválido o ha expirado, debe eliminarse del almacenamiento seguro.	
	CA3: El usuario debe ser redirigido automáticamente a la pantalla de inicio de sesión, indicando un mensaje claro del motivo.	

Tabla 10. HU-SEC-05: Control de múltiples sesiones por usuario

Código: HU-SEC-05	Título: Control de múltiples sesiones por usuario	Prioridad: Alta
Descripción:	Como usuario, quiero mantener hasta tres sesiones activas en diferentes dispositivos para poder acceder desde distintos entornos sin cerrar otras sesiones.	
Criterios de aceptación:	CA1: El servicio de seguridad permitirá un máximo de tres sesiones activas por cuenta.	
	CA2: Si se inicia una cuarta sesión, la aplicación mostrará una vista de administración de sesiones para poder cerrar cualquier otra sesión activa.	
	CA3: El usuario podrá ver y cerrar manualmente sus sesiones activas desde la interfaz de gestión.	

Tabla 11. HU-SEC-06: Acceso a aplicaciones según permisos

Código: HU-SEC-06	Título: Acceso a aplicaciones según permisos	Prioridad: Alta
Descripción:	Como usuario, quiero que la aplicación me permita únicamente acceder a las aplicaciones y funcionalidades a las que tengo acceso para evitar confusión y mejorar mi experiencia de uso.	
Criterios de aceptación:	CA1: La aplicación debe validar los permisos de cada usuario para controlar el acceso.	
	CA2: Si el usuario intenta acceder a una aplicación sin tener los permisos correspondientes, se debe mostrar un mensaje de acceso denegado.	

Tabla 12. HU-SEC-07: Autenticación biométrica

Código: HU-SEC-07	Título: Autenticación biométrica	Prioridad: Media
Descripción:	Como usuario autenticado, quiero que se me solicite autenticación biométrica (si la tengo configurada en mi dispositivo) cada vez que acceda a una aplicación protegida, para mantener la seguridad de mis datos.	
Criterios de aceptación:	CA1: Cada acceso a una app con permisos protegidos debe requerir autenticación biométrica local.	
	CA2: Si la autenticación falla, no se deberá permitir el acceso de ninguna manera.	
	CA3: El uso de la autenticación biométrica es local, y solo se activará si el usuario tiene activada dicha configuración en su dispositivo.	

Tabla 13. HU-SEC-08: Configuración de parámetros generales desde frontend administrativo

Código: HU-SEC-08	Título: Configuración de parámetros generales desde frontend administrativo	Prioridad: Media
Descripción:	Como administrador, quiero gestionar los parámetros generales del servicio de seguridad desde una interfaz para personalizar y controlar configuraciones clave.	
Criterios de aceptación:	CA1: El sistema debe permitirme modificar configuraciones como tiempos de expiración de tokens y credenciales de Microsoft, entre otros.	
	CA2: Solo el administrador autorizado puede acceder a esta sesión.	

Tabla 14. HU-SEC-09: Verificación de conexión a internet

Código: HU-SEC-09	Título: Verificación de conexión a internet.	Prioridad: Alta
Descripción:	Como usuario, quiero que la aplicación verifique la conexión a internet antes de ejecutar procesos de autenticación o validación de sesión, para evitar errores de navegación o autenticación fallida.	
Criterios de aceptación:	CA1: La aplicación debe detectar si no hay conexión antes de realizar llamadas a la API.	
	CA2: En caso de no haber conexión, debe mostrar un widget informativo.	
	CA3: La aplicación deberá detectar automáticamente si se restableció la conexión a internet y permitir seguir interactuando en la aplicación.	

Tabla 15. HU-SEC-10: Acceso como invitado sin autenticación

Código: HU-SEC-10	Título: Acceso como invitado sin autenticación	Prioridad: Media
Descripción:	Como visitante, quiero acceder a la aplicación en modo invitado para explorar funcionalidades públicas sin necesidad de iniciar sesión.	
Criterios de aceptación:	CA1: El modo invitado debe solo permitirme ingresar a las aplicaciones o enlaces públicos.	
	CA2: Si el invitado intenta acceder a funciones o aplicaciones protegidas se le mostrara un mensaje de que no está autorizado.	
	CA3: El modo invitado no registra ninguna actividad en el servicio de seguridad.	

Tabla 16. HU-SEC-11: Carga de apps en el home

Código: HU-SEC-11	Título: Carga de apps en el Home	Prioridad: Alta
Descripción:	Como usuario quiero que la pantalla de inicio (Home) cargue las aplicaciones disponibles y me permita acceder según mis permisos para asegurar una experiencia segura.	
Criterios de aceptación:	CA1: La lista de las aplicaciones se construye en base a un registro de apps disponibles.	
	CA2: Aplicaciones externas, como Paseo Virtual UTN se redirigirán de forma automática a la Store de descarga sino la posee o abrirá la app.	
	CA3: El listado de las aplicaciones se utilizará tanto en el modo invitado como en el usuario que inició sesión.	

Tabla 17. HU-SEC-12: Envío de código OTP al correo institucional (MFA Habilitado)

Código: HU-SEC-12	Título: Envío de código OTP al correo institucional (MFA habilitado)	Prioridad: Alta
Descripción:	Como usuario con autenticación multifactor (MFA) activada en mi cuenta de Microsoft, quiero recibir un código de verificación en mi correo institucional para completar el proceso de inicio de sesión, garantizando una autenticación más segura.	
Criterios de aceptación:	CA1: El servicio de seguridad debe detectar si la cuenta del usuario tiene MFA Activado	
	CA2: El código OTP debe tener una vigencia limitada (5 minutos).	
	CA3: El usuario solo podrá continuar si ingresa correctamente el código recibido.	
	CA4: Si el código es incorrecto o expira, se debe mostrar un mensaje de error y permitir reenviar el código.	

Tabla 18. HU-SEC-13: Api Gateway para redireccionar solicitudes

Código: HU-SEC-13	Título: API Gateway para redireccionar solicitudes	Prioridad: Alta
Descripción:	Como sistema con arquitectura de microservicios, quiero que todas las solicitudes de los clientes pasen a través de un API Gateway para redireccionarlas al microservicio correspondiente.	
Criterios de aceptación:	CA1: El API Gateway debe actuar como proxy inverso para los microservicios registrados.	
	CA2: Cada solicitud del cliente debe ser validada en el Gateway antes de ser redireccionada.	
	CA3: La comunicación entre el Gateway y cualquier microservicio se efectúa solo si el API Key es el correcto.	
	CA4: El Gateway debe negar rutas que no tengan un header de autorización	
	CA5: El Gateway debe permitir rutas que no requieren header autorización como el login.	

Tabla 19. HU-SEC-14: Control de versión

Código: HU-SEC-14	Título: Control de versión de la app	Prioridad: Media
Descripción:	Como usuario de la aplicación móvil de la Universidad Técnica del Norte, quiero que el sistema verifique automáticamente si existe una nueva versión obligatoria de la app, para asegurarme de utilizar siempre la versión más reciente con todas las funcionalidades, mejoras y correcciones de seguridad implementadas.	
Criterios de aceptación:	CA1: Al iniciar la aplicación, debe realizarse una verificación automática de versión contra los valores definidos en el backend.	
	CA2: Si la versión instalada es inferior a la versión mínima requerida, el sistema debe redirigir al usuario a una pantalla de actualización obligatoria, impidiendo el uso de la aplicación hasta que se actualice.	
	CA3: El backend debe exponer un endpoint que retorne al menos los siguientes datos: version_actual, version_minima, verificar_version (booleano), y un mensaje_actualizacion (texto personalizado para mostrar al usuario).	
	CA4: Desde el panel administrativo, debe existir la posibilidad de activar o desactivar la verificación de versión, habilitando un modo mantenimiento si fuera necesario.	
	CA5: En caso de que la versión instalada cumpla con los requisitos, el usuario podrá continuar normalmente al flujo de navegación correspondiente (splash, login, home, etc.).	

Con base en los requisitos recolectados, se procedió a organizar y estimarlo en el Product Backlog. Para ello, el equipo SCRUM realizó una sesión de estimación utilizando la escala Fibonacci, tal como se recomienda en libro *Agile Estimating and Planning* [95]. Los resultados de esta estimación se presentan en *Tabla 20*.

Tabla 20. Product Backlog del proyecto

ORDEN	CÓDIGO	DESCRIPCION	ESTIMACIÓN (Puntos de historias)
1	HU-SEC-01	Inicio de sesión	8
2	HU-SEC-12	Envío de código OTP al correo institucional (MFA habilitado)	5

3	HU-SEC-13	API Gateway para redireccionar solicitudes	8
4	HU-SEC-03	Refresco automático del token de acceso	5
5	HU-SEC-04	Manejo de tokens expirados o inválidos	5
6	HU-SEC-05	Control de múltiples sesiones por usuario	8
7	HU-SEC-06	Acceso a aplicaciones según permisos	5
8	HU-SEC-11	Carga de apps en Home	5
9	HU-SEC-10	Acceso como invitado sin autenticación	3
10	HU-SEC-02	Cambio de contraseña de Microsoft	3
11	HU-SEC-07	Autenticación biométrica	3
12	HU-SEC-09	Verificación de conexión a internet	5
13	HU-SEC-14	Control de versión	5
14	HU-SEC-08	Configuración de parámetros generales desde frontend administrativo	3

2.2. Desarrollo de Fase II: Juego

2.2.1. Sprint 0.

El sprint 0 estable los principios de arquitectónicos que servirán de guía para el desarrollo del módulo de seguridad. Se sentaron las bases de las estructuras técnicas y organizativas necesarias para garantizar la escalabilidad y sostenibilidad del proyecto. Los principales puntos abordados son:

- Arquitectura general de la plataforma.

La arquitectura de la plataforma está basada en una arquitectura microservicios, donde cada elemento del sistema se desarrolla y lanza de forma independiente. La decisión arquitectónica permitió una evolución modular del sistema y facilitó su mantenimiento, su despliegue autónomo y la asignación de responsabilidades específicas a nivel de servicio.

La plataforma está compuesta por una aplicación móvil construida en flutter, la cual centraliza todas las funcionalidades dirigidas a los usuarios finales. Esta aplicación móvil consume servicios backend a través de un API Gateway, componente clave de la solución que actúa como punto de entrada único para todas las peticiones provenientes del cliente. Todos los microservicios, con acceso a base de datos institucional Oracle mediante conexión directa o triggers definidos, y el API Gateway están desarrollados con NestJS y desplegados mediante contenedores Docker sobre un servidor Linux. Todo lo descrito se puede visualizar en *Figura 12*.

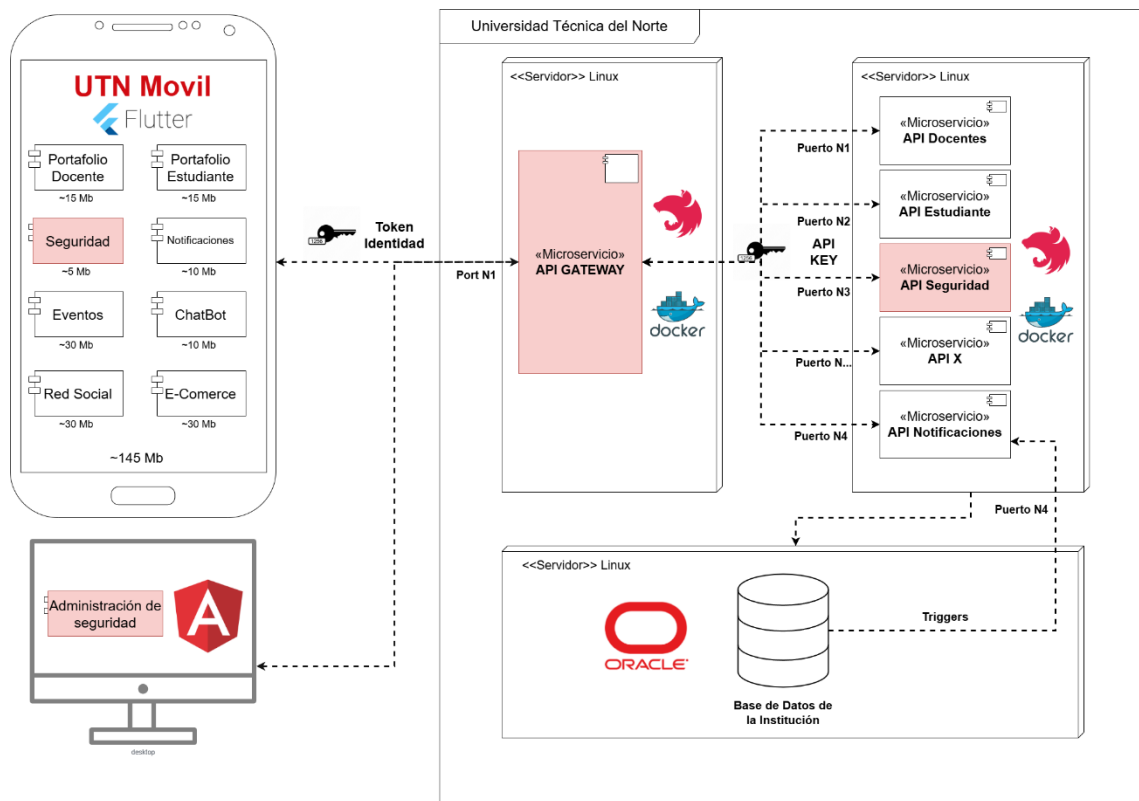


Figura 12. *Arquitectura Microservicios General para la plataforma UTN Móvil.*

Adicionalmente, se desarrolló un panel administrativo web utilizando tecnología Angular, dirigido a la gestión del módulo de seguridad, el cual también se conecta a través del Gateway. Gracias al enfoque modular descrito, el sistema puede crecer progresivamente integrando nuevos microservicios sin afectar los ya desplegados.

- *Estructura de la API Microservicios de Seguridad.*

El microservicio de seguridad se desarrolló utilizando el framework NestJS y TypeScript, siguiendo los principios de arquitectura modular y separación de responsabilidades. La estructura establecida en el proyecto es reutilizable la cuál puede ser replicada por otros microservicios de en la plataforma UTN Móvil. La estructura general del proyecto se encuentra en el directorio **src** como se puede observar en Figura 13.

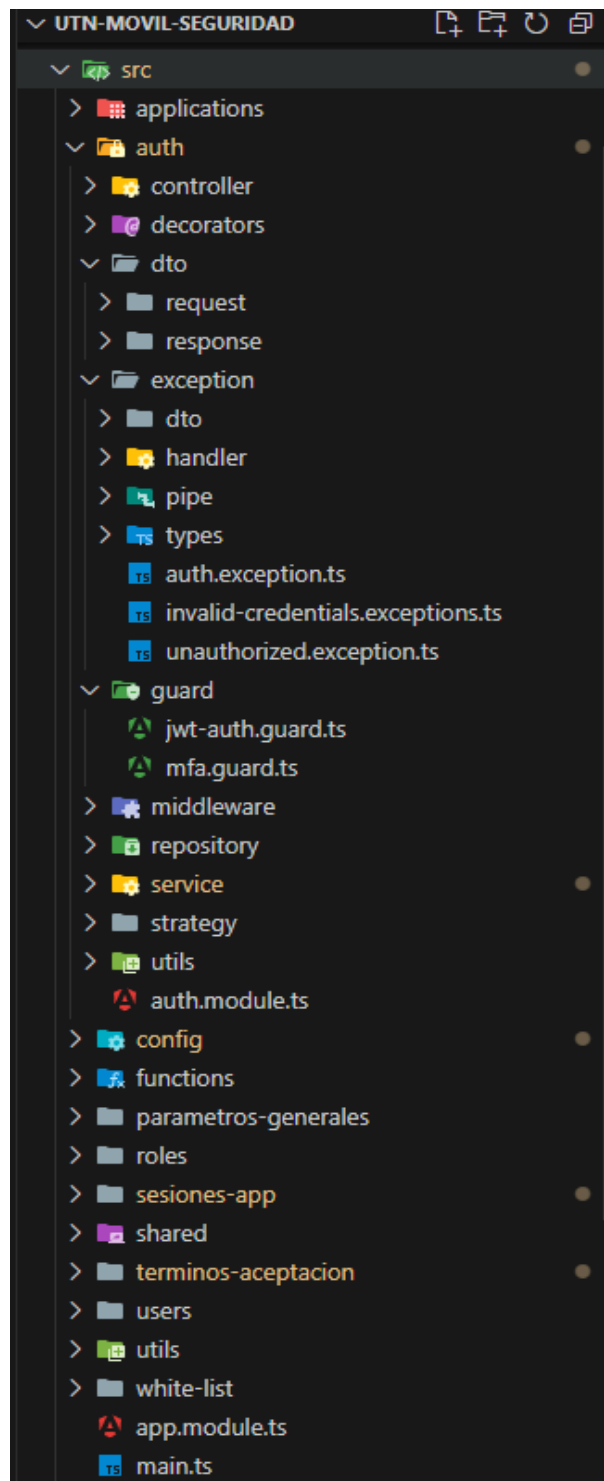


Figura 13. Estructura del API Microservicio de Seguridad

En primer lugar, el archivo **main.ts** es el punto de arranque de la aplicación y contiene la configuración inicial para levantar el servicio. Junto a él, se encuentra **app.module.ts**, que centraliza la configuración global del microservicio, importando módulos internos como **auth**, **users**, **terminos-aceptacion**, entre otros.

El módulo **auth** agrupa la lógica relacionada a la autenticación y autorización, organizando su funcionalidad en subdirectorios como:

- I. *controller*: Define los endpoints expuestos por el servicio. Gestiona las rutas relacionadas a login, verificando MFA, cierre de sesiones, entre otras.
- II. *decorators*: Contiene las funciones reutilizables que ayudan en la metadata de los controladores o métodos, como por ejemplo marcar rutas públicas o requerir MFA.
- III. *dto*: Se subdivide en *request* y *response*, permitiendo tipar y validar de forma robusta los cuerpos de entrada y salida en cada solicitud.
- IV. *guard*: Se incluye los guardias personalizados encargados de proteger las rutas del sistema evaluando si el token es válido o si el doble factor ha sido verificado.
- V. *service*: Se encarga de la lógica del negocio implementada como la validación de usuarios, verificación de MFA, generación de tokens o revocación de las sesiones activas.
- VI. *repository*: Se define el acceso a la base de datos a través de consultas específicas utilizando el patrón de repositorio para desacoplar la lógica de negocio del acceso a datos.
- VII. *strategy*: Contiene estrategias de autenticación como JWT para la integración con PassportJS.
- VIII. *types*: Agrupa tipos personalizados utilizados en los decoradores y servicios.

Además del módulo **auth**, el microservicio incluye otros módulos relevantes como:

- I. *usuarios*: Encargado de la gestión de cuentas de usuarios.
- II. *sessions-app*: Permite la administración de sesiones activas por dispositivos en donde se incluyen su validación y la terminación remota.
- III. *terminos-aceptacion*: Gestiona la lógica de aceptación de términos y condiciones, y políticas de privacidad por parte de los usuarios.
- IV. *white-list*: Controla las rutas que son permitidas acceder a ciertos endpoints son necesidad de requerir un header de autorización.

El directorio **config** contiene la configuración central del microservicio, incluyendo parámetros globales, manejo de roles, middlewares personalizados y valores compartidos. Todos estos módulos siguen el principio de inyección de dependencias para facilitar el mantenimiento. Cabe destacar que se utiliza TypeORM para la conexión con la base de datos Oracle, utilizando variables de entorno (.env) para proteger credenciales sensibles. La validación de datos se realizó con *class-validator*, y el mapeo de objetos con *class-transformer*.

La estructura modular permitió mantener un código limpio, escalable y mantenible.

- Estructura del API Gateway

El API Gateway es uno de los componentes centrales de la arquitectura basada en microservicios. La función principal es actuar como un punto de entrada único para todas las peticiones del cliente. El API Gateway centraliza y gestiona el enrutamiento y el filtrado de tráfico.

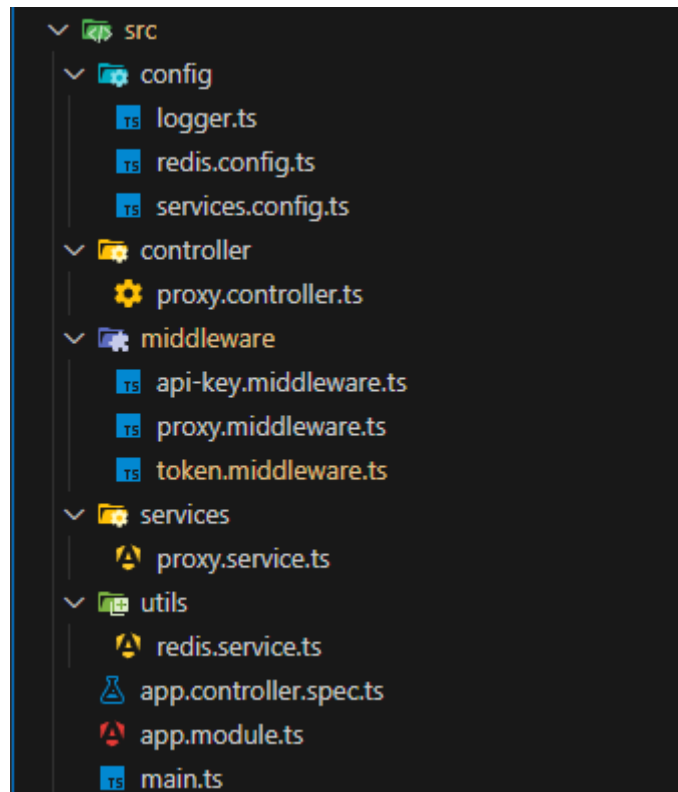


Figura 14. Estructura del API Gateway

La estructura del proyecto, mostrado en Figura 14, se organizó siguiendo principios de modularidad y separación de responsabilidades:

- I. main.ts: Archivo de entrada de la aplicación donde se inicializa el servidor y se configura los middlewares globales.
- II. app.module.ts: Es el módulo de la raíz de NestJS donde se declaran las dependencias globales del proyecto y se importan los controladores, servicios y configuraciones necesarias.
- III. proxy.controller.ts: Es el controlador principal encargado de recibir y redirigir las solicitudes HTTP.
- IV. proxy.service.ts: Se encarga de encapsular la lógica del reenvío de las solicitudes HTTP.
- V. middleware: Aquí se agrupan todos los middlewares críticos que son ejecutados antes de enrutar las peticiones.
- VI. config: Contiene las configuraciones del entorno y los parámetros reutilizables.

VII. `redis.service.ts`: Es un servicio auxiliar que encapsula las operaciones comunes sobre Redis como la validación y almacenamiento temporal con el fin de evitar llamadas repetidas.

- *Estructura de la Aplicación Móvil*

La aplicación móvil se realizó utilizando el framework Flutter, bajo la arquitectura limpia. La estructura del proyecto permitió mantener una alta cohesión por funcionalidades.

Toda la aplicación se encuentra contenida dentro del directorio **lib**, el cual es la raíz lógica del proyecto en Flutter. Dentro de este, la carpeta principal **apps**, se agrupó los distintos módulos o sub-aplicaciones como **auth**, **portafolio_docente**, **portafolio_estudiante**, entre otros, como se puede observar en Figura 15. Cada sub-aplicación posee su propio sistema de rutas, dependencias y lógica de negocio desacoplado, lo que facilitó su independencia y reusabilidad.

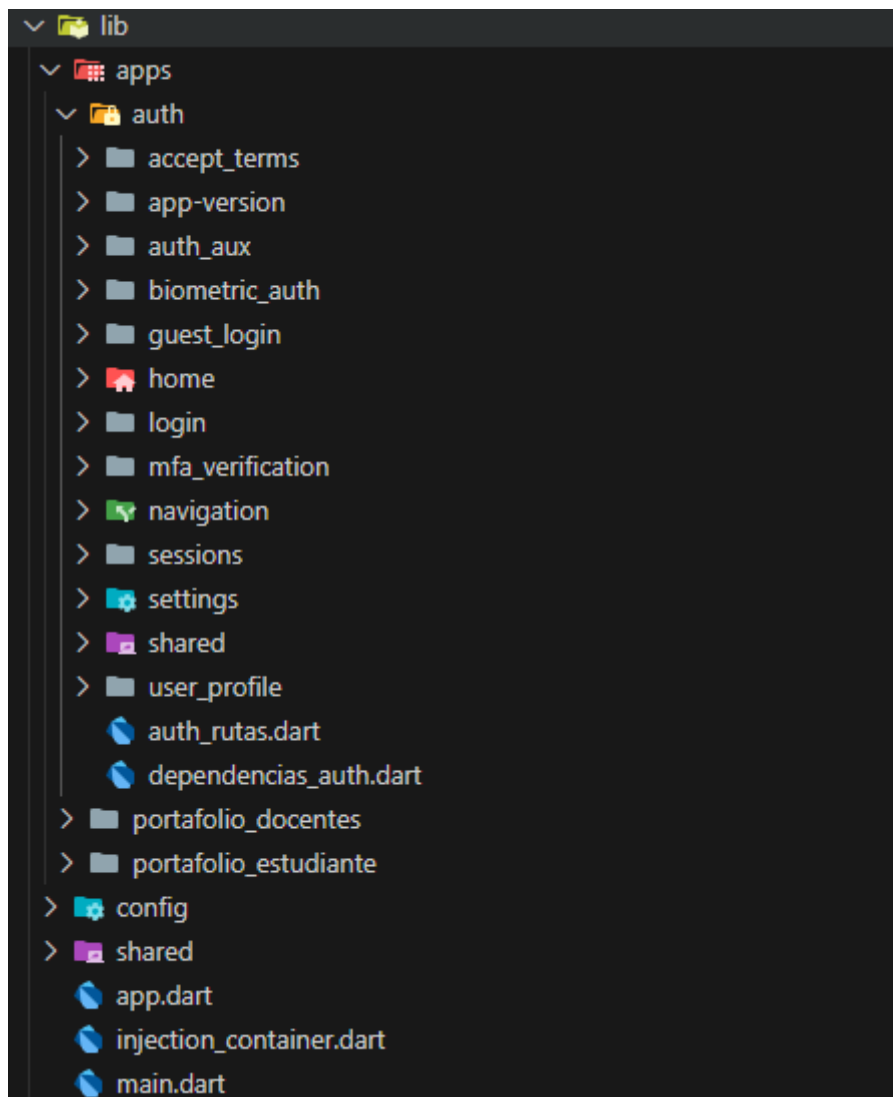


Figura 15. Estructura de carpetas por sub-aplicación

I. Estructura de carpetas por sub-aplicación.

Cada sub-aplicación contiene la siguiente división lógica, mostrada en Figura 16:

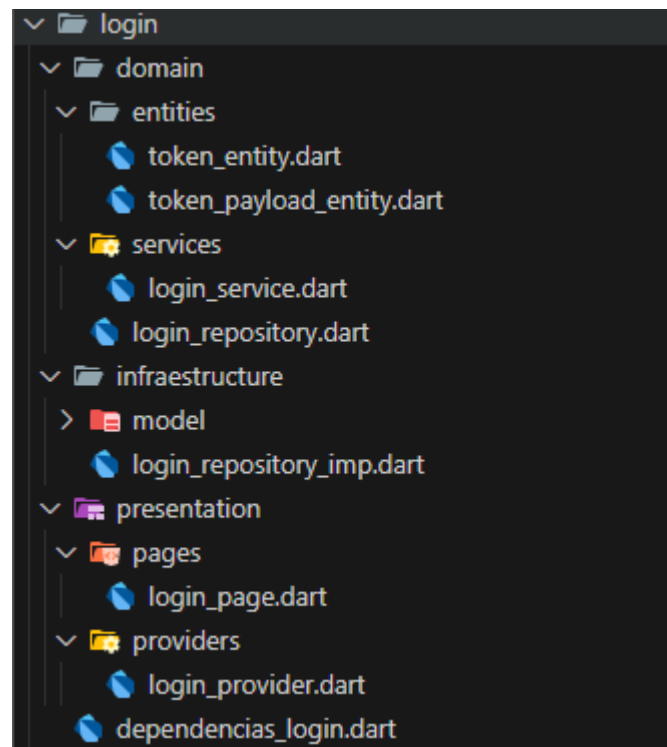


Figura 16. Estructura de capas para cada caso (Caso Login)

- a. *domain*: Esta capa define el núcleo del caso de uso. Incluye entidades, así como interfaces de servicios y repositorios abstractos que declaran los métodos requeridos por la lógica del negocio, sin preocuparse por su implementación.
- b. *infraestructura*: Esta implementa las interfaces definidas en el dominio. El *repository_imp*, en esta capa, se encarga de conectar con la Api Rest y transforma las respuestas en modelos que puedan ser utilizados por la lógica del dominio. De esta manera, se logró desacoplar las dependencias externas y facilitó el mantenimiento.
- c. *presentation*: Esta capa grupa todos los elementos (widgets) visuales e interacción con el usuario. Se encuentra divididos en dos sesiones, las **pages**, que son las pantallas o vistas, y los **providers**, que son gestores de estado que utilizan el patrón *Provider* junto con la inyección de dependencias para acceder a la lógica de negocio.

II. Inyección de dependencias.

```

8  void initDependenciasLogin(GetIt getIt) {
9      // Repositories
10     getIt.registerLazySingleton<LoginRepository>(() => LoginRepositoryImp(getIt()));
11
12     // Services
13     getIt.registerLazySingleton<LoginService>(() => LoginService(
14         getIt.get<LoginRepository>(),
15         getIt.get<TokenService>(),
16         getIt.get(),
17         getIt.get(),
18     ));
19
20     // Providers
21     getIt.registerFactory<LoginProvider>(() => LoginProvider(getIt(), getIt(), getIt()));
22 }

```

Figura 17. Inyección de dependencias (Caso login)

Se utilizó la librería *get_it* para gestionar la inyección de dependencias en toda la aplicación. Cada sub-aplicación cuenta con un archivo específico. Un ejemplo es *dependencias_login.dart*, en donde se registraron los repositorios, servicios y providers necesarios, esto podemos observar en Figura 17. Estos archivos fueron invocados desde el archivo central *dependencias_auth.dart* mediando métodos como podemos observar en Figura 18.

```

11
12 void initDependenciasAuth(GetIt getIt) {
13     initDependenciasUserProfile(getIt);
14     initDependenciasLogin(getIt);
15     initDependenciasGuestLogin(getIt);
16     initDependenciasMfa(getIt);
17     initDependenciasAcceptTerms(getIt);
18     initDependenciasSessions(getIt);
19     initDependenciasSettings(getIt);
20     initDependenciasAuthAux(getIt);
21     initDependenciasAppVersion(getIt);
22 }
23

```

Figura 18. Consolidación de dependencias (Módulo Auth)

Los servicios y repositorios son registrados como Lazy Singleton para garantizar una única instancia durante todo el ciclo de vida de la aplicación, mientras que los providers se registraron como Factory para asegurar una instancia nueva por cada render de pantalla.

III. Gestión de rutas.

La navegación de la aplicación se gestionó utilizando la librería *go_router*. En el archivo *auth_rutas.dart* se definieron rutas específicas del módulo auth como /login, /mfa, /home, /splash, entre otros, gestionadas con transiciones personalizadas como se puede observar en Figura 19. Además, cada ruta puede tener condiciones de redirección.

```
21 class AuthRutas {
22   static final rutasAuth = [
23     GoRoute(
24       name: 'home',
25       path: '/home',
26       redirect: (context, state) => verificarTerminosYCondiciones(context),
27       pageBuilder: (context, state) => buildPageWithDefaultTransition(
28         context: context,
29         state: state,
30         child: const HomePage(),
31       ),
32     ), // GoRoute
33     GoRoute(
34       name: 'splash',
35       path: '/',
36       pageBuilder: (context, state) => buildPageWithDefaultTransition(
37         context: context,
38         state: state,
39         child: const SplashPage(),
40       ),
41     ), // GoRoute
42     GoRoute(
43       name: 'login',
44       path: '/login',
45       pageBuilder: (context, state) => buildPageWithDefaultTransition(
46         context: context,
47         state: state,
48         child: const LoginPage(),
49       ),
50     ), // GoRoute
51     GoRoute(
52       name: 'mfa',
53       path: '/mfa',
54       pageBuilder: (context, state) {
55         final email = state.extra as String;
56         return buildPageWithDefaultTransition(
57           context: context,
58           state: state,
59           child: MfaPage(email: email),
60         );
61       },
62     ), // GoRoute
63   ],
64 }
```

Figura 19. Gestión de rutas (Módulo Auth)

Esta estructura garantizó un enrutamiento modular y extensible, facilitando la incorporación de nuevas funcionalidades en futuras fases del desarrollo.

2.2.2. Sprint 1.

I. Planificación General

En el primer sprint se abordó el desarrollo de las tres primeras historias de usuario (HU) las cuales fueron priorizadas acorde al orden establecido en el Product Backlog mostrado en *Tabla 20*. La implementación de las funcionalidades descritas en las HU son la base mínima operativa del sistema.

El Sprint Backlog correspondiente, detallado en *Tabla 21*, se definió de forma colaborativa durante la reunión de planificación, estableciendo objetivos específicos y distribuyendo tareas entre los miembros del equipo. La estimación para este sprint es de un total de 40 horas, de las cuales 35 horas se destinaron al desarrollo efectivo del total de las tareas asignadas que incluyeron desarrollo, validación, revisión de código y pruebas iniciales, y 5 horas se reservaron como estrategia para ajuste necesario o imprevistos.

A continuación, se detallan las tareas planificadas:

Tabla 21. Sprint Backlog - Sprint 1

HU	NOMBRE	TAREAS	HORAS
HU-SEC-01: Inicio de sesión	Inicio de sesión	Diseñar flujo de login	2
		Desarrollar endpoint en backend	4
		Desarrollar pruebas unitarias en endpoint	2
		Implementar pantalla de inicio de sesión en Flutter	4
HU-SEC-12: Envío de código OTP al correo institucional (MFA Habilitado)	Envío de código OTP al correo institucional	Diseñar flujo de MFA y redacción de lógica de envío de OTP	2
		Desarrollar endpoint de envío OTP en backend	4
		Desarrollar pruebas unitarias del envío de OTP	2
		Implementar pantalla de ingreso de código OTP en Flutter	4
HU-SEC-13: Api Gateway para redireccionar solicitudes	API Gateway	Definir estructura base del Gateway y diseño del flujo de rutas	2
		Implementar lógica de proxy y reenvío con NestJS	3
		Crear middleware para autenticación y validación de token	2
		Realizar pruebas funcionales del API Gateway	4
Total: 35 horas de implementación + 5 horas de ajuste = 40 horas			

II. Incremento

Como resultado del Sprint 1 se generó un incremento funcional clave para la plataforma UTN Móvil, el cual está completamente funcional, después de haber sido validado mediante pruebas unitaria y funcionales. Este incremento sirvió como base para el flujo de autenticación central de toda la aplicación. A continuación, se detallan lo que se ha desarrollado, organizado por historias de usuarios.

- Inicio de sesión (HU-SEC-01: Inicio de sesión)

El incremento correspondiente a esta historia de usuario implementó el proceso de inicio de sesión en la aplicación, utilizando las credenciales institucionales. Con el fin de entender este proceso, en Figura 20 se presente un diagrama de secuencia donde se muestra el recorrido de la información, contempla desde el ingreso de las credenciales hasta la muestra de la información.

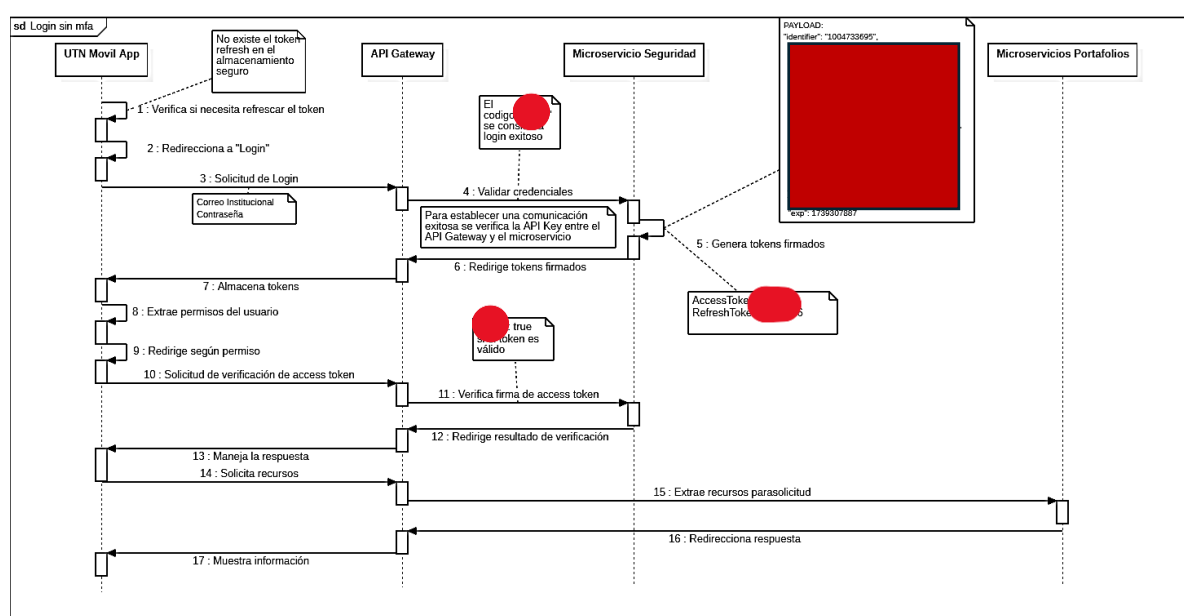


Figura 20. Diagrama de secuencia - Inicio de sesión

Como incremento final de esta historia de usuario se obtuvo una interfaz funcional que permitió al usuario iniciar sesión de forma clara, segura y visualmente amigable. En Figura 21 se observa la pantalla login implementada, donde el usuario puede ingresar su correo institucional y contraseña, y recibe alertas en caso de errores.



UTN
IBARRA - ECUADOR

Iniciar Sesión

Correo Institucional

Contraseña

Ingresar

[Ingresar como invitado](#)

[¿Olvidaste tu contraseña?](#)

© 2025, Universidad Técnica del Norte. Todos los derechos reservados

Figura 21. *Pantalla de Inicio de Sesión*

Este incremento representó un primer paso para el completar la siguiente historia de usuario, correspondiente a la verificación del código OTP detallado en

Tabla 17. HU-SEC-12: Envío de código OTP al correo institucional (MFA Habilitado).

- Envío de Código OTP (HU-SEC-12: Envío de código OTP al correo institucional (MFA Habilitado)HU-SEC-12: Envío de código OTP al correo institucional (MFA Habilitado)HU-SEC-12: Envío de código OTP al correo institucional (MFA Habilitado))

El incremento generado en esta historia de usuario correspondió a la implementación del envío del código OTP al correo institucional de usuario, como medida de seguridad adicional posterior al inicio de sesión. Con ello se aseguró que únicamente el titular legítimo de la cuenta pueda continuar. Para comprender este proceso con claridad, en Figura 22 se presenta un diagrama de secuencia que detalla el recorrido completo.

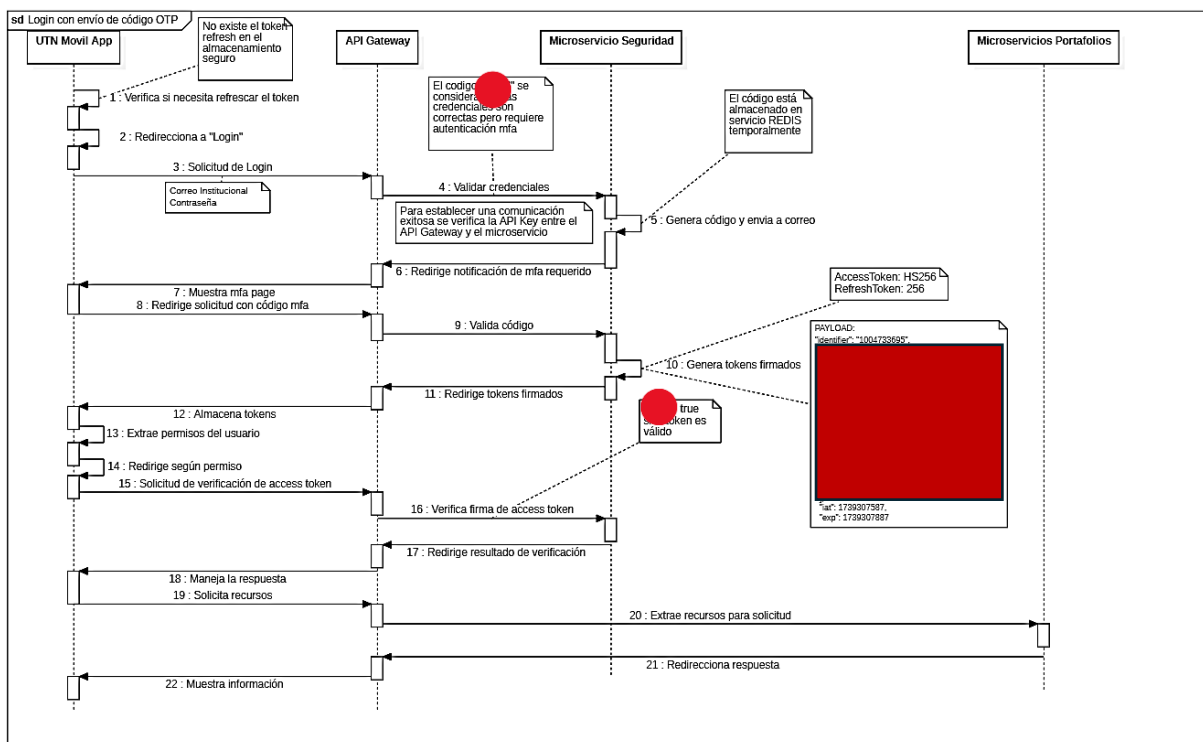


Figura 22. Diagrama de secuencia - Inicio de Sesión con envío de código OTP

Como resultado visible para el usuario final, se desarrolló una pantalla dedicada al ingreso del código OTP. En Figura 23 se puede observar la interfaz resultante.

UTN
IBARRA - ECUADOR

Verificación MFA

Por favor, ingresa el código de verificación enviado a tu correo.

 Código MFA

Reenviar código

Validar

Cancelar

El código de verificación fue enviado al correo:
micabascangop@utn.edu.ec

© 2025, Universidad Técnica del Norte. Todos los derechos reservados

Figura 23. Pantalla de ingreso de código OTP

- API Gateway (HU-SEC-13: Api Gateway para redireccionar solicitudes)

El incremento desarrollado en esta historia de usuario corresponde a la implementación del API Gateway como componente central para la gestión de las solicitudes dentro de la arquitectura de microservicios. Este Gateway funciona como un proxy inverso, recibiendo las peticiones provenientes de la aplicación móvil y redirigiéndolas hacia el microservicio correspondiente. El flujo de este proceso se puede observar en Figura 24.

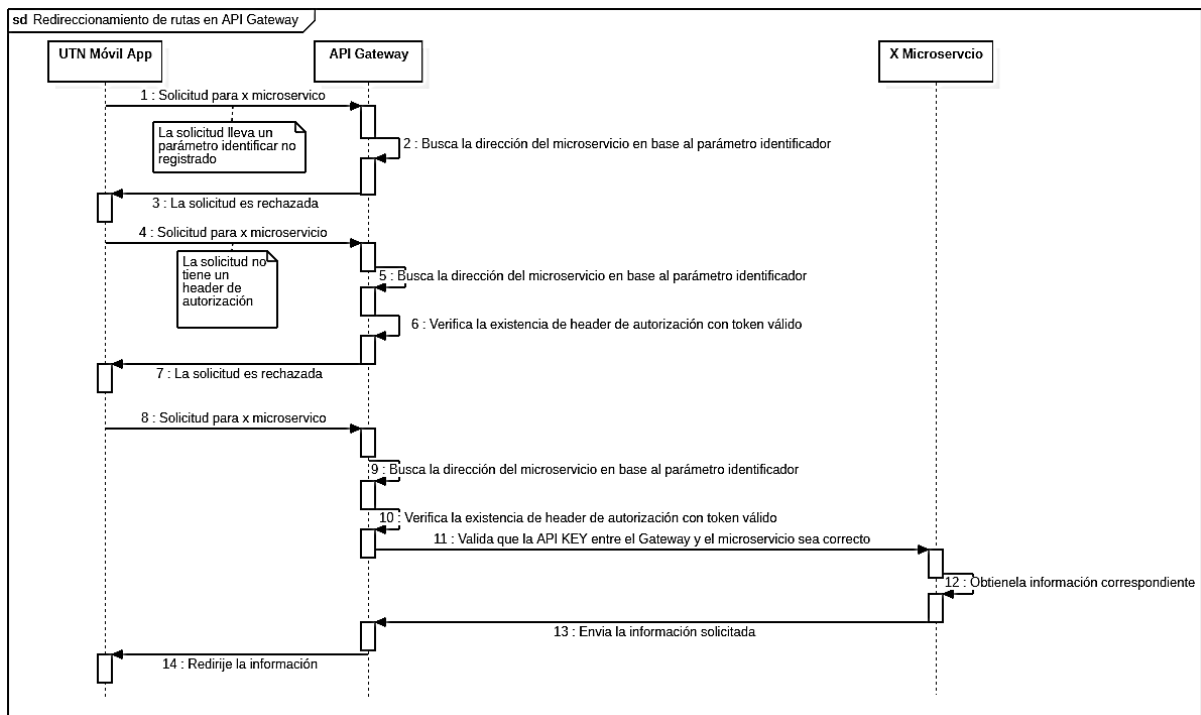


Figura 24. Diagrama de secuencia - Redireccionamiento de solicitudes en API Gateway

Este componente fue desarrollado utilizando NestJS, empleando middlewares para interceptar y validar tokens antes de redireccionar las solicitudes. Para verificar el su correcto funcionamiento, se realizaron las siguientes pruebas unitarios detallados en *Tabla 22*.

Tabla 22. Resumen de pruebas unitarias ejecutadas a API Gateway

N°	ARCHIVO USADO	NOMBRE DE TEST	CA CUBIERTO
1	proxy.service.spec.ts	Debe invocar el proxy correctamente para un microservicio registrado	CA1
2	proxy.service.spec.ts	Debe lanzar <code>HttpException</code> si la URL no tiene segmentos válidos	CA1
3	proxy.service.spec.ts	Debe lanzar <code>HttpException</code> si el microservicio no está registrado	CA1
4	token.middleware.spec.ts	Permite acceso si está en white list	CA5
5	token.middleware.spec.ts	Lanza <code>UnauthorizedException</code> si no hay token	CA4
6	token.middleware.spec.ts	Permite acceso si el token es válido	CA2 – CA3
7	token.middleware.spec.ts	Lanza <code>UnauthorizedException</code> si el token es inválido	CA2 – CA3

En Figura 25 se puede observar el resultado obtenido de las pruebas unitarias realizadas a este proyecto.

```

> utn-movil-api-gateway@0.0.1 test
> jest

PASS src/test/proxy.service.spec.ts
PASS src/test/token.middleware.spec.ts

Test Suites: 2 passed, 2 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        3.746 s, estimated 5 s
Ran all test suites.

```

Figura 25. Resultados de test unitario en API Gateway

III. Revisión

A continuación, en *Tabla 23* se detalla el correspondiente cumplimiento de los criterios de aceptación de cada una de las historias de usuario desarrolladas en este Sprint 1.

Tabla 23. Revisión de criterios de aceptación involucrados en Sprint 1

HISTORIA DE USUARIO	CRITERIO DE ACEPTACIÓN	CUMPLE
HU-SEC-01: Inicio de sesión	CA1: El servicio de seguridad debe verificar que el usuario existe y las credenciales son correctas.	SI
	CA2: Las contraseñas deben enviarse codificadas para asegurar el envío de todos los caracteres.	SI
	CA3: Al iniciar sesión, se generará un token de acceso seguro con expiración controlada, y un token de refresco con expiración prolongada y controlada.	SI
HU-SEC-12: Envío de código OTP al correo institucional (MFA Habilitado)	CA1: El servicio de seguridad debe detectar si la cuenta del usuario tiene MFA Activado	SI
	CA2: El código OTP debe tener una vigencia limitada (5 minutos).	SI
	CA3: El usuario solo podrá continuar si ingresa correctamente el código recibido.	SI
	CA4: Si el código es incorrecto o expira, se debe mostrar un mensaje de error y permitir reenviar el código.	SI
	CA1: El API Gateway debe actuar como proxy inverso para los microservicios registrados.	SI

HU-SEC-13: Api Gateway para redireccionar solicitudes	CA2: Cada solicitud del cliente debe ser validada en el Gateway antes de ser redireccionada.	SI
	CA3: La comunicación entre el Gateway y cualquier microservicio se efectúa solo si el API Key es el correcto.	SI
	CA4: El Gateway debe negar rutas que no tengan un header de autorización	SI
	CA5: El Gateway debe permitir rutas que no requieren header autorización como el login.	SI

IV. Retrospectiva

En la reunión retrospectiva correspondiente a este Sprint, se destacó como acierto el uso de una arquitectura basada en microservicios, esto permitió desarrollar componentes independientes, escalables y fáciles de probar. La implementación del API Gateway como capa de entrada centralizada también fue valorada positivamente, ya que simplifica la gestión de las solicitudes.

Para el aspecto visual se reconoció el acierto de aplicar la paleta institucional en las pantallas móviles. Como aspecto a mejorar se identificaron ciertos diseños de botones, fuente de letra y colores que no estaban alineados con el tema de la aplicación aplicado. Se propuso ajustar los diseños y mantenerlos durante todo el desarrollo de las pantallas futuras.

2.2.3. Sprint 2.

I. Planificación General

El sprint 2 se abordó funcionalidades esenciales para fortalecer la seguridad, gestión de sesiones y expiración de autenticación en la plataforma. La priorización de las actividades se refleja en el Product Backlog mostrado en *Tabla 20*. Las funcionalidades de las historias de usuario se orientan en fortalecer la gestión de la autenticación, control de sesiones activas y el tratamiento de expiración de tokens dentro de la plataforma.

El Sprint Backlog correspondiente, detallado en *Tabla 24*, ha se definió de forma colaborativa durante la reunión de planificación, estableciendo objetivos específicos y distribuyendo las tareas técnicas necesarias. La estimación para este esprint, igual al anterior, es de un total de 40 horas, de las 35 fueron destinadas al desarrollo efectivo de las tareas programadas, y 5 se reservaron como estrategia de contingencia para atender ajustes imprevistos o refinamientos.

A continuación, se detallan las tareas planificadas:

Tabla 24. *Sprint Backlog - Sprint 2*

HU	NOMBRE	TAREAS	HORAS
HU-SEC-03	Refresco automático del token de acceso	Diseñar flujo de comunicación del flujo de refresco de token de acceso	2
		Desarrollar endpoint de verificación y refresco de token en backend	4
		Desarrollar pruebas unitarias del refresco del token de acceso en endpoint	2
		Implementar lógica en cliente Flutter	4
HU-SEC-04	Manejo de tokens expirados o inválidos	Diseñar flujo de validación de tokens expirados o inválidos	2
		Desarrollar lógica de negocio en backend	4
		Desarrollar pruebas unitarias del manejo de tokens en endpoint	2
		Implementar lógica a estas excepciones en Flutter	4
HU-SEC-05	Control de múltiples sesiones por usuario	Diseñar flujo de múltiples sesiones	2
		Desarrollar endpoint de iniciar y cerrar sesiones	3
		Desarrollar pruebas unitarias del control de múltiples sesiones	2
		Implementar pantalla de administración de sesiones en cliente Flutter	4
Total: 35 horas de implementación + 5 horas de ajuste = 40 horas			

II. Incremento

Como resultado del Sprint 2 se generó un incremento orientando a fortalecer el proceso de renovación del token de acceso en la aplicación móvil. Este proceso garantizó que el usuario no necesita volver a ingresar sus credenciales si su token expira durante una sesión activa, mejorando así la experiencia de usuario y asegurando la continuidad segura en el acceso a los recursos. A continuación, se muestra el desarrollo de las historias de usuario:

- Refresco automático del token de acceso (HU-SEC-03: Refresco automático del token de acceso)

El incremento correspondiente es la implantación del proceso de renovación del token de acceso en la aplicación móvil. El proceso de renovación garantizó que el usuario no necesite volver a ingresar sus credenciales si su token de acceso expira durante una sesión activa.

Para entender el flujo, en Figura 26 se presenta un diagrama de secuencia muestra la solicitud de renovación y respuesta de un token nuevo.

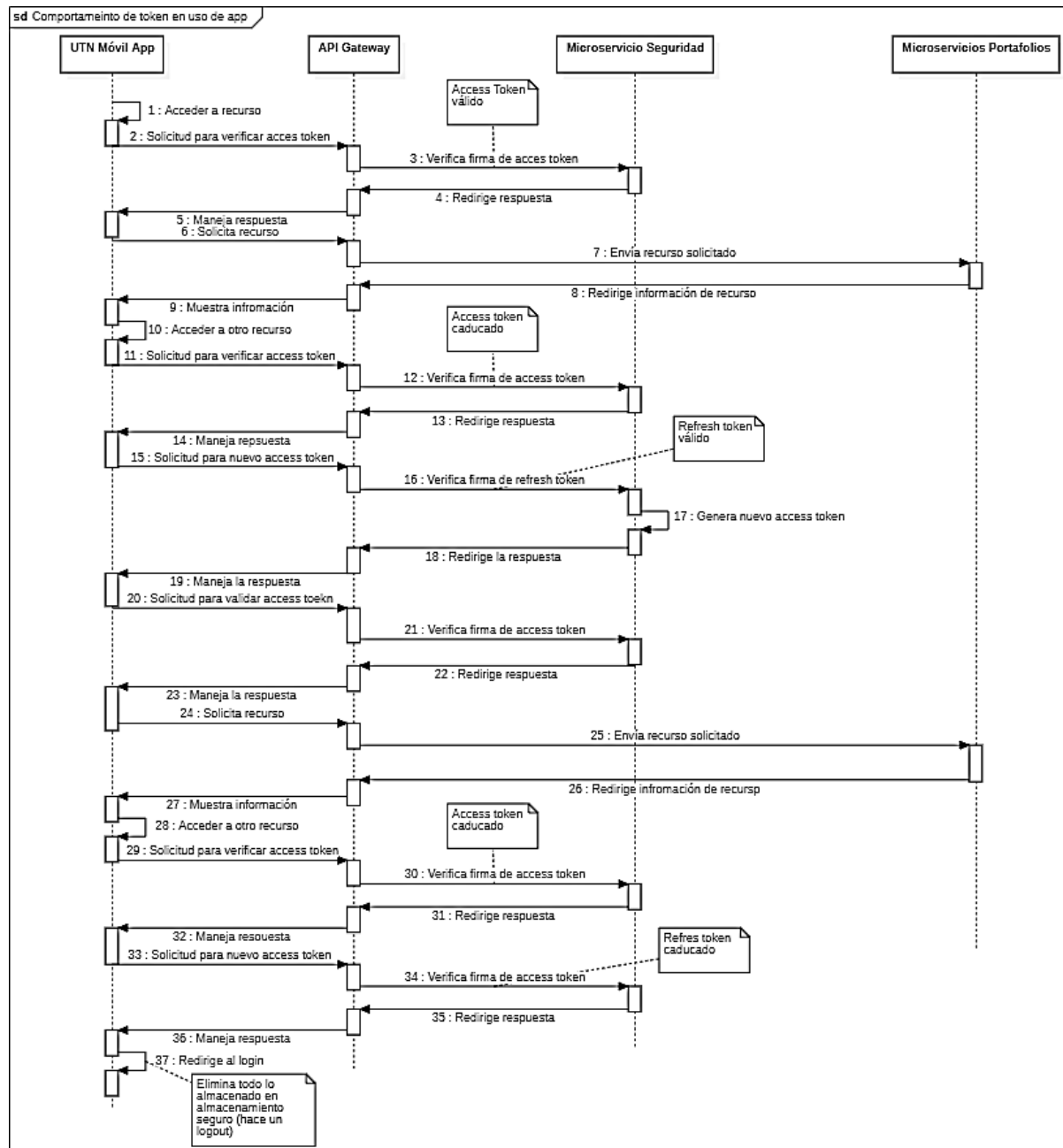


Figura 26. Diagrama de secuencia - Comportamiento de token de acceso y refresco

Como resultado final, se desarrolló y validó la lógica de renovación de automática tanto en el backend como en el cliente Flutter. Para validar el cumplimiento funcional y de seguridad de esta HU, se realizaron pruebas unitarias sobre el servicio de *TokenService*, verificando escenarios de éxito y fallos previstos. En Tabla 25 se resumen las pruebas ejecutadas:

Tabla 25. Resumen de pruebas unitarias a TokenService

N°	ARCHIVO USADO	NOMBRE DE TEST	CA CUBIERTO
1	token.service.spec.ts	Debe refrescar el accessToken si el refreshToken es válido	CA1 – CA2
2	token.service.spec.ts	Debe fallar si el refreshToken es inválido	CA3
3	token.service.spec.ts	Debe fallar si la sesión asociada está cerrada	CA13

Finalmente, en Figura 27 se observa la ejecución exitosa de las pruebas unitarias correspondientes a este incremento.

```
PS C:\Users\sebas\Desktop\UTN MOVIL\utn-movil-seguridad> npm run test src/auth/test/token.service.refresh.spec.ts
> utn-movil-seguridad@0.0.1 test
> jest src/auth/test/token.service.refresh.spec.ts

PASS src/auth/test/token.service.refresh.spec.ts
TokenService - Refresco automático del token (HU-SEC-03)
  ✓ debería refrescar el accessToken si el refreshToken es válido (CA1, CA2) (12 ms)
  ✓ X debería fallar si el refreshToken es inválido (CA3) (10 ms)
  ✓ X debería fallar si la sesión está cerrada (CA3) (3 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:  0 total
Time:        3.824 s, estimated 4 s
Ran all test suites matching /src\\auth\\test\\token.service.refresh.spec.ts/i.
```

Figura 27. Resultados de pruebas unitarias – Renovación automática de token.

- Manejo de tokens expirados o inválidos (HU-SEC-04: Manejo de tokens expirados o inválidos)

Se desarrolló un interceptor global en Flutter que monitorea todas las respuestas provenientes de servicios protegidos. En caso de detectar un error, como por ejemplo 401 Unauthorized, la aplicación ejecuta una limpieza inmediata del estado de autenticación, eliminando el token de almacenamiento seguro, y redirigiendo al usuario de forma automática a la pantalla de login. Este flujo puede ser observado en *Figura 26* que se presentó en la historia de usuario anterior.

Como parte del incremento, se validó el correcto funcionamiento mediante pruebas manuales y simulaciones controladas, cubriendo escenarios como:

- Token vencido naturalmente por tiempo.
- Token inválido intencionalmente (modificado)
- Usuario sin token
- Token válido (flujo normal)

En Figura 28. Pantalla de login con mensaje de sesión caducada se puede observar una captura del mensaje mostrado al usuario al detectar que su sesión ha expirado, notificando de forma clara y amigable la necesidad de iniciar sesión nuevamente.

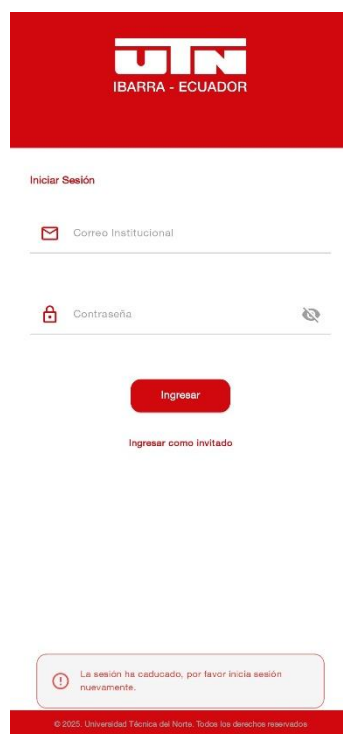


Figura 28. Pantalla de login con mensaje de sesión caducada

- Control de múltiples sesiones (HU-SEC-05: Control de múltiples sesiones por usuario)

El incremento correspondiente se asocia directamente a la implementación de la gestión de múltiples sesiones activas dentro de la plataforma. La implementación permitió al usuario iniciar sesión desde distintos dispositivos, sin forzar el cierre inmediato de otras sesiones activas, siempre y cuando no exceda el límite definido.

Con el fin de asegurar el control, se implementó una verificación de la cantidad de sesiones activas por usuario. En el caso de detectarse una excedente del límite, automáticamente se muestra una interfaz para gestionar las sesiones manualmente, lo que permite al usuario cerrar aquellas que no desea mantener activas.

En Figura 29 se muestra el diagrama de secuencia que representa este proceso.

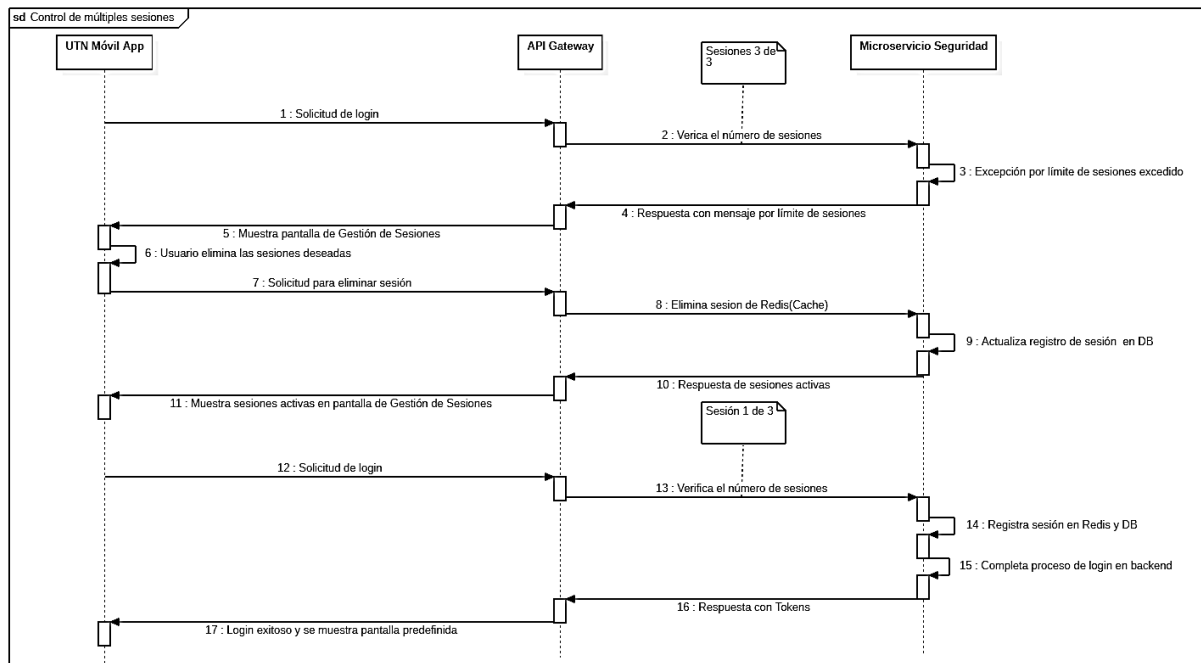


Figura 29. Diagrama de secuencia - Gestión de múltiples sesiones

Como resultado visible para esta historia de usuario, en Figura 30 se puede observar la pantalla de gestión de sesiones en caso de que el usuario intente crear una cuarta sesión.

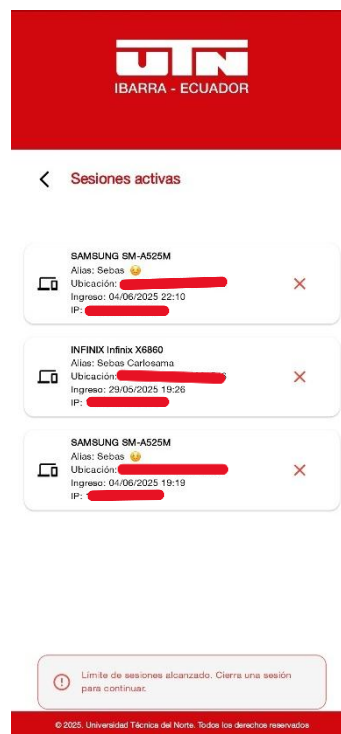


Figura 30. Pantalla 'Gestión de sesiones activas'

III. Revisión

En este sprint se consolidó el flujo de autenticación de la plataforma UTN Móvil. En *Tabla 26* se presenta el detalle del cumplimiento de los criterios de aceptación correspondiente a cada historia de usuario.

Tabla 26. Revisión de criterios de aceptación involucrados Sprint 2

HISTORIA DE USUARIO	CRITERIO DE ACEPTACIÓN	CUMPLE
HU-SEC-03: Refresco automático del token de acceso	CA1: El servicio de seguridad debe generar un nuevo token para reemplazar al que expiró si el usuario está usando la aplicación.	SI
	CA2: La renovación del token debe realizarse en segundo plano.	SI
	CA3: Si el token de refresco expira, el token de acceso no podrá volver a refrescarse y la sesión caducará, solicitando que el usuario nuevamente se autentique.	SI
HU-SEC-04: Manejo de tokens expirados o inválidos	CA1: La aplicación debe interceptar errores de tipo 401 o inválidos en todas las llamadas protegidas.	SI
	CA2: Si el token es inválido o ha expirado, debe eliminarse del almacenamiento seguro.	SI
	CA3: El usuario debe ser redirigido automáticamente a la pantalla de inicio de sesión, indicando un mensaje claro del motivo.	SI
HU-SEC-05: Control de múltiples sesiones por usuario	CA1: El servicio de seguridad permitirá un máximo de tres sesiones activas por cuenta.	SI
	CA2: Si se inicia una cuarta sesión, la aplicación mostrará una vista de administración de sesiones para poder cerrar cualquier otra sesión activa.	SI
	CA3: El usuario podrá ver y cerrar manualmente sus sesiones activas desde la interfaz de gestión.	SI

IV. Retrospectiva

Durante este sprint se destacó como acierto principal la correcta implementación del control de sesiones activas y el refresco automático de tokens, funcionalidades críticas que permitieron fortalecer la seguridad de la plataforma.

Se propuso como acción concreta extender, en caso de ser necesario, las horas reservadas para tareas de ajuste técnico y validación de integraciones, así como reforzar el uso de flujos visuales (diagramas de secuencia) para facilitar la comprensión entre todos los integrantes del equipo.

2.2.4. Sprint 3

I. Planificación General

Las funcionalidades del sprint fueron orientadas a optimizar el acceso a las aplicaciones dentro de la plataforma, de esta forma se garantiza una experiencia de usuario acorde al rol que posea. Las funcionalidades implementadas permitieron aplicar validaciones dinámicas según el tipo de permiso (RBAC). Se muestran de forma dinámica las aplicaciones disponibles en el Home y se habilita el acceso como invitado. Las historias de usuario fueron priorizadas en el Product Backlog mostrado en *Tabla 20*, y el cumplimiento resulto esencial para robustecer el flujo de navegación y acceso en la plataforma.

El Sprint Backlog mostrado en *Tabla 27*, fue definido de forma colaborativa durante la reunión de planificación del sprint. Se establecieron los objetivos técnicos de cada historia de usuario y tareas específicas requeridas para su implementación. La estimación total de este sprint es de 40 horas, 30 horas se destinaron al desarrollo efectivo, validación y pruebas. Las 10 horas restantes se reservaron como margen de ajuste antes cualquier imprevisto.

A continuación, se detalla las tareas planificadas para este sprint:

Tabla 27. Sprint Backlog - Sprint 3

HU	NOMBRE	TAREAS	HORAS
HU-SEC-06	Acceso a aplicaciones según permisos	Diseñar flujo de acceso a aplicaciones según los permisos	2
		Desarrollar lógica para decodificar permisos del token	5
		Implementar lógica para dar permisos de acceso	4
HU-SEC-11	Carga de apps en el Home	Desarrollar archivo base para centralizar la configuración visual de cada uno de los botones de acceso a los módulos	1
		Desarrollar lógica para mapear aplicaciones con permisos	5
		Implementar pantalla de home dinámico	4
HU-SEC-10	Acceso como invitado sin autenticación	Desarrollar lógica para acceder sin login al home	5
		Implementar pantalla de home como invitado	4
Total: 30 horas de implementación + 10 horas de ajuste = 40 horas			

II. Incremento

El incremento correspondiente a este sprint representó un avance significativo en la gestión de acceso de usuarios dentro de la plataforma UTN Móvil. Durante este ciclo se incorporó funcionalidades clave que permitieron validar permisos de forma dinámica, habilitar el acceso como invitado y mostrar las aplicaciones disponibles según el tipo de usuario. Estos desarrollos fueron implementados tanto en el backend como en el cliente móvil, permitiendo una navegación coherente, segura y personalizada.

A continuación, se describen los resultados obtenidos por cada historia de usuario incluida en este sprint:

- Validar acceso según permiso (HU-SEC-06: Acceso a aplicaciones según permisos)

Esta historia de usuario permitió implementar la lógica de validación de acceso a las distintas funcionalidades de la plataforma utilizando el modelo RBAC. Se emplearon tokens JWT con payload enriquecido que contiene los roles y permisos del usuario.

La estructura del token de acceso devuelta por el backend, para que la plataforma móvil pueda decodificarla y usarla, se muestra en *Figura 31*.



Figura 31. Estructura del token de acceso (Parcialmente oculta)

Para entender un poco el flujo que toma la aplicación para validar el acceso, se puede observar en el diagrama de secuencia de *Figura 32*. Diagrama de secuencia - Acceso a módulo según permisos.

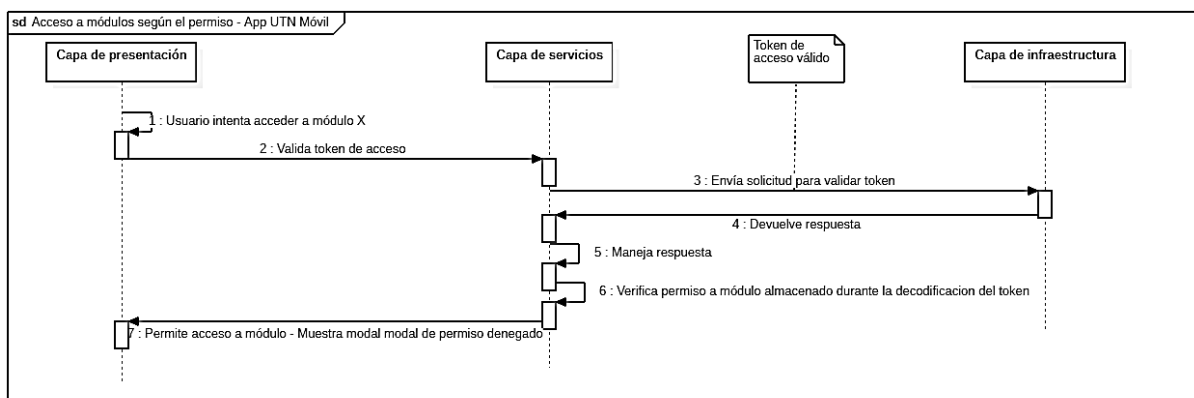


Figura 32. Diagrama de secuencia - Acceso a módulo según permisos

Además, los endpoints fueron protegidos mediante guards personalizados, asegurando que solo los usuarios con los permisos adecuados puedan acceder. En casos específicos donde se requiere

acceso libre, se emplearon decoradores como `@Public`, mostrado en *Figura 33*. Decorador `Public`, para marcar explícitamente las rutas exentas a autenticación o algún permiso establecido.

```
1 import { SetMetadata } from '@nestjs/common';
2
3 export const IS_PUBLIC_KEY = 'isPublic';
4 export const Public = () => SetMetadata(IS_PUBLIC_KEY, true);
5 |
```

Figura 33. Decorador `Public`

En *Figura 34* se puede observar el modal que se muestra al usuario cuando intente acceder a un módulo no autorizado.

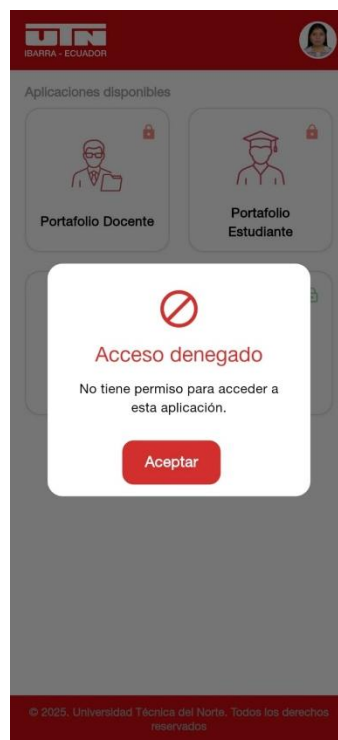


Figura 34. Modal mostrado cuando usuario no tiene permiso para x módulo

- Cargar acceso a apps dinámicamente en el Home (HU-SEC-11: Carga de apps en el home)

Este incremento permitió implementar un sistema dinámico de cargar los botones para acceder a los módulos en la pantalla principal de la plataforma UTN Móvil. A partir del token JWT decodificado tras el inicio de sesión, se extraen los permisos y nombres de las aplicaciones que el usuario tiene habilitadas, y se renderizan automáticamente en el Home según dichos datos.

La lógica que se implementó garantiza que cada usuario visualice únicamente las aplicaciones a las que tiene acceso, mejorando la personalización, seguridad y usabilidad de la plataforma. Esto también evita el hardcoding de accesos, facilitando el mantenimiento del sistema en caso de cambios futuros en los permisos. De igual forma, se incorporó un archivo de configuración, ilustrado en Figura 35, que actúa como registro centralizado de los módulos disponibles. Este archivo permitió definir el nombre, ícono, ruta inicial, orden de visualización de los botones de acceso en el Home, facilitando la organización y personalización de la interfaz.

```
class ApplicationsConstants {
    static const Map<String, ApplicationInfo> apps = {
        'APP_ESTUDIANTE': ApplicationInfo(
            label: 'Portafolio Estudiante',
            iconPath: 'assets/icons/estudiante_home.png',
            route: '/estudiante/calendario-actividades',
        ),
        'APP_DOCENTE': ApplicationInfo(
            label: 'Portafolio Docente',
            iconPath: 'assets/icons/docente_home.png',
            route: '/docente/registro-academico',
        ),
        'APP_SETTINGS': ApplicationInfo(
            label: 'Acerca de',
            iconPath: 'assets/icons/about_home.png',
            route: '/settings',
        ),
        'APP_EXTERNA': ApplicationInfo(
            label: 'Waze',
            iconPath: 'assets/icons/actividades-docente.svg',
            route: 'waze',
        ),
    };

    static ApplicationInfo getApp(String key) {
        return apps[key] ??
            const ApplicationInfo(
                label: 'App Desconocida',
                iconPath: 'assets/icons/list.svg',
                route: '/',
            );
    }

    static List<String> getAvailableAppKeys(Set<String> permissions) {
        return apps.keys
            .where((key) => permissions
                .any((perm) => perm.contains(key.replaceAll(
```

Figura 35. Archivo *applications_constants*

A continuación, en Figura 36 se muestra un ejemplo visual del Home, donde las tarjetas de acceso a las aplicaciones aparecen en función de los permisos cargados dinámicamente.



Figura 36. Pantalla 'Home' para aplicación UTN Móvil

Además, es importante mencionar que ciertas secciones de la aplicación, como *Acerca de* y *Paseo Virtual*, son accesibles de forma pública, sin necesidad de iniciar sesión. Sin embargo, en el caso de *Acerca de*, el contenido visual se adapta dinámicamente cuando el usuario ha iniciado sesión, mostrando apartados adicionales. Esta diferencia se puede observar en Figura 37.



Figura 37. Pantalla 'Acerca de' cuando el usuario inició sesión

- Acceso como invitado sin autenticación (HU-SEC-10: Acceso como invitado sin autenticación)

Este incremento permitió habilitar una opción de ingreso a la plataforma UTN Móvil como invitado, es decir, sin la necesidad de pasar por el proceso de autenticación tradicional. Esta funcionalidad fue desarrollada para ofrecer acceso limitado a ciertos módulos informativos o públicos, mejorando la accesibilidad y permitiendo que usuarios no pertenecientes a la organización exploren ciertas funcionalidades básicas de la plataforma.

A continuación, en Figura 38 se muestra un ejemplo de la pantalla principal cuando el usuario ha ingresado como invitado.

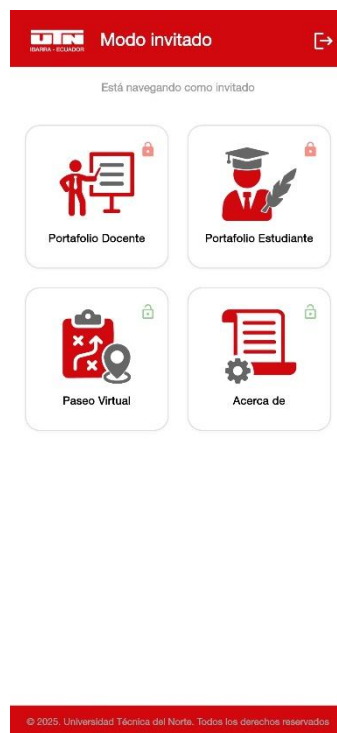


Figura 38. Pantalla 'Home' de modo invitado

Como se mencionó en el incremento anterior, ciertas secciones de la plataforma modifican su apariencia y contenido en función del estado de sesión del usuario. En el caso específico del modo invitado, la sección *Acerca de* presenta una versión reducida y adaptada, que oculta información sensible o específica del perfil autenticado. En Figura 39 se puede observar la vista correspondiente a esta sección cuando se accede sin autenticación.



Figura 39. Pantalla 'Acerca de' para el modo invitado

III. Revisión

Durante el Sprint 3 se verificó el cumplimiento de los criterios de aceptación correspondiente a las tres historias de usuario seleccionadas para este ciclo. A través de pruebas funcionales, revisión de lógica condicional y validación de comportamiento en diferentes escenarios, se confirmó que las funcionalidades implementadas cumplen con lo establecido. En Tabla 28 se resumen el cumplimiento de los criterios de aceptación.

Tabla 28. Revisión de criterios de aceptación involucrados en Sprint 3

HISTORIA DE USUARIO	CRITERIO DE ACEPTACIÓN	CUMPLE
	CA1: La aplicación debe validar los permisos de cada usuario para controlar el acceso.	SI

HU-SEC-06: Acceso a aplicaciones según permisos	CA2: Si el usuario intenta acceder a una aplicación sin tener los permisos correspondientes, se debe mostrar un mensaje de acceso denegado.	SI
HU-SEC-11: Carga de apps en el home HU-SEC-04: Manejo de tokens expirados o inválidos	CA1: La lista de las aplicaciones se construye en base a un registro de apps disponibles.	SI
	CA2: Aplicaciones externas, como Paseo Virtual UTN se redirigirán de forma automática a la Store de descarga sino la posee o abrirá la app.	SI
	CA3: El listado de las aplicaciones se utilizará tanto en el modo invitado como en el usuario que inició sesión.	SI
HU-SEC-10: Acceso como invitado sin autenticación	CA1: El modo invitado debe solo permitirme ingresar a las aplicaciones o enlaces públicos.	SI
	CA2: Si el invitado intenta acceder a funciones o aplicaciones protegidas se le mostrara un mensaje de que no está autorizado.	SI
	CA3: El modo invitado no registra ninguna actividad en el servicio de seguridad.	SI

IV. Retrospectiva

En este sprint se reconocieron avances importantes en la personalización, accesibilidad y diseño visual de la plataforma UTN Móvil. Uno de los avances más destacados fue la implementación dinámica del Home, que permite cargar las aplicaciones disponibles en función de los permisos otorgados al usuario. Esta mejora no solo optimizó la experiencia de navegación, sino que también fortaleció la seguridad mediante una gestión granular de acceso basado en roles.

Se valoró positivamente la incorporación del modo invitado debido a que extiende el alcance de la aplicación porque permite ingresar a módulos públicos sin la necesidad de autenticarse.

Otro punto que demandó una cantidad considerable de tiempo y planificación fue la ampliación de la sección *Acerca de*, donde se integró contenido institucional relevante, marco legal, contacto y atribuciones. Esta sección fue adaptada para mostrar contenido dinámico en función del estado del usuario, implicando también la organización y estructura visual del mismo.

Como área de mejora se refactorizó componentes compartidos y se aplicó una mayor reutilización de widgets ara evitar duplicación de lógica y estilos.

2.2.5. Sprint 4

I. *Planificación General*

La ejecución del sprint 4 aborda la garantía de la seguridad, la accesibilidad y la continuidad de la experiencia del usuario en la plataforma. Las historias de usuario se centraron en el manejo de autenticación en condiciones específicas como el cambio de contraseña, ala autenticación biométrica local y la verificación de conectividad durante el uso de la aplicación. Se priorizaron las funcionalidades acordes al Product Backlog mostrado en *Tabla 20*,.

El sprint tiene una planificación de 30 horas totales. Las primeras 25 horas son destinadas al desarrollo de las tareas programadas. Las 4 horas restantes se asignaron como margen de contingencia.

A continuación, en Tabla 29 se presentan las tareas planificadas

Tabla 29. *Sprint Backlog - Sprint 4*

HU	NOMBRE	TAREAS	HORAS
HU-SEC-02	Cambio de contraseña de Microsoft	Desarrollar lógica para redireccionar al cambio de contraseña	1.5
		Implementar widget de redireccionamiento	1.5
HU-SEC-07	Autenticación biométrica	Diseñar flujo de autenticación biométrica	3
		Desarrollar lógica para autenticación biométrica local	5
		Implementar pantalla de autenticación biométrica	3
HU-SEC-09	Verificación de conexión a internet.	Diseñar flujo de verificación de conexión a internet	3
		Desarrollar lógica para verificar la conexión a internet	5
		Implementar pantalla de verificación de conexión a internet	3
Total: 25 horas de implementación + 5 horas de ajuste = 30 horas			

II. *Incremento*

El incremento correspondiente a este sprint representó un paso clave en la mejoría de la experiencia de autenticación dentro de la plataforma UTN Móvil. Las historias abordadas en este sprint

integran capacidades como redirección al portal oficial de cambio de contraseña de Microsoft, validación de conectividad previa al login y autenticación biométrica local en dispositivos compatibles.

A continuación, se describen los resultados obtenidos para cada historia de usuario trabajada durante este sprint:

- Cambio de contraseña de Microsoft (HU-SEC-02: Cambio de contraseña de Microsoft)

Este incremento implementó la funcionalidad que permitió a los usuarios que tienen credenciales institucionales de Microsoft redirigirse de forma segura a la plataforma oficial de restablecimiento de contraseña. Esta medida responde a necesidades específicas, como que el usuario no recuerda su contraseña y entonces se le ofrece un acceso rápido al portal de Microsoft para cambiar su contraseña. La funcionalidad se ha incorporado como un botón en la pantalla de login de la aplicación UTN Móvil, tal como se muestra en Figura 40.



Figura 40. Pantalla de 'Login' incluido botón de redireccionamiento para cambio de contraseña

- Autenticación biométrica (HU-SEC-07: Autenticación biométrica)

Este incremento introdujo la funcionalidad de autenticación biométrica como capa adicional de seguridad dentro de la plataforma UTN Móvil. A partir de este Sprint, se requiere validación biométrica, como huella digital, pin, patrón, reconocimiento facial u otra tecnología compatible, en el acceso a módulos protegidos por permisos especiales.

En Figura 41 se puede observar el diagrama de secuencia para conocer el flujo que toma esta autenticación.

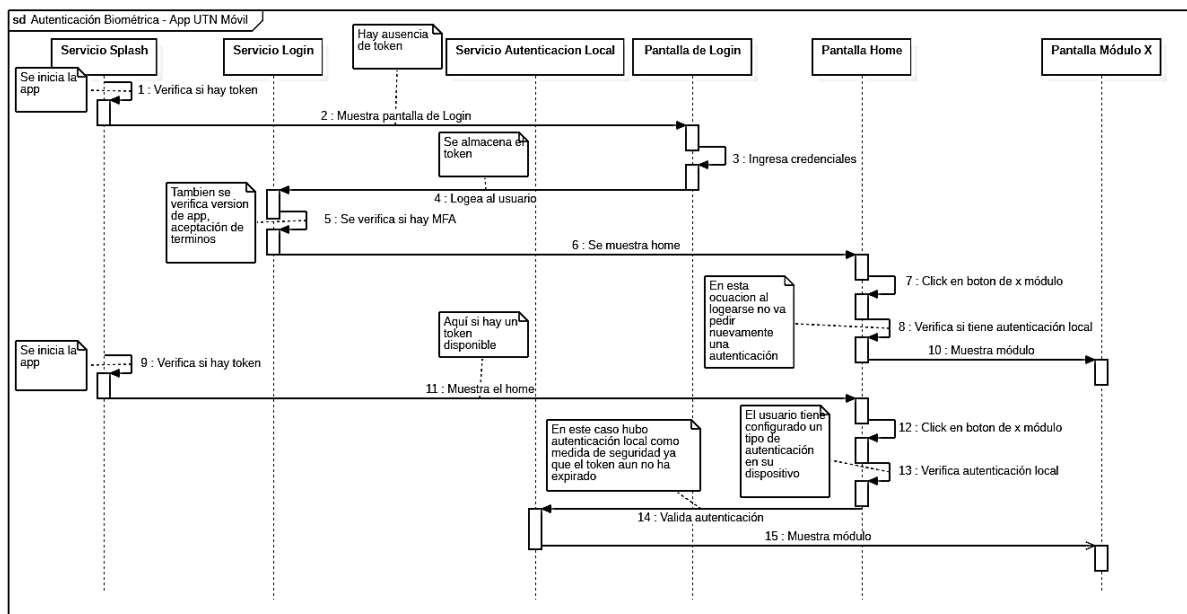


Figura 41. Diagrama de secuencia - Autenticación en UTN Móvil

En caso de que falle la validación biométrica, el acceso es denegado de inmediato, impidiendo acceder y cargar el contenido protegido y asegurando que solo el titular legítimo del dispositivo y sesión pueda continuar. Esta lógica se implementó completamente en el cliente Flutter, aprovechando bibliotecas compatibles con Android e iOS.

Cabe mencionar que esta autenticación se realizó de forma completamente local y no expone ningún dato biométrico fuera del dispositivo. La funcionalidad solo se activa si el usuario ha configurado previamente su método de autenticación local en su dispositivo. En Figura 42 se muestra la pantalla usada para este tipo de autenticación.

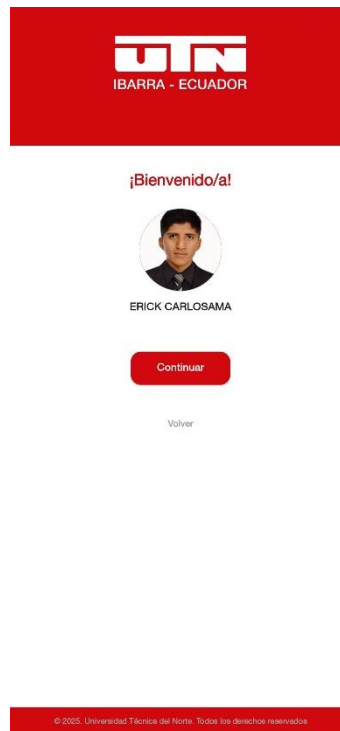


Figura 42. Pantalla de 'Autenticación biométrica local'

- Verificación de conexión a internet (HU-SEC-09: Verificación de conexión a internet)

Este incremento introdujo un mecanismo proactivo de verificación de conectividad a internet dentro de la plataforma UTN Móvil, con el objetivo de evitar errores al intentar autenticar usuarios o consultar datos sin conexión disponible.

La lógica de validación ha sido integrada directamente en todos los flujos de la aplicación, comprueba el estado actual de red y bloquea cualquier operación si no hay conexión disponible. Esta operación es ejecutada desde el inicio y durante el uso de la aplicación. Si la red restaura la conexión, se detecta automáticamente este cambio de estado de red permitiendo al usuario continuar con normalidad sin necesidad de reiniciar la aplicación.

A continuación, en *Figura 43* se muestra el diagrama de secuencia de firma simple para entender un poco el flujo de esta verificación.

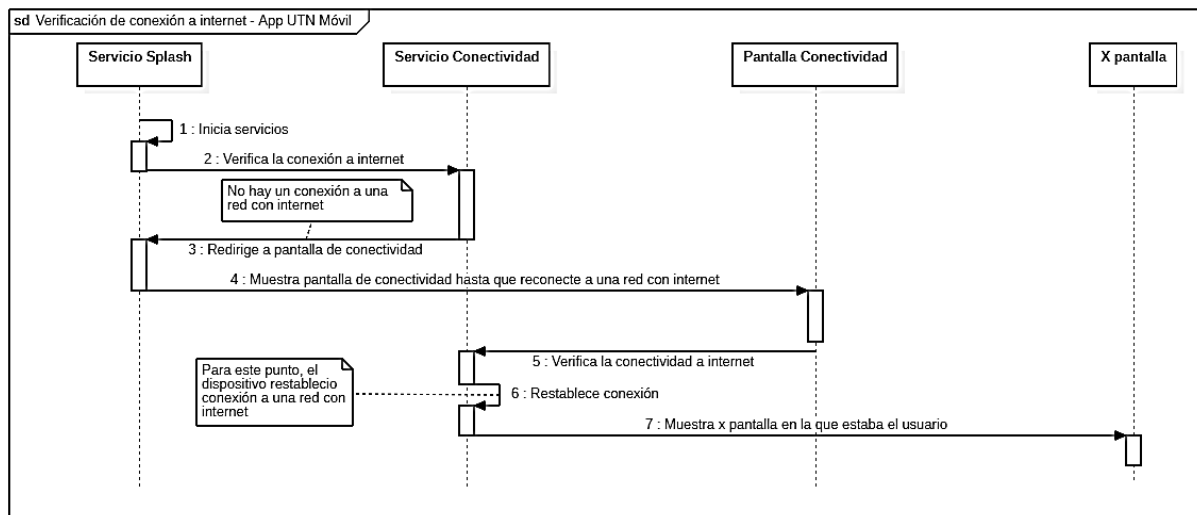


Figura 43. Diagrama de secuencia - Verificación de conexión a internet

La pantalla para la detección de conectividad se puede observar en *Figura 44*, esta es mostrada automáticamente cuando no hay conectividad a internet. La pantalla cumple dos funciones fundamentales, por un lado, informa al usuario de forma clara sobre la falta de conexión y por otro lado ofrece un botón para reintentar la verificación manualmente, aunque también se realiza automáticamente.

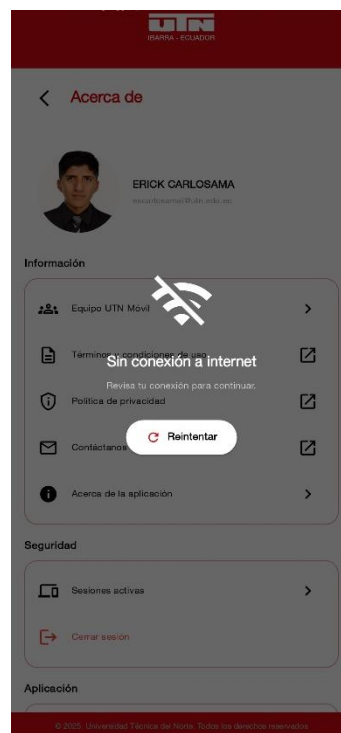


Figura 44. Pantalla de 'Conectividad'

III. Revisión

Para estos sprint se revisó el cumplimiento de los criterios aceptación de cada historia de usuario. A continuación, en *Tabla 30* se muestra un resumen del cumplimiento de estos criterios de aceptación.

Tabla 30. Revisión de criterios de aceptación en Sprint 4

HISTORIA DE USUARIO	CRITERIO DE ACEPTACIÓN	CUMPLE
HU-SEC-02: Cambio de contraseña de Microsoft	CA1: Debe haber un enlace en la aplicación que redirija a la página de cambio de contraseña de Microsoft.	SI
	CA2: La redirección debe abrirse en el navegador predeterminado del dispositivo.	SI
HU-SEC-07: Autenticación biométrica HU-SEC-04: Manejo de tokens expirados o inválidos	CA1: Cada acceso a una app con permisos protegidos debe requerir autenticación biométrica local.	SI
	CA2: Si la autenticación falla, no se deberá permitir el acceso de ninguna manera.	SI
	CA3: El uso de la autenticación biométrica es local, y solo se activará si el usuario tiene activada dicha configuración en su dispositivo.	SI
HU-SEC-09: Verificación de conexión a internet	CA1: La aplicación debe detectar si no hay conexión antes de realizar llamadas a la API.	SI
	CA2: En caso de no haber conexión, debe mostrar un widget informativo.	SI
	CA3: La aplicación deberá detectar automáticamente si se restableció la conexión a internet y permitir seguir interactuando en la aplicación.	SI
	CA4: Desde el widget informativo debe ofrecer la opción de verificar si se restableció el internet	SI

IV. Retrospectiva

Para este sprint se destacó el éxito general de las funcionalidades implementadas, con especial énfasis en la autenticación biométrica, que fue una de las características más valoradas por los usuarios. Esta implementación, que permiten a los usuarios acceder a las aplicaciones protegidas sin necesidad de ingresar credenciales en cada uso de la app, ha mejorado significativamente la

experiencia de uso. Los usuarios ahora solo necesitan autenticarse cuando el token de refresco ha caducado, lo que optimiza el tiempo y hace la plataforma más amigable. Esta característica fue bien recibida y mejoró la percepción general sobre la seguridad y usabilidad de la aplicación.

Podemos decir que este sprint fue exitoso, y cumplido en los tiempos establecidos. Se afinaron detalles menores en los tiempos de desfase que hay en cada sprint.

2.2.6. Sprint 5

I. Planificación General

Las funcionalidades abordadas en este sprint fueron críticas para contribuir la integración completa de la plataforma y la mejora de la experiencia de los usuarios.

Durante este sprint se contempló un entorno de pre-producción de forma local. Se utilizó Docker y pruebas automatizadas para garantizar la calidad del código antes de su despliegue. Finalmente, también se contempló en este sprint, una interfaz de configuración de parámetros del módulo de autenticación y autorización.

El Sprint Backlog correspondiente a este sprint se ha detallado en *Tabla 31*. La estimación para este sprint es de un total de 50 horas, 45 horas fueron destinadas al desarrollo efectivo de las tareas programadas. Las 5 horas restantes se reservaron como estrategia de contingencia para abordar ajustes imprevistos o refinamientos.

A continuación, se detallan las tareas planificadas:

Tabla 31. *Sprint Backlog - Sprint 5*

HU	NOMBRE	TAREAS	HORAS
Integración de Microservicios	Integrar completamente microservicios con el Gateway	Registrar módulos en archivos y .env del API Gateway	1
		Definir puertos, rutas y API Keys para cada módulo	1
HU-SEC-14	Control de versión de la app	Diseñar flujo de control de versión de app	2
		Desarrollar lógica para controlar versión de app	5
		Implementar pantalla de control de versión de app	2
Afinar registro de sesiones	Afinar registro de sesiones con datos adicionales (ubicación, alias, etc)	Implementar registro de datos adicionales en sesiones (Backend)	4
		Implementar envío de datos adicionales para sesiones (Frontend)	4

Unificar diseños Flutter	Unificar diseño Flutter: Home, Login, Acerca de, etc.	Establecer diseño unificado en Home, Login y Acerca de (componentes visuales como colores, fuentes, estilos, etc)	3
Establecer entorno de pre-producción	Establecer entorno de pre-producción (Docker, pruebas)	Crear y configurar entorno de pre-producción con Docker	4
		Crear las imágenes de Docker por cada módulo a probar utilizando Docker File	2
		Realizar pruebas directas al módulo, desde el API Gateway y desde el cliente flutter.	4
HU-SEC-08	Desarrollar interfaz de configuración desde frontend admin	Desarrollar interfaz para administrar configuraciones desde frontend	8
		Testear funcionamiento	3
		Integrar colores y diseños similares a la app móvil	2
Total: 45 horas de implementación + 5 horas de ajuste = 50 horas			

II. *Incremento*

El Incremento correspondiente para este Sprint 5 representó un avance importante en la consolidación técnica y visual de la plataforma UTN Móvil. En este sprint se realizó la integración completa de todos los microservicios dentro de la API Gateway, se implementó el control de versión para la app móvil, y se realizó ajustes que fortalecen tanto la trazabilidad de sesiones como la experiencia visual del usuario. Estas mejoras permitieron que el sistema opere de forma más robusta, escalable y coherente con los lineamientos institucionales.

A continuación, se presentan los resultados obtenidos para cada historia de usuario y tarea técnica ejecutada durante este sprint:

- Integración completa de microservicios al Gateway

Durante este incremento se completó la configuración del componente API Gateway para gestionar correctamente el redireccionamiento de solicitudes hacia todos los microservicios disponibles en la plataforma UTN Móvil. Para ello, se actualizaron los archivos de configuración, `services.config.ts` y el archivo `.env`, incorporando los nombres, puertos, ruta base y claves de autenticación (API Key) de cada uno de los módulos que forman parte del sistema distribuido.

En Figura 45 se muestra un fragmento en donde se implementó la lógica de carga e inicialización dinámica de los servicios configurados. El archivo permite registrar las URL base y API Keys de cada uno de los microservicios. La estrategia es que funcione como un proxy inverso completamente desacoplado. La implementación de esta funcionalidad facilitó la escalabilidad del sistema, ya que integra nuevos servicios añadiendo su configuración correspondiente. La validación de rutas y calves

también fue contemplado, con el fin de que únicamente se establezcan comunicación con microservicios válidos.

```
private async loadServicesConfig() {

    services.estudiante.baseUrl = this.configService.get<string>('ESTUDIANTE_BASE_URL');
    services.estudiante.apiKey = this.configService.get<string>('ESTUDIANTE_API_KEY');

    services.docente.baseUrl = this.configService.get<string>('DOCENTE_BASE_URL');
    services.docente.apiKey = this.configService.get<string>('DOCENTE_API_KEY');
```

Figura 45. Fragmento del archivo *app.module.ts* que registra dinámicamente los microservicios en el Gateway

- Control de versión de la app (HU-SEC-14: Control de versión)

El incremento asociado a esta historia de usuario introdujo un mecanismo automatizado de verificación de versión para la aplicación UTN Móvil. Esta funcionalidad garantizó que los usuarios utilicen siempre una versión actualizada que cumple con los requerimientos mínimos de funcionamiento y seguridad.

En Figura 46 se muestra el flujo de esta operación.

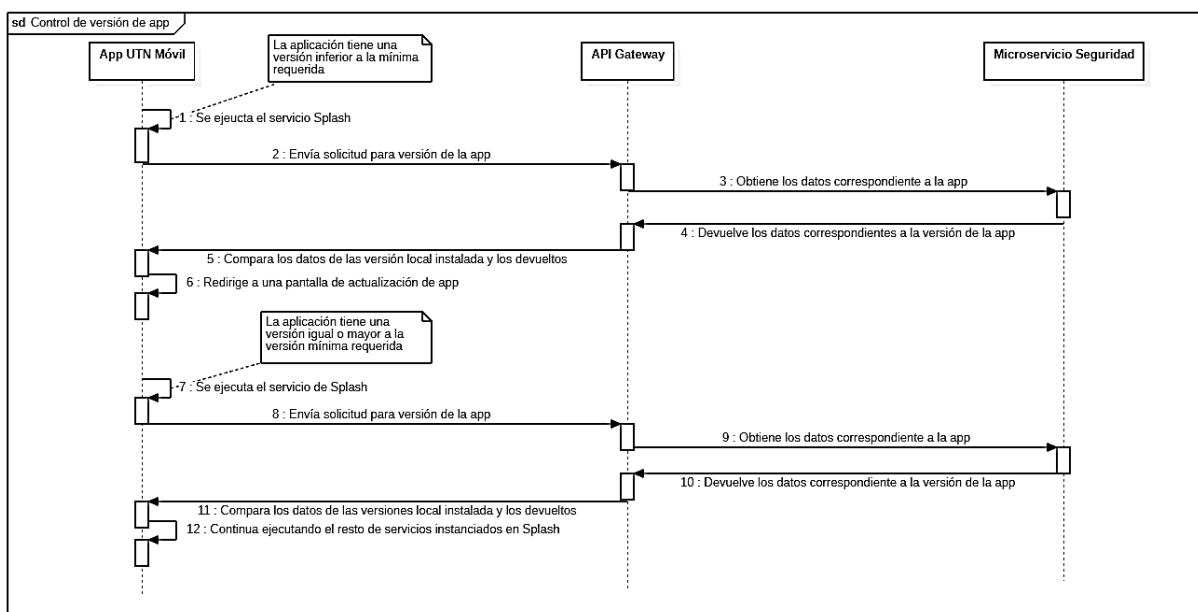


Figura 46. Diagrama de secuencia - Control de versión

En el caso de que la versión instalada sea inferior a la mínima permitida y la verificación esté activa, se despliega una pantalla especial bloqueando el acceso, como se ilustra en Figura 47.



Figura 47. Pantalla 'Actualización de versión'

Adicional a esta operación, se habilitó una opción para activar o desactivar la verificación de versión, permitiendo a los administradores establecer períodos de mantenimiento o despliegue controlado.

- Afinar registro de sesiones con datos adicionales.

Como parte de la mejora continua del módulo de sesiones activas, se realizó un ajuste que permita registrar y visualizar información adicional relevante en cada sesión de usuario, tales como ubicación, dirección IP, alias del dispositivo y otros metadatos del entorno desde el cual se inició la sesión.

Del lado del backend, se modificó la lógica de almacenamiento de sesiones para incluir campos adicionales como ubicación, ip, alias del dispositivo, entre otros. Estos datos se integraron a la entidad `sessions_app`, y su persistencia quedó habilitada para ser consultada en tiempo real o mediante auditoría posterior. El envío de esta información adicional desde el cliente Flutter, al momento de crear o renovar una sesión, se observa en el fragmento de código mostrado en Figura 48.

```

Future<void> login(
  String email, String password, BuildContext context) async {
  try {
    String encodedPassword = [REDACTED];
    final deviceInfo = await _deviceInfoService.getDeviceInfo();
    final [REDACTED];
    final [REDACTED];

    final token = await _loginRepository.login(
      email: email,
      encodedPassword: encodedPassword,
      [REDACTED],
      [REDACTED],
    );
  }
}

```

Figura 48. Fragmento de código para envío de datos adicionales de sesión desde Flutter

Además, en la vista de SessionsPage, ya implementada en sprints anteriores, se adaptó la interfaz para mostrar estos nuevos atributos de manera visualmente amigable y comprensible. En Figura 49 se puede apreciar un ejemplo con los datos adicionales ya reflejados por sesión activa.

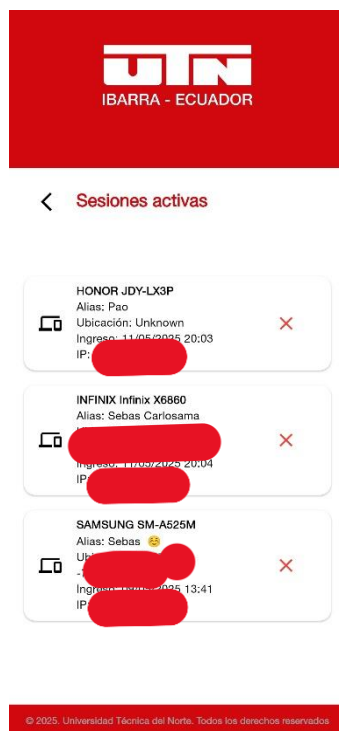


Figura 49. Pantalla de 'Gestión de sesiones'

Este ajuste permitió obtener mayor contexto sobre las sesiones iniciadas, lo que incrementa la trazabilidad, la transparencia y la seguridad general del sistema.

- Unificar diseño visual en Flutter

Como parte de este incremento se procedió a estandarizar la estética general de la plataforma UTN Móvil. Esto incluyó la unificación del esquema de colores, uso de tipografías, espaciado entre elementos, estructura de encabezados y disposición de botones, esto se puede observar en Figura 50.



Figura 50. Pantalla 'Home'

- Establecer entorno de pre-producción (Docker y pruebas)

Se configuró un entorno de pre-producción local que utiliza contenedores Docker, lo que permitió simular el comportamiento de la plataforma en condiciones similares al entorno real. El entorno permitió ejecutar pruebas controladas con el fin de verificar que los módulos respondan correctamente a distintos escenarios de autenticación, validación y redireccionamiento. Como se muestra en Figura 51, se puede observar la ejecución del entorno con una composición básica de servicios.

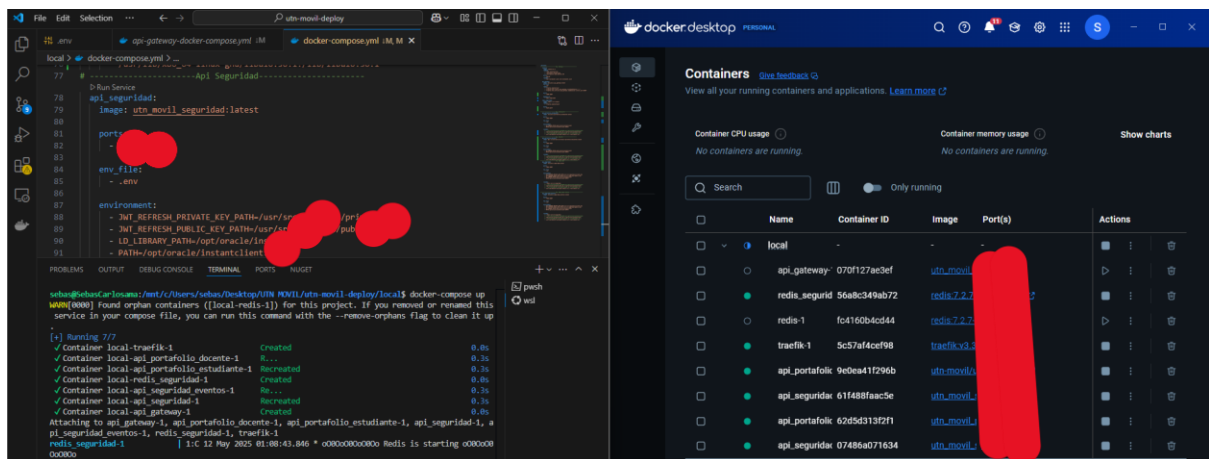


Figura 51. Composición de servicios en entorno Docker para pre-producción

- Frontend Administrativo (HU-SEC-08: Configuración de parámetros generales desde fronted administrativo)

Este incremento introdujo una interfaz visual orientada a usuarios administradores, desde la cual es posible gestionar los parámetros generales del módulo de seguridad. La interfaz ha sido desarrollada como parte del fronted administrativo de la plataforma, brindando acceso exclusivo a los valores que anteriormente solo podían ser modificados desde la base de datos o mediante código fuente.

En Figura 52 podemos observar el formulario para editar los parámetros generales utilizados en la plataforma móvil.

Figura 52. Pantalla 'Formulario para parámetros generales'

De igual forma, en Figura 53 tenemos otra vista para ver todos parámetros generales dentro de una tabla.

ID	Nombre	Descripción	Valor	Estado	Creado	Modificado	Acciones
1	Microsoft OAuth URL			Activo	3/19/25, 12:00 AM	4/28/25, 12:00 AM	[Iconos]
2				Activo	3/19/25, 12:00 AM	4/28/25, 12:00 AM	[Iconos]
3				Activo	3/19/25, 12:00 AM	4/28/25, 12:00 AM	[Iconos]
4				Activo	3/19/25, 12:00 AM	5/21/25, 12:00 AM	[Iconos]
5				Activo	3/19/25, 12:00 AM	5/21/25, 12:00 AM	[Iconos]
6				Activo	3/19/25, 12:00 AM	5/21/25, 12:00 AM	[Iconos]

Figura 53. Pantalla 'Listado de parámetros generales'.

La sección de sesiones es otra de las vistas relevantes ya que tenemos, por un lado, las sesiones activas en ese momento como se observa en Figura 54.

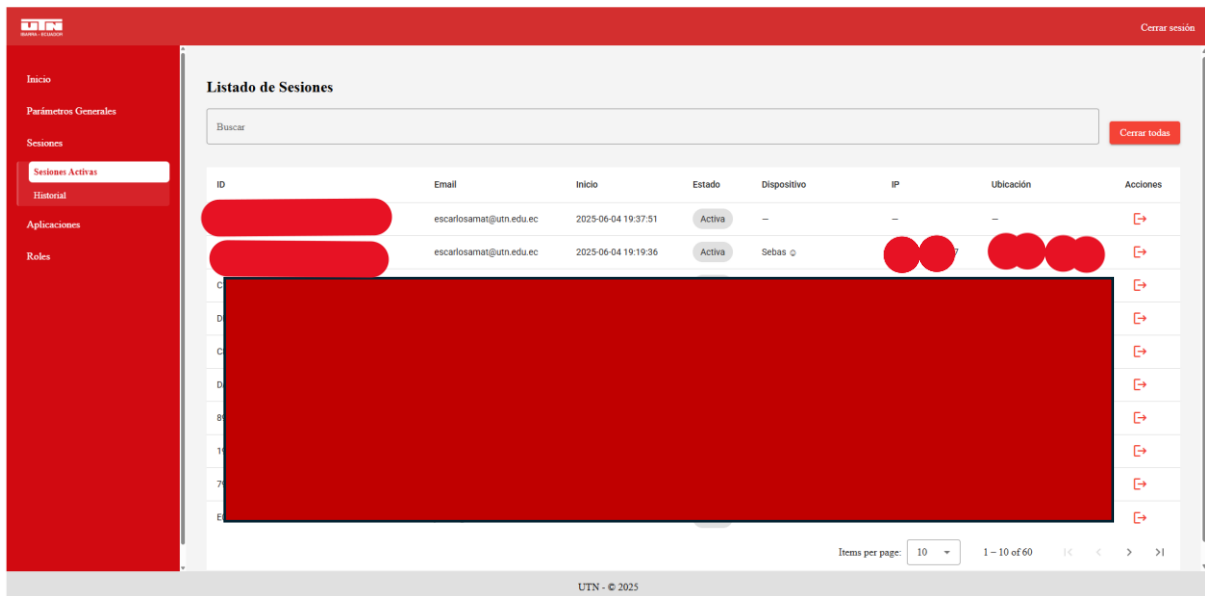


Figura 54. Pantalla 'Listado de sesiones activas'

Y también tenemos el historial de sesiones como se observa en Figura 55.

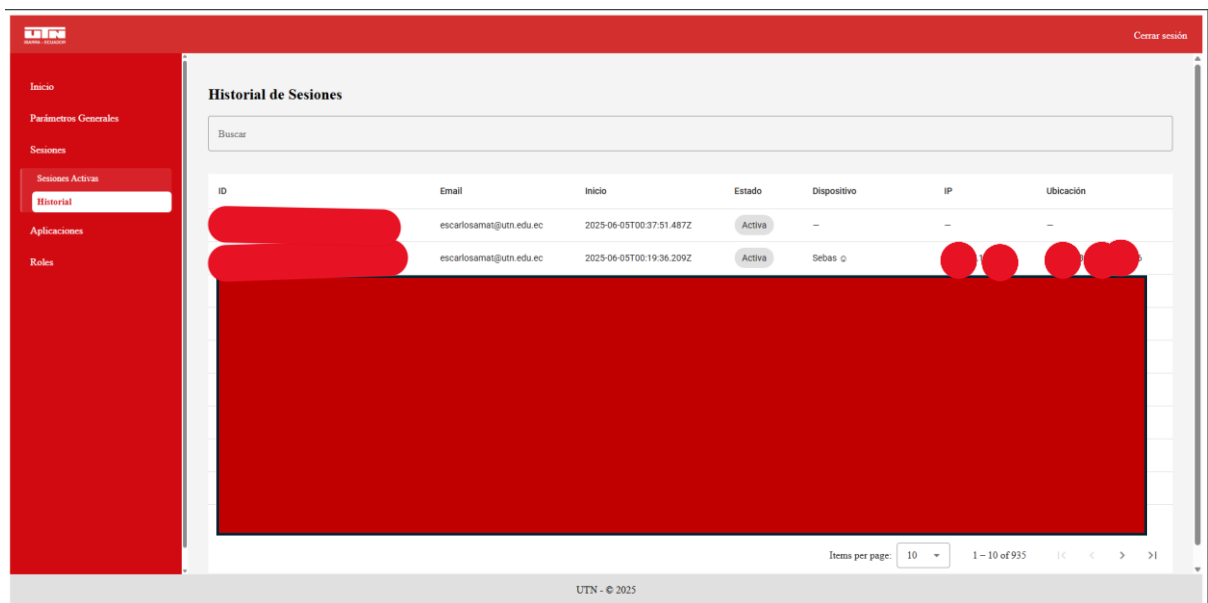


Figura 55. Pantalla 'Historial de sesiones'

La sección de Aplicaciones es igual de relevante, pues permite agregar nuevos módulos y funciones relacionadas al módulo como se observa en Figura 56.

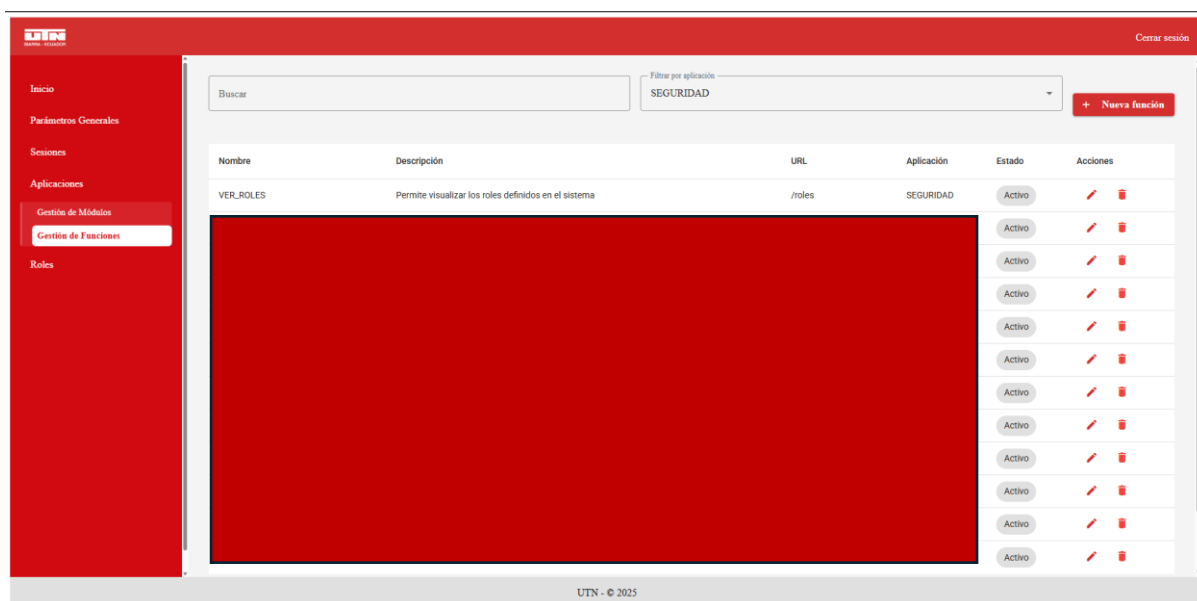


Figura 56. Pantalla 'Gestión de Funciones'

Y relacionada a esto, está la gestión de roles de la plataforma UTN Móvil, la asignación de esos roles a usuarios y la asignación de esas funciones a los roles tal como se puede observar en *Figura 57*.

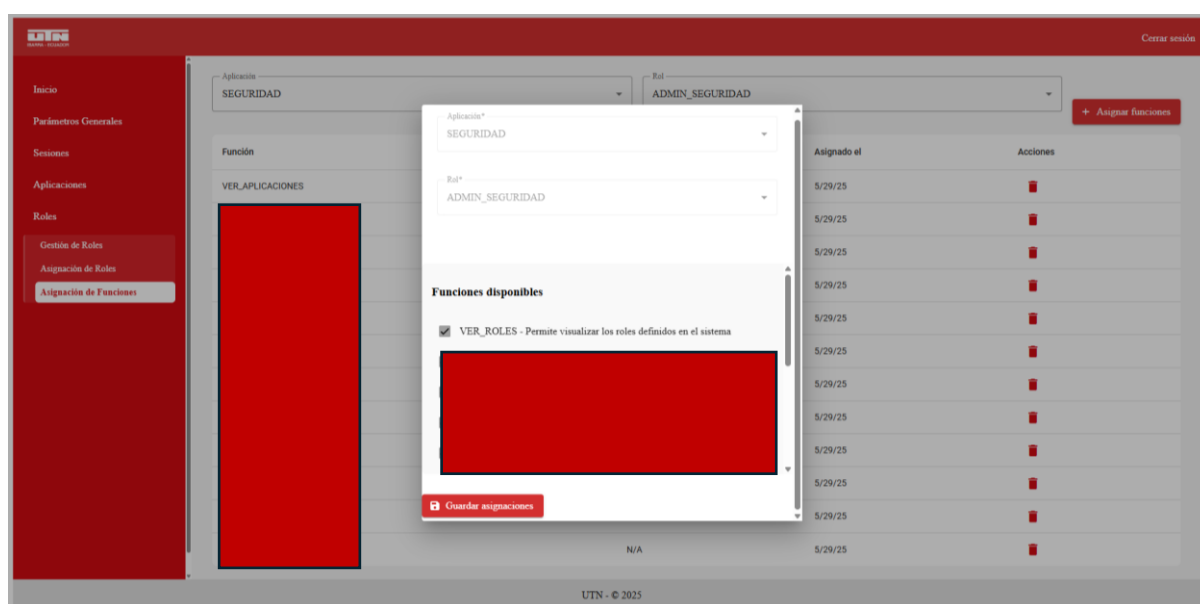


Figura 57. Pantalla 'Asignación de funciones'

III. Revisión

Durante este sprint se validaron satisfactoriamente los criterios de aceptación correspondientes a las historias de usuario y ajustes técnicos contemplados. A continuación, en Tabla 32 se detalla el cumplimiento de cada uno con base en las pruebas funcionales y de integración realizados.

Tabla 32. Revisión de criterios de aceptación en Sprint 5

HISTORIA DE USUARIO / TAREA	CRITERIO DE ACEPTACIÓN	CUMPLE
Integración de microservicios	Los microservicios fueron registrados dinámicamente en el archivo de configuración del API Gateway, incluyendo base URL y claves API.	SI
HU-SEC-14: Control de versión de la app HU-SEC-04: Manejo de tokens expirados o inválidos	CA1: Verificación automática al iniciar la app	SI
	CA2: Redirección a pantalla de actualización si no cumple versión mínima	SI
	CA3: Backend expone endpoint con campos version_actual, version_minima, verificar_version y mensaje_actualizacion	SI
	CA4: Parámetro verificar_version editable desde panel de configuración	SI
	CA5: Flujo normal si la versión es válida	SI
Afinar registro de sesiones	Registro y visualización de alias, IP y marca del dispositivo por sesión	SI
Unificación de diseño en Flutter	Estilos consistentes en Home, Login y Acerca de	SI
Entorno de pre-producción (Docker)	Entorno configurado con imágenes Docker para pruebas locales	SI
HU-SEC-08: Interfaz de configuración administrativa	CA1: Permite modificar parámetros como tokens, claves y modo mantenimiento	
	CA2: Solo accesible para administradores autenticados	

IV. Retrospectiva

Este sprint marcó la culminación de una etapa clave en desarrollo de la plataforma UTN Móvil, consolidando no solo funcionalidades pendientes, sino también mejorando la experiencia administrativa y de despliegue. Uno de los principales logros fue la integración completa de los microservicios con el API Gateway. Esta integración, ahora gestionada dinámicamente a través de archivos .env y estructuras de configuración centralizada, permite mantener un sistema flexible, escalable y seguro.

Se destacó la implementación del control de versión de la aplicación móvil. La funcionalidad refuerza el cumplimiento de estándares de seguridad y garantiza una experiencia.

Desde el punto de vista técnico, la configuración del entorno de pre-producción permitió realizar pruebas de integración reales sin afectar el entorno de producción. De igual forma la creación de una interfaz administrativa genera una mejora en la gestión del sistema.

Este sprint no solo cerró con un backlog completamente funcional, sino que sentó las bases para futuras fases de mantenimiento, monitoreo y escalamiento de la plataforma.

2.3. Desarrollo de Fase III: Post-Juego

Durante esta última etapa del proyecto, se completaron las actividades fundamentales para la estabilización, documentación y validación del sistema desarrollado. Esta fase tuvo como objetivo principal consolidar el prototipo funcional de la plataforma UTN Móvil, asegurando su funcionamiento estable y su facilidad de transferencia tecnológica hacia futuros equipos.

Se llevaron a cabo pruebas funcionales completas sobre varias las historias de usuario implementadas. Las validaciones se ejecutaron en un entorno controlado, utilizando la arquitectura completa del sistema. Como resultado, se confirmó el cumplimiento de todos los requisitos funcionales definidos en las fases previas.

De forma paralela, se realizó el despliegue de la infraestructura mediante Docker, lo que permitió encapsular cada microservicio. Para esta tarea, se desarrollaron archivos DockerFile específicos por módulo y un docker-compose.yml que orqueste la infraestructura, lo cual simplifica la instalación y ejecución del sistema tanto en entornos de pruebas como en producción.

Se recomendó implementar sesiones de capacitación o transferencia de conocimiento mediante herramientas colaborativas, con el fin de garantizar la adopción del sistema por parte de nuevos equipos o responsables institucionales.

III. CAPÍTULO 3: VALIDACION DE RESULTADOS

3.1. Definición de criterios de evaluación de seguridad

Con el objetivo de validar la seguridad de solución tecnológica desarrollada para la plataforma UTN Móvil, se utilizó como referencia estándar internacional OWASP Top Ten 2021, el cual reúne las diez categorías más críticas de vulnerabilidades en aplicaciones modernas. La evaluación se llevó a cabo mediante pruebas controladas de penetración, validación de configuraciones, revisión de componentes y análisis del comportamiento de la aplicación frente a ataques comunes.

A continuación, se presenta en *Tabla 33* la alineación entre cada categoría OWASP Top Ten, los aspectos evaluados, las herramientas aplicadas y evidencias generadas durante el proceso de validación.

Tabla 33. Evaluación de seguridad basada en OWASP Top Ten y herramienta Kali Linux

Categoría OWASP 2021	¿QUÉ SE EVALÚA?	HERRAMIENTA	EVIDENCIA ESPERADA / ACTIVIDAD
A01	Control de acceso deficiente	Burp Suite	Pruebas de acceso no autorizado a endpoints protegidos, validación de RBAC
A02	Uso débil o incorrecto de cifrado, mala gestión de claves	Visual Studio Code	Revisión de tokens y datos sensibles cifrados
A03	Inyecciones SQL/Comando/LDAP a través de formularios o endpoints	SQLmap, Postman	Pruebas de inputs con payloads maliciosos y scripts de inyección
A04	Falta de seguridad desde la arquitectura y flujo de la aplicación	Análisis manual	Revisión del diseño, flujos inseguros y ataques lógicos potenciales
A05	Errores en configuraciones de servidor, puertos, headers, .env	Nmap, Nikto	Escaneo de puertos abiertos, análisis de header, configuración del backend

A06	Uso de dependencias con CVEs o software obsoleto	Nmap	Npm Audit, verificación de librerías obsoletas y pruebas de escaneo
A07	Fallos en login, gestión de sesiones, MFA	Análisis manual, Visual Code	Verificación de implementación en el backend
A08	Fallos en manipulación de datos	Análisis manual, hash, Visual Code	Validación de integración de payloads, firma de tokens y control en despliegues.
A09	Ausencia de registros y alertas frente a eventos críticos	Visual Code, Logger	Verificación de logs en sesiones, errores y trazabilidad de eventos
A10	Uso del gateway como proxy hacia servicios internos o externos no controlados.	Análisis de código estático	Código de la construcción dinámica de target y la reescritura de rutas.

3.2. Aplicación de herramientas Kali por categoría OWASP

La ejecución de las siguientes herramientas por cada categoría, tienen como objetivo validar los mecanismos de protección implementados y documentar el comportamiento del sistema.

3.2.1. A01 – Broken Access Control

La categoría hace referencia a vulnerabilidades derivadas de un control de acceso mal implementado o inexistente. La evaluación se centró en comprobar si los roles definidos estaban correctamente restringidos.

I. Desarrollo de la evaluación

Se autenticó un usuario con el rol 'LECTOR_SEGURIDAD en la plataforma UTN Móvil. Mediante Burp Suite, se interceptó una solicitud hacia el backend y se extrajo el token JWT correspondiente. Este no contenía el permiso para VER_APLICACIONES, como se observa en Figura 58.

```

{
  "role": "LEO",
  "permissions": [
    "VER",
    "VER",
    "VER",
    "VER",
    "VER",
    "VER"
  ]
}

```

Figura 58. Fragmento de Token generado para usuario sin permiso.

II. Prueba de manipulación y autorización

En Burp Suite se agregó manualmente el token en el header *Authorization* como se observa en Figura 59.

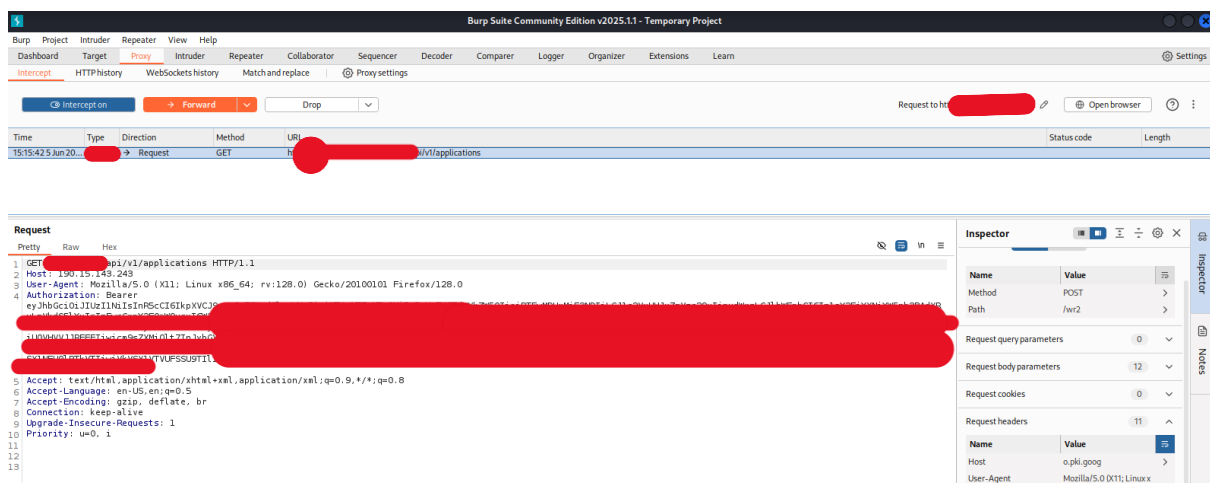


Figura 59. Pantalla de Burp Suite donde receptamos la solicitud y agregamos el token (Prueba 01 de A01)

Se despachó la solicitud y, en la sección del historial de Burp Suite, se vio que esta solicitud no fue autorizada debido a la falta de permisos del token para acceder al recurso, lo que resulto en un error 401 como se observa en Figura 60.

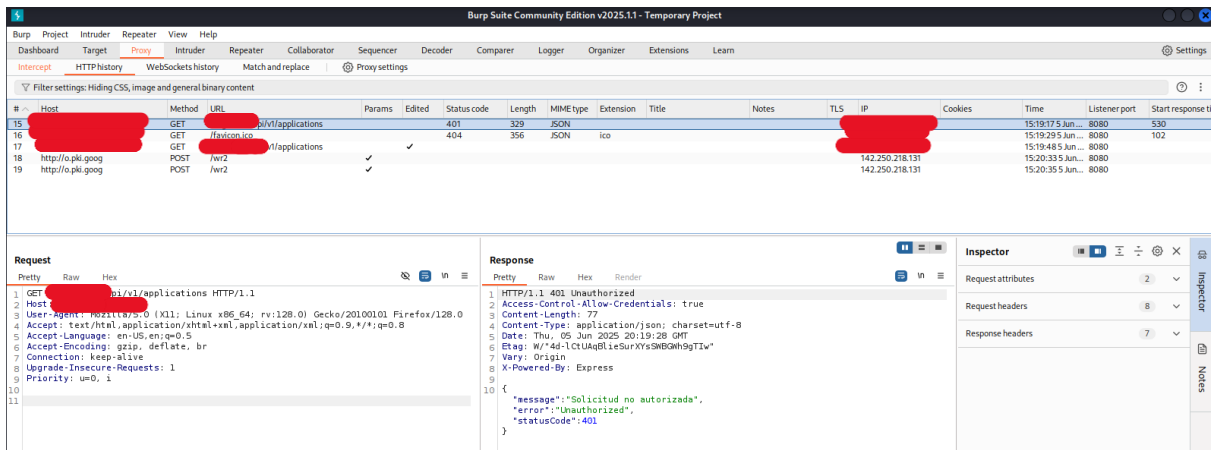


Figura 60. Pantalla de historial de solicitudes de Burp Suite (Prueba 01 de A01)

III. Resultado

El token JWT del usuario no tenía el permiso adecuado para acceder a los recursos restringidos. La evaluación demuestra que el control basado en permisos fue implementado correctamente.

El siguiente caso fue desarrollado con un usuario sin ningún rol administrativo.

I. Desarrollo de la evaluación.

Para esta evaluación se autenticó un usuario sin un rol administrativo en la plataforma UTN Móvil. Mediante Burp Suite, se interceptó una solicitud legítima hacia el backend y se extrajo el token JWT correspondiente. Este no contenía ningún rol administrativo, como se observa en Figura 61.

```

"ap[REDACTED]": [
  {
    "app[REDACTED]": "[REDACTED]",
    "rol[REDACTED]": [
      {
        "rol[REDACTED]": "DOC[REDACTED]",
        "per[REDACTED]": [
          "PERMISO[REDACTED]"
        ]
      }
    ]
  }
],
{
  "ap[REDACTED]": [REDACTED],
  "rol[REDACTED]": [
    {
      "rol[REDACTED]": "DOC[REDACTED]",
      "per[REDACTED]": [
        "PERMISO[REDACTED]"
      ]
    }
  ]
},
],

```

Figura 61. Fragmento de token generado para usuario sin rol Administrativo

II. Prueba de manipulación y autorización.

En Burp Suite se agregó manualmente el token en el header *Authorization* como se observa en Figura 62.

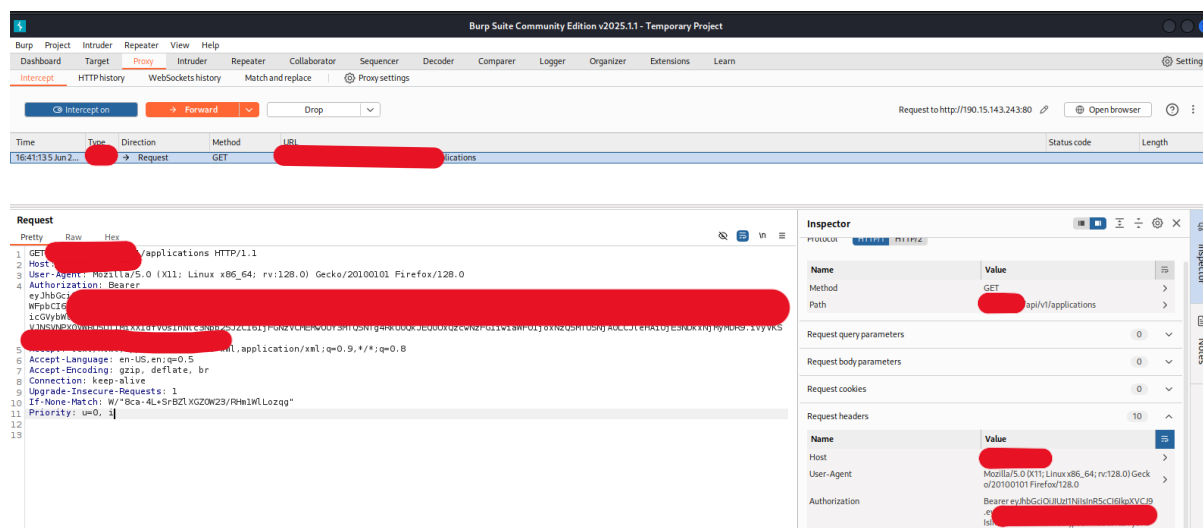


Figura 62. Pantalla de Burp Suite donde receptamos la solicitud y agregamos el token (Prueba 02 de A01)

Se despachó la solicitud y, en la sección del historial de Burp Suite, se vio que esta solicitud no fue autorizada debido a la falta de roles del token para acceder al recurso, lo que resultó en un error 403 como se observa *Figura 63*.

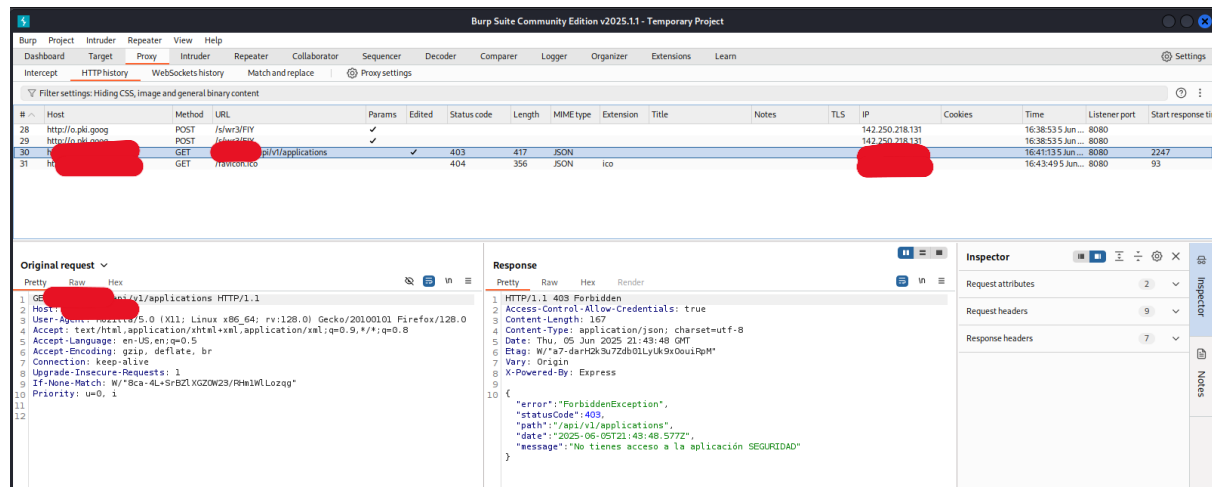


Figura 63. Pantalla de historial de solicitudes de Burp Suite (Prueba 02 de A01)

III. Resultado

Se bloqueó correctamente la solicitud ya que el token del usuario no tenía ningún rol administrativo. La evaluación demuestra que el control basado en roles se implementó correctamente.

Como recomendación de esta evaluación, se debe asegurar que todos los roles en el sistema estén correctamente definidos y que los permisos estén asignados de manera correcta para evitar accesos no autorizados.

3.2.2. A02 – Cryptographic Failures

La categoría abarca la implementación de cifrado débil o incorrecto que puede comprometer la confidencialidad de los datos sensibles. Se incluye problemas como el uso de algoritmos de cifrado obsoletos, gestión deficiente de claves o la transmisión sin cifrado de información sensible.

La evaluación se centró en los módulos desarrollados para autenticación y autorización. Específicamente en la generación de tokens, almacenamiento de contraseñas y protección de datos sensibles.

I. Desarrollo de la evaluación.

Se analizaron dos aspectos fundamentales:

- Manejo de claves y tokens

Se verificó que el sistema utiliza tokens JWT firmados con algoritmos simétricos. Esta firma se realiza dinámicamente utilizando una clave secreta robusta que es cargada desde el servicio de parámetros, lo que evita que este hardcodeada en el código. En Figura 64 se observa la estrategia JWT donde se utilizan un proveedor asíncrono para obtener la clave secreta desde el servicio Parámetros.

```
@Injectable()
export class JwtStrategy extends PassportStrategy(Strategy, 'jwt') {

  /**
   * Configura la estrategia JWT de Passport para usar una clave dinámica
   * obtenida desde el servicio de parámetros del sistema.
   *
   * @param parametrosService Servicio para consultar la clave JWT desde configuración dinámica.
   */
  constructor(private readonly parametrosService: ParametrosService) {}

  super({
    jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
    ignoreExpiration: false,
    secretOrKeyProvider: async (_, done) => {
      try {
        const secret = await parametrosService.getParameterValue('JWT_SECRET');
        done(null, secret);
      } catch (err) {
        done(err, null);
      }
    },
  });
}
```

Figura 64. Configuración dinámica de la estrategia JWT con clave obtenida desde parámetros del sistema

La firma y el tiempo de expiración del token del token se establecen en el método newAccessToken, que firma la carga útil con algoritmos asimétricos, y define una duración obtenida también de los parámetros de configuración. Esto asegura que los tokens sean válidos por un tiempo controlado y firmados de forma segura, como se evidencia en Figura 65.

```
async newAccessToken(payload: any): Promise<string> {
  return jwt.sign(payload, this.accessSecret, {
    algorithm: 'HS256',
    expiresIn: '1h',
  });
}
```

Figura 65. Generación del token de acceso firmado con algoritmo simétrico y expiración configurada.

El sistema implementa un mecanismo robusto para los tokens de refresco mediante algoritmos asimétricos. En Figura 66 se muestra el método para obtener un nuevo token de refresco.

```

async newRefreshToken(payload: { [REDACTED] }): Promise<string> {
  return jwt.sign(payload, this.refreshPrivateKey, {
    algorithm: [REDACTED],
    expiresIn: [REDACTED]
  });
}

```

Figura 66. Generación del refresh token firmado con algoritmo asimétrico y expiración controlada.

- Encriptación de datos sensibles.

Durante el proceso de login, se emplea una decodificación a la contraseña, como se muestra en Figura 67, ya que desde el fronted se envía codificada. Esta técnica no constituye un cifrado como tal.

```

const decoded = Buffer.from(loginRequest.password, [REDACTED]);
const recoded = decoded.toString([REDACTED]).replace(/=+$/, '');
const original = loginRequest.password.replace(/=+$/, '');

```

Figura 67. Decodificación de contraseña durante el inicio de sesión.

La autenticación de usuarios se delega a un sistema robusto y certificado como Microsoft Identity Platform.

En cuanto al manejo de otros datos sensibles, se constató que el sistema cifra automáticamente los valores de parámetros que contiene en su identificador términos. Este cifrado se realiza antes de almacenar dichos parámetros en la base de datos. En Figura 68 se muestra de qué manera se aplica esta lógica.

```

// Cifrar automáticamente si el parámetro es de tipo contraseña o secreto
if (dto.valor && (dto.[REDACTED].includes([REDACTED]) || dto.[REDACTED].includes([REDACTED])) {
  dto.valor = this.encryptionService.encrypt(dto.valor);
}

```

Figura 68. Cifrado automático de valores sensibles antes de guardar parámetros.

Este cifrado se lleva a cabo mediante el algoritmo simétrico, utilizando una clave derivada de una variable de entorno. En Figura 69 se muestra la implementación del servicio EncryptionService, responsable del cifrado y descifrado seguro de estos valores.

```

private readonly algorithm = 'XXXXXXXXXX';
private readonly key = crypto.scryptSync('XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX', 'XXXXXXXXXX', 32);
private readonly iv = Buffer.alloc(16, 0);

/**
 * Cifra un texto plano utilizando el algoritmo XXXXXXXXXX.
 * @param text Texto a cifrar.
 * @returns Texto cifrado en formato hexadecimal.
 */
encrypt(text: string): string {
    const cipher = crypto.createCipheriv(this.algorithm, this.key, this.iv);
    let encrypted = cipher.update(text, 'utf8', 'hex');
    encrypted += cipher.final('hex');
    return encrypted;
}

```

Figura 69. Servicio de cifrado y descifrado.

II. Resultado.

La evaluación realizada a los mecanismos de criptográficos implementados demostró un cumplimiento adecuado con las buenas prácticas recomendadas por OWASP. Se confirmó el uso de algoritmos seguros con claves dinámicas no expuestas y cifrado automático de parámetros sensibles mediante algoritmo síncrono. En conjunto, la aplicación de los mecanismos permiten mitigar eficazmente vulnerabilidades criptográficas, fortaleciendo la confidencialidad e integridad de la plataforma.

3.2.3. A03 – Injection

La categoría abarca vulnerabilidades relacionadas a la inyección de comandos maliciosos. Las inyecciones pueden comprometer la integridad de los datos. En algunos casos puede permitir el acceso no autorizado a información sensible.

En la arquitectura de la plataforma UTN Móvil, basado en microservicios y expuesta a través de un API Gateway, las pruebas de inyección fueron dirigidas siempre a las rutas públicas del Gateway, ya que representa el punto real de exposición hacia los usuarios.

I. Desarrollo de la evaluación

La evaluación se enfocó específicamente en los endpoints que reciben datos del cliente.

- Endpoint evaluado /api/v1/parametros/:id

Se utilizó SQLMap para comprobar si el parámetro dinámico :id era vulnerable a inyección SQL. La prueba fue ejecutada contra el endpoint del API Gateway mediante el siguiente comando:

```
sqlmap -u "https://url/redirigir/api/v1/parametros/..." --batch --level=5 --random-agent
```

Los resultados se muestran en Figura 70.



```
(root@sebas)-[~]
# sqlmap -u "https://url/redirigir/api/v1/parametros/..." --batch --level=5 --random-agent

{1.9.2#stable}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It
is the end user's responsibility to obey all applicable local, state and federal laws. Developers assu
me no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 20:18:43 /2025-06-13/

[20:18:43] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (X11; U; SunOS sun4u; en-GB;
rv:1.8.0.1) Gecko/20060206 Firefox/1.5.0.1' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
[20:18:43] [WARNING] you've provided target URL without any GET parameters (e.g. 'http://www.site.com/a
rticle.php?id=1') and without providing any POST parameters through option '--data'
do you want to try URI injections in the target URL itself? [Y/n/q] Y
[20:18:43] [INFO] testing connection to the target URL
[20:18:45] [CRITICAL] not authorized, try to provide right HTTP authentication type and valid credentia
ls (401). If this is intended, try to rerun by providing a valid value for option '--ignore-code'
[20:18:45] [WARNING] 401 (Unauthorized) - 1 times

[*] ending @ 20:18:45 /2025-06-13/
```

Figura 70. Comando de SQLMap ejecutado sobre el endpoint GET /parametros/:id para evaluar inyección en URL.

La respuesta fue un 401, demostrando que el recurso tiene un mecanismo de autenticación. Con el fin de tener una evaluación más profunda, se evaluó GET api/v1/parametros/.. un endpoint con contexto público como se observa en *Figura 71*.

```
(root@sebas)-[~]
# sqlmap -u "https://[REDACTED]/api/v1/parametros/[REDACTED]" --batch --level=5 --random-agent

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 20:24:34 /2025-06-13/

[20:24:34] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.67 Safari/537.36' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
[20:24:34] [WARNING] you've provided target URL without any GET parameters (e.g. 'http://www.site.com/article.php?id=1') and without providing any POST parameters through option '--data'
do you want to try URI injections in the target URL itself? [Y/n/q] Y
[20:24:34] [INFO] testing connection to the target URL
[20:24:36] [INFO] checking if the target is protected by some kind of WAF/IPS
[20:25:06] [CRITICAL] heuristics detected that the target is protected by some kind of WAF/IPS
are you sure that you want to continue with further target testing? [Y/n] Y
[20:25:06] [WARNING] please consider usage of tamper scripts (option '--tamper')
[20:25:06] [INFO] testing if the target URL content is stable
[20:25:06] [INFO] target URL content is stable
[20:25:06] [INFO] testing if URI parameter '#1*' is dynamic
[20:25:06] [WARNING] URI parameter '#1*' does not appear to be dynamic
[20:25:06] [CRITICAL] not authorized, try to provide right HTTP authentication type and valid credentials (401). If this is intended, try to rerun by providing a valid value for option '--ignore-code'
[20:25:06] [WARNING] error codes detected during run:
401 (Unauthorized) - 2 times

[*] ending @ 20:25:06 /2025-06-13/
```

Figura 71. Comando de SQLMap ejecutado sobre el endpoint GET /parametros/app/version para evaluar inyección en URL.

En base a los resultados, a pesar de ser una ruta expuesta, la herramienta detectó restricciones activas, incluyendo mecanismos de protección tipo Web Application Firewall y una respuesta 401 ante intentos de acceso automatizados. Esto indica que, incluso en endpoints públicos, existen medidas que filtran el tráfico no autorizado o sospechoso, reforzando la postura de seguridad ante ataques automatizados de inyección.

II. Evaluación manual

Se realizó una prueba manual de autenticada al endpoint PUT /api/v1/parámetros/.... Utilizando la herramienta Postman se envió un JSON con campos manipulados para simular inyección de código malicioso.

El campo descripción se utilizó para insertar el string 'OR 1=1 --', simulando una inyección SQL básica. El campo valor incluyó un script HTML con el fin de evaluar si el sistema almacenaba contenido sin sanitización. Utilizando un token válido en el encabezado Authorization, se logró realizar la prueba en un contexto de sesión autenticada.

En Figura 72, se observa que al enviar la solicitud el servidor responde con un error de tipo *socket hang up*, lo que indica que la conexión fue terminada abruptamente.

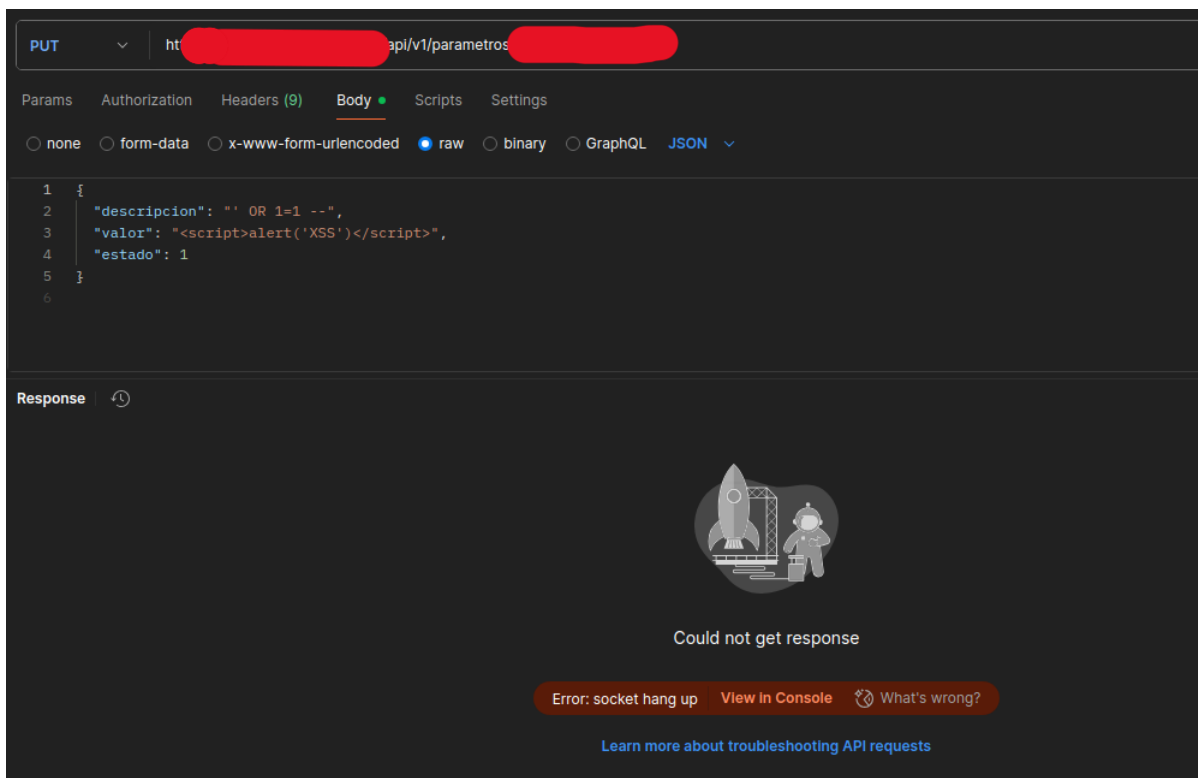


Figura 72. Resultado de la prueba manual con payloads maliciosos enviada mediante Postman al endpoint protegido PUT /parametros/... .

III. Resultado

Las pruebas realizadas en esta categoría demostraron que se ha implementado mecanismos que mitigan de forma eficiente los riesgos asociados a intentos de inyección de código malicioso, tanto en endpoints protegidos como en rutas expuesta públicamente. De igual forma, la evaluación manual mediante Postman en un contexto autenticado provocó un cierre abrupto de la conexión, lo cual evidencia una respuesta defensiva del backend frente a entradas potencialmente peligrosas.

3.2.4. A04 – Injection Desing

La categoría se enfoca en deficiencias de seguridad originadas desde el diseño o arquitectura del sistema. Incluye flujos inseguros, carencia de controles lógicos, ausencia de validaciones de autorización o confianza excesiva en el cliente.

Se evaluaron distintos escenarios del diseño de flujo y distribución entre microservicios, frontend y API Gateway.

I. Análisis de arquitectura.

Se identificó que la solución tecnológica de la plataforma UTN Móvil se basa en una arquitectura de microservicios orquestados mediante contenedores Docker y organizados en capas lógicas, de aplicación, dominio, infraestructura y presentación. El acceso a los servicios se realiza exclusivamente a través de un API Gateway centralizado, el cual expone su interfaz pública mediante el gestor de tráfico Traefik, como se puede observar en *Figura 73*.

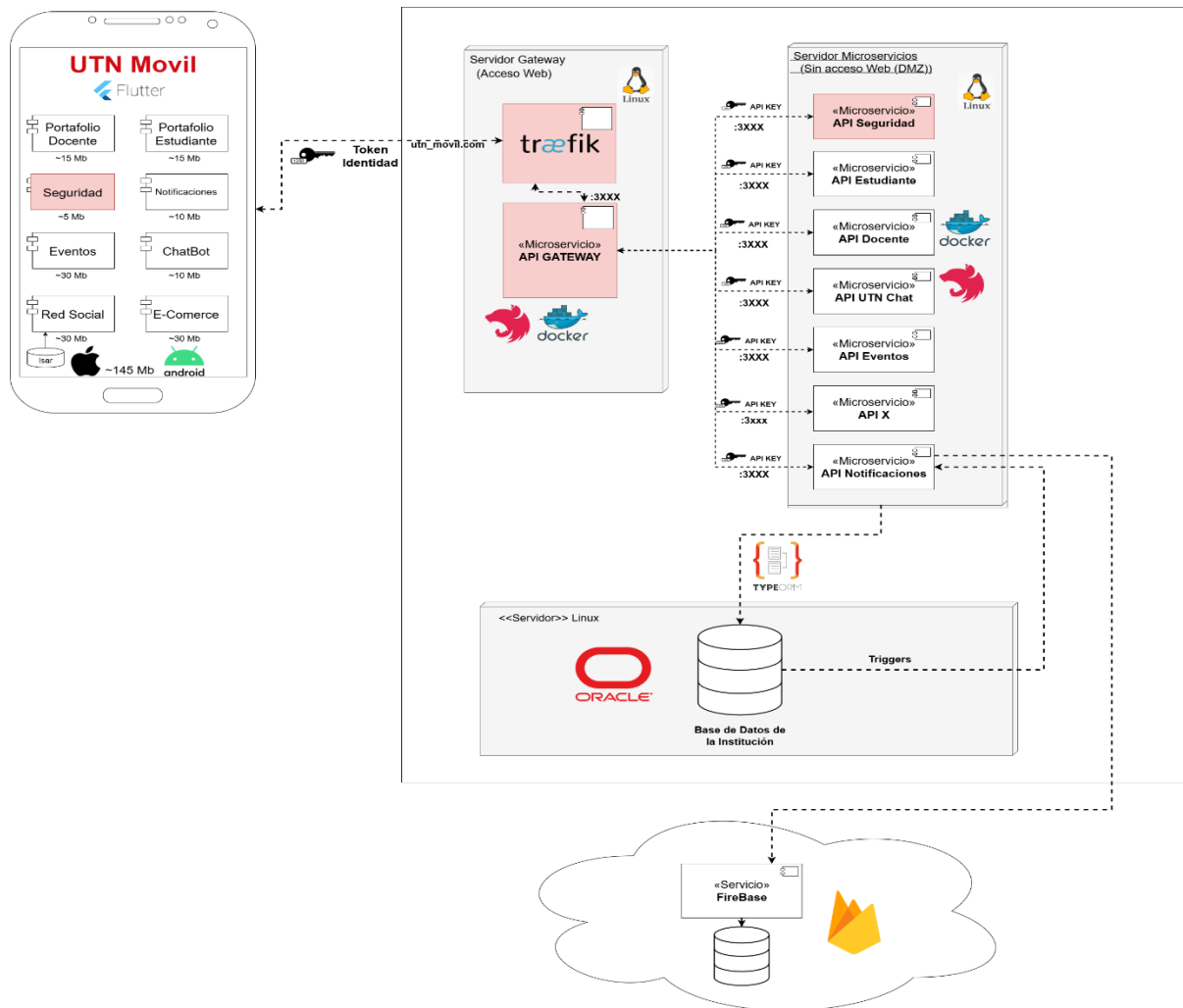


Figura 73. Arquitectura general de la plataforma UTN Móvil, compuesta por frontend, microservicios desacoplados, API Gateway y base de datos centralizada.

A través del análisis de los componentes y flujos del sistema, se identificaron los siguientes puntos destacados:

- Exposición controlada: todos los endpoints de sistema están expuestos exclusivamente desde el API Gateway. La comunicación entre el microservicio y el Gateway está protegida por un API Key único por cada microservicio.
- Validación de roles y permisos a nivel de backend: Los microservicios aplican mecanismos que aseguran que los accesos no dependan exclusivamente del cliente.

- Restricción de acceso basado en perfil y contexto: el sistema verifica si el usuario está activo y lo cruza con su rol, permiso y estado institucional.
- Verificación de versión mínima de aplicación móvil: se fuerza el cumplimiento de una versión mínima definida en la base de datos.
- Modularidad escalable y mantenible: el diseño en capas lógicas, aplicación, dominio, infraestructura y presentación, permiten aislar la lógica de negocio. Esa organización promueve prácticas de desarrollo seguro y sostenible.

II. *Identificación de posibles riesgos lógicos*

Durante el análisis se revisaron posibles escenarios lógicos relacionados con el uso prolongado de tokens o el acceso repetido a ciertos recursos del sistema. Aunque no se identificaron vulnerabilidades activas, se constató que hay unas oportunidades de mejora que refuerzan un diseño arquitectónico que ya muestra una orientación clara hacia principios de seguridad robustos y escalables.

III. *Resultado*

La evaluación evidenció que la plataforma implementa un enfoque sólido basado en microservicios. Los permisos son validados tanto en el frontend como en el backend. La implementación de estas características permitió reducir el acoplamiento y proteger los servicios internos frente a accesos no autorizados.

3.2.5. A05 – Security Misconfiguration

La categoría abarca configuraciones incorrectas o inseguras en los servidores y entornos de ejecución. También abarca los ajustes en las cabeceras HTTPS, puertos abiertos y archivos .env.

I. *Análisis de configuración del entorno y servidores.*

- Escaneo de puertos y servicios

Para realizar la evaluación se ejecutó un comando en la herramienta Nmap. El análisis permitió determinar que se expone puertos como el puerto 80/tcp con servicio SSL/HTTP, lo cual indica que el servidor está utilizando una configuración HTTPS. Además, se observó que el puerto 113/tcp está cerrado, lo que representa una buena práctica de seguridad. Los resultados completos de este escaneo se pueden observar en Figura 74.

```
(root@sebas)-[~]
# nmap -sV -T4 -Pn [REDACTED]

Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-19 15:06 -05
Nmap scan report for 190.15.143.243
Host is up (0.019s latency).
Not shown: 996 filtered tcp ports (no-response), 1 filtered tcp ports (admin-prohibited)
PORT      STATE SERVICE      VERSION
80/tcp    open  ssl/http     Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
113/tcp   closed ident

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 169.41 seconds
```

Figura 74. Resultado del escaneo de puertos realizado con Nmap para la IP pública del servidor del Gateway.

En Figura 75 se muestra el resultado del escaneo de Nmap, donde el estado del puerto 443 es reportado como filtrado. Podemos confirmar que el servidor está protegido y que el acceso al servicio HTTPS está restringido.

```
(root@sebas)-[~]
# nmap -p 443 -Pn [REDACTED]

Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-19 15:43 -05
Nmap scan report for [REDACTED]
Host is up (0.067s latency).

PORT      STATE SERVICE
443/tcp   filtered https

Nmap done: 1 IP address (1 host up) scanned in 0.58 seconds
```

Figura 75. Resultado del escaneo de puertos, donde el puerto 443 aparece como filtrado.

- Evaluación de cabeceras HTTP

Se ejecutó un escaneo con la herramienta Nikto para identificar posibles vulnerabilidades. Los detalles completos de este escaneo pueden observarse en Figura 76.

```
(root@sebas)-[~]
# nikto -h h [REDACTED]

- Nikto v2.5.0

+ Target IP: [REDACTED]
+ Target Hostname: [REDACTED]
+ Target Port: [REDACTED]
+ Start Time: 2025-06-19 16:54:27 (GMT-5)

+ Server: No banner retrieved
+ /: Retrieved x-powered-by header: Express.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
```

Figura 76. Resultado del escaneo con Nikto

Luego de este diagnóstico, se implementó las cabeceras de seguridad necesarias en el servidor, añadiendo el middleware correspondiente para configurar las cabeceras como X-Frame-Options, X-Content-Type-Options y Strict-Transport-Security. Esto ayuda a prevenir clickjacking, content sniffing y asegurar que las conexiones solo se realicen a través de HTTPS. Los detalles de la implementación de este middleware pueden observarse en *Figura 77*.

```
import { Injectable, NestMiddleware } from '@nestjs/common';
import { Request, Response, NextFunction } from 'express';

@Injectable()
export class SecurityHeadersMiddleware implements NestMiddleware {
  use(req: Request, res: Response, next: NextFunction) {
    res.setHeader('X-Frame-Options', 'SAMEORIGIN'); // Prevenir clickjacking
    res.setHeader('X-Content-Type-Options', 'nosniff'); // Prevenir sniffing de contenido
    res.setHeader('Strict-Transport-Security', 'max-age=31536000; includeSubDomains'); // HSTS
    next();
  }
}
```

Figura 77. Implementación del middleware de cabeceras de seguridad en NestJS

- Manejo de variables de entorno (.env)

Se confirmó que los microservicios y el API Gateway cargan sus claves sensibles y configuraciones críticas a través de variables de entorno, archivos .env, asegurando que no se expongan ninguna información sensible en el código fuente. Este enfoque es esencial para proteger las credenciales, las claves de API y otros secretos necesarios para el funcionamiento de los servicios. En *Figura 78* se puede observar un ejemplo aplicado en el Gateway.

```
services.estudiante.baseUrl = this.configService.get<string>('ESTUDIANTE_BASE_URL');
services.estudiante.apiKey = this.configService.get<string>('ESTUDIANTE_API_KEY');
```

Figura 78. Configuración de las variables de entorno en el API Gateway, donde se cargan las claves y URLs para los servicios

El acceso a estas variables de entorno está estrictamente restringido en el entorno productivo, lo que minimiza el riesgo de filtraciones accidentales o exposición de secretos a través de repositorios o configuraciones incorrectas. En los microservicios, la conexión a la base de datos también se realiza mediante variables de entorno para garantizar que las credenciales de la base de datos no se expongan en el código. Esto se hace mediante la configuración a Oracle en el microservicio, como se muestra en *Figura 79*.

```

imports: [ConfigModule],
useFactory: async (configService: ConfigService) => ({
  type: "oracle",
  host: configService.get<string>(''),
  port: configService.get<number>(''),
  username: configService.get<string>(''),
  password: configService.get<string>(''),
  serviceName: configService.get<string>(''),
  thickMode: true,
  entities: [

```

Figura 79. Configuración de la conexión a la base de datos en un microservicio, donde las credenciales se obtienen de las variables de entorno para proteger la información sensible.

II. Resultado.

El resultado realizado sobre la infraestructura y configuración del sistema reveló que se identificó cabeceras esenciales de seguridad esenciales que no estaban configuradas correctamente. Además, se verificó que el sistema utiliza adecuadamente las variables de entorno para manejar claves sensibles y configuraciones críticas.

3.2.6. A06 - Vulnerable and outdated componets

La categoría aborda los riesgos derivados del uso de bibliotecas o componentes que presentan vulnerabilidades conocidas. La exposición a puede permitir a un atacante aprovechar debilidades en versiones antiguas para ejecutar códigos maliciosos, escalar privilegios o comprometer la seguridad del sistema.

I. Desarrollo de la evaluación

A continuación, vamos se realizaron las siguiente herramientas y enfoques a cada dependencia y componente tanto del frontend como del backend.

- Análisis de dependencias en NodeJS

La ejecución del comando `npm audit` identificó vulnerabilidades, como se puede observar en *Figura 80*. Las vulnerabilidades se distribuyen de la siguiente manera: 2 de severidad baja, 2 moderadas y 5 altas.

```

PS C:\Users\sebas\Desktop\UTN_MOVIL\utn-movil-seguridad> npm audit
# npm audit report

@babel/helpers <7.26.10
Severity: moderate
Babel has inefficient RegExp complexity in generated code with .replace when transpiling named capturing groups - https://github.com/advisories/GHSA-968p-4wvh-cqc8
fix available via `npm audit fix`
node_modules/@babel/helpers

@nestjs/common <10.4.16
Severity: moderate
nest allows a remote attacker to execute arbitrary code via the Content-Type header - https://github.com/advisories/GHSA-cj7v-w2c7-cp7c
fix available via `npm audit fix`
node_modules/@nestjs/common

axios 1.0.0 - 1.8.1
Severity: high
axios Requests Vulnerable To Possible SSRF and Credential Leakage via Absolute URL - https://github.com/advisories/GHSA-jr5f-v2jv-69x6
fix available via `npm audit fix`
node_modules/axios

brace-expansion 1.0.0 - 1.1.11 || 2.0.0 - 2.0.1
brace-expansion Regular Expression Denial of Service vulnerability - https://github.com/advisories/GHSA-v6h2-p8h4-qcjq
brace-expansion Regular Expression Denial of Service vulnerability - https://github.com/advisories/GHSA-v6h2-p8h4-qcjq
fix available via `npm audit fix`
node_modules/eslint/eslintrc/node_modules/brace-expansion
node_modules/ghumanwhocodes/config-array/node_modules/brace-expansion
node_modules/jest/reporters/node_modules/brace-expansion
node_modules/brace-expansion
node_modules/eslint/node_modules/brace-expansion
node_modules/fork-ts-checker-webpack-plugin/node_modules/brace-expansion
node_modules/jake/node_modules/brace-expansion
node_modules/jest-config/node_modules/brace-expansion
node_modules/jest-runtime/node_modules/brace-expansion

```

Figura 80. Informe inicial del análisis de dependencias con npm audit.

Este análisis reveló que las versiones de algunas librerías necesitaban ser actualizadas para mitigar riesgos de seguridad. Con el fin de resolver estos riesgos se ejecutó el comando npm audit fix para actualizar automáticamente las dependencias afectadas y se eliminaron las versiones vulnerables. En *Figura 81* se puede observar el resultado de la ejecución del comando. Sin embargo, se presentaron advertencias sobre dependencias de pares conflictivas. A pesar de esto, las vulnerabilidades fueron resueltas exitosamente, y se verificó que no quedaran vulnerabilidades en el sistema.

```

● PS C:\Users\sebas\Desktop\UTN_MOVIL\utn-movil-seguridad> npm audit fix
npm warn ERESOLVE overriding peer dependency
npm warn While resolving: @nestjs/terminus@10.2.0
npm warn Found: reflect-metadata@0.2.2
npm warn node_modules/reflect-metadata
npm warn reflect-metadata@"^0.2.0" from the root project
npm warn 7 more (@nestjs/core, @nestjs/mapped-types, ...)
npm warn
npm warn Could not resolve dependency:
npm warn peer reflect-metadata@"0.1.x" from @nestjs/terminus@10.2.0
npm warn node_modules/@nestjs/terminus
npm warn optional @nestjs/terminus@"10.2.0" from @nestjs-modules/ioredis@2.0.2
npm warn node_modules/@nestjs-modules/ioredis
npm warn
npm warn Conflicting peer dependency: reflect-metadata@0.1.14
npm warn node_modules/reflect-metadata
npm warn peer reflect-metadata@"0.1.x" from @nestjs/terminus@10.2.0
npm warn node_modules/@nestjs/terminus
npm warn optional @nestjs/terminus@"10.2.0" from @nestjs-modules/ioredis@2.0.2
npm warn node_modules/@nestjs-modules/ioredis

added 11 packages, removed 6 packages, changed 32 packages, and audited 792 packages in 15s

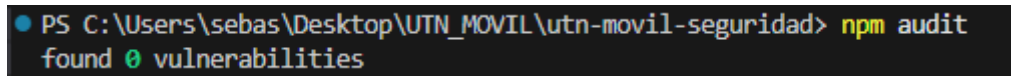
130 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

```

Figura 81. Resolución de vulnerabilidades con npm audit fix.

Se realizó un análisis adicional para verificar que las vulnerabilidades fueron efectivamente corregidas. Como se muestra en Figura 82, el informe final indica que no se encontraron vulnerabilidades en el proyecto.



```
PS C:\Users\sebas\Desktop\UTN_MOVIL\utn-movil-seguridad> npm audit
found 0 vulnerabilities
```

Figura 82. Resultado del análisis posterior con *npm audit*.

- Verificación de servicios externos

Se evaluaron las versiones de servicio externos utilizados en la plataforma, en particular Redis y Oracle, que cumplen funciones críticas en la persistencia y cacheo de datos. Este análisis permitió determinar si los servicios en ejecución continúan recibiendo soporte de seguridad y si se encuentran dentro de un ciclo de vida mantenido por sus desarrolladores.

En Figura 83, se muestra que la versión de Redis continuará recibiendo actualizaciones de seguridad únicamente hasta el 28 de febrero de 2026, lo que implica que su uso prolongado debe ser considerado con precaución, especialmente en sistema productivos que manejen información sensible.

Release	Released	Active Support	Security Support	Latest
8.0	1 month and 3 weeks ago (02 May 2025)	Yes	Yes	8.0.2 (27 May 2025)
7.4	11 months ago (29 Jul 2024)	Yes	Ends in 1 year and 5 months (30 Nov 2026)	7.4.4 (27 May 2025)
7.2	1 year and 10 months ago (15 Aug 2023)	Ended 11 months ago (29 Jul 2024)	Ends in 8 months (28 Feb 2026)	7.2.9 (27 May 2025)
7.0	3 years ago (27 Apr 2022)	Ended 1 year and 10 months ago (15 Aug 2023)	Ended 11 months ago (29 Jul 2024)	7.0.15 (09 Jan 2024)

[Show more unmaintained releases](#)

Figura 83. Ciclo de soporte de Redis versión 7.2, indicando el fin de soporte activo y la vigencia del soporte de seguridad hasta 2026 [96].

En cuanto a la base de datos Oracle, el sistema utiliza la versión Oracle 19c, la cual sigue recibiendo soporte dentro del plan Premier Support hasta el 31 de diciembre de 2029 y Extended Support hasta el 31 de diciembre de 2032, como se observa en Figura 84. Esto convierte a Oracle 19c en una versión adecuada para entornos productivos a largo plazo, dado sus estatus como versión LTS.

Release	Released	Premier Support	Extended Support
23ai (LTR)	1 year and 9 months ago (19 Sep 2023)	Ends in 6 years (31 Dec 2031)	Yes
21c	3 years and 10 months ago (13 Aug 2021)	Ends in 2 years (31 Jul 2027)	Unavailable
19c (LTR)	6 years ago (25 Apr 2019)	Ends in 4 years and 6 months (31 Dec 2029)	Ends in 7 years (31 Dec 2032)
18c	6 years and 11 months ago (23 Jul 2018)	Ended 3 years and 12 months ago (30 Jun 2021)	Unavailable

[Show more unmaintained releases](#)

Figura 84. Tabla de soporte oficial de Oracle Database, donde se evidencia que la versión 19c continúa dentro del ciclo de mantenimiento [97].

- Revisión de vulnerabilidades en el fronted (Flutter)

En la revisión de las dependencias del frontend de la aplicación, se evaluaron las versiones de las librerías y dependencias. No se identificaron vulnerabilidades críticas en las dependencias principales como `firenze_core`, `go_router` o `flutter_secure_storage`.

En Figura 85 se muestra la salida del comando ejecutado que confirma las versiones más recientes de las librerías.

```
PS C:\Users\sebas\Desktop\UTN_MOVIL\utn_movil_app> flutter pub outdated
Showing outdated packages.
[*] indicates versions that are not the latest available.
```

Package Name	Current	Upgradable	Resolvable	Latest
direct dependencies:				
file_picker	*8.3.7	*8.3.7	*9.2.3	10.2.0
fl_chart	*0.70.2	*0.70.2	1.0.0	1.0.0
flutter_local_notifications	*17.2.4	*17.2.4	19.3.0	19.3.0
geolocator	*13.0.4	*13.0.4	14.0.1	14.0.1
go_router	*14.8.1	*14.8.1	15.2.4	15.2.4
html_editor_enhanced	*2.6.0	*2.6.0	2.7.0	2.7.0
intl	*0.19.0	*0.19.0	*0.19.0	0.20.2
mobile_scanner	*6.0.10	*6.0.10	7.0.1	7.0.1
table_calendar	*3.1.3	*3.1.3	*3.1.3	3.2.0
timezone	*0.9.4	*0.9.4	0.10.1	0.10.1
dev dependencies:				
flutter_lints	*4.0.0	*4.0.0	*5.0.0	6.0.0
transitive dependencies:				
async	*2.12.0	*2.12.0	*2.12.0	2.13.0
audio_session	*0.1.25	*0.1.25	*0.1.25	0.2.2
fake_async	*1.3.2	*1.3.2	*1.3.2	1.3.3
flutter_local_notifications_linux	*4.0.1	*4.0.1	6.0.0	6.0.0
flutter_local_notifications_platform_interface	*7.2.0	*7.2.0	9.1.0	9.1.0
flutter_local_notifications_windows	-	-	1.0.0	1.0.0
flutter_secure_storage_linux	*1.2.3	*1.2.3	*1.2.3	2.0.1
flutter_secure_storage_macos	*3.1.3	*3.1.3	*3.1.3	4.0.0
flutter_secure_storage_platform_interface	*1.1.2	*1.1.2	*1.1.2	2.0.1
flutter_secure_storage_web	*1.2.1	*1.2.1	*1.2.1	2.0.0
flutter_secure_storage_windows	*3.1.2	*3.1.2	*3.1.2	4.0.0
geolocator_android	*4.6.2	*4.6.2	5.0.1+1	5.0.1+1
js	*0.6.7	*0.6.7	*0.6.7	0.7.2
just_audio	*0.9.46	*0.9.46	*0.9.46	0.10.4
leak_tracker	*10.0.8	*10.0.8	*10.0.8	11.0.1
leak_tracker_flutter_testing	*3.0.9	*3.0.9	*3.0.9	3.0.10
leak_tracker_testing	*3.0.1	*3.0.1	*3.0.1	3.0.2
lints	*4.0.0	*4.0.0	*5.1.1	6.0.0
material_color_utilities	*0.11.1	*0.11.1	*0.11.1	0.13.0
meta	*1.16.0	*1.16.0	*1.16.0	1.17.0
permission_handler	*11.4.0	*11.4.0	*11.4.0	12.0.0+1
permission_handler_android	*12.1.0	*12.1.0	*12.1.0	13.0.1
petitparser	*6.1.0	*6.1.0	*6.1.0	7.0.0
test_api	*0.7.4	*0.7.4	*0.7.4	0.7.6
vector_math	*2.1.4	*2.1.4	*2.1.4	2.2.0
vm_service	*14.3.1	*14.3.1	*14.3.1	15.0.2
win32	*5.13.0	*5.13.0	*5.13.0	5.14.0
xml	*6.5.0	*6.5.0	*6.5.0	6.6.0

Figura 85. Salida del comando flutter pub outdated que muestra las dependencias desactualizadas y sus versiones más recientes disponibles.

La mayoría de los paquetes son populares y bien mantenidas. Alguno de los paquetes de menor uso o menos mantenidos deberían ser evaluados regularmente para evitar problemas de seguridad.

- Revisión de contenedores Docker

Se realizó una revisión de las imágenes de Docker utilizadas en el sistema. Se verificó si las versiones de las imágenes base eran las más actualizadas y si se había aplicado alguna política de seguridad en las imágenes personalizadas.

Se utilizó la imagen node:22.13.0-slim para el contenedor del backend como se observa en *Figura 86*. Aunque es una versión reciente, se recomienda verificar si hay actualizaciones de seguridad disponibles.

```
# Fase de producción-----  
FROM node:22.13.0-slim AS production  
  
WORKDIR /usr/src/app  
  
COPY --from=builder /usr/src/app/dist ./dist  
COPY package*.json ./  
  
ENV NODE_ENV=production
```

Figura 86. Información de la imagen base de Node.js utilizada en el contenedor Docker.

Para el servicio de Redis, se utilizó la imagen redis:7.2.7-alpine como se observa en *Figura 87*. Aunque es ligera y segura, es importante verificar si hay actualizaciones de seguridad.

```
redis_seguridad:  
  image: redis:7.2.7-alpine
```

Figura 87. Información de la imagen base de Redis utilizada en el contenedor Docker.

II. Resultado

La evaluación de esta categoría reveló varias áreas que requieren atención. A través del análisis de dependencias de NodeJS, se identificaron vulnerabilidades en axios, multer y express, las cuales fueron corregidas, asegurando que las dependencias estén actualizadas y seguras.

La evaluación de los contenedores Docker utilizados en la plataforma, incluyendo las imágenes de NodeJs y Redis, se encuentra con una versión estable y actualizada, pero es importante seguir monitoreando y aplicando actualizaciones de seguridad de manera regular. La plataforma UTN Móvil demuestra un enfoque sólido en la gestión de sus componentes, pero es crucial mantener un control continuo sobre las dependencias y servicios externos, implementando actualizaciones periódicas y auditorías de seguridad para mitigar los riesgos asociados a componentes desactualizados y vulnerables.

3.2.7. A07 – Identification and authentication failures

La categoría abarca vulnerabilidades relacionadas con el manejo inadecuado de autenticación. La de autenticación y gestión de sesiones se ha diseñado para garantizar la seguridad mediante la implementación de tokens de acceso, autenticación de usuarios y la validación de sesiones activas.

I. *Desarrollo de la evaluación.*

- Mecanismo de Login

El sistema permite a los usuarios autenticarse utilizando su correo electrónico y una contraseña, la cual está codificada antes de ser enviada al servidor. En Figura 88, se muestra la validación de la autenticación de las credenciales del usuario mediante un servicio externo de autenticación de Microsoft.

```
const { valid, mfaRequired } = await this.microsoftAuthService.validateMicrosoftCredentials(  
  loginRequest.username,  
  loginRequest.password,  
);
```

Figura 88. Validación de credenciales mediante autenticación de Microsoft.

Si las credenciales son válidas y requieren MFA, el sistema responderá solicitando la verificación adicional como se observa en Figura 89.

```
if (mfaRequired) {  
  await this.mfaService.initiateMfa(metadata);  
  return { mfaRequired: true };  
}
```

Figura 89. Respuesta si el usuario tiene MFA activado.

- Gestión de sesiones

Las sesiones son gestionadas de manera robusta para garantizar que los usuarios solo tengan acceso a sus sesiones activas. Cada vez que un usuario se autentica, se crea una nueva sesión en Redis con la información correspondiente, como se observa en Figura 90. Este proceso se maneja en el servicio SesionesAppService.

```

async createSession(session: Partial<SessionApp>): Promise<SessionApp> {
  const newSession = await this.repository.create({
    ...session,
    status: 'active',
    createdAt: new Date(),
  });

  const sessionId = (session.sessionId as Buffer).toString('hex').toUpperCase();

  await this.redisService.getClient().hSet(`sessions:${session.userId}`, sessionId, JSON.stringify({
    createdAt: session.createdAt,
    deviceModel: session.deviceModel,
    // ... other session data
  }));

  return newSession;
}

```

Figura 90. Método para crear una sesión

Esto asegura que las sesiones activas sean gestionadas de forma eficiente y que se pueda verificar la validez de la sesión antes de permitir el acceso a los recursos.

- Limitación de sesiones

El sistema también permite configurar un límite en el número de sesiones activas por usuario, lo que impide el abuso de múltiples sesiones concurrentes. Este límite es consultado desde Redis como se observa en *Figura 91*.

```

async validateSessionLimit( [redacted] : string): Promise<void> {
    const redisClient = this.redisService.getClient();
    const activeSessions = await redisClient.hGetAll(`sessions:[redacted]`);
    const currentCount = Object.keys(activeSessions).length;

    [redacted]

    if (currentCount >= max) {
        throw new LimiteSesionesExcedidoException(max);
    }
}

```

Figura 91. Método para validar el límite de sesiones por usuario.

Si un usuario ha alcanzado el número máximo de sesiones permitidas, el sistema devuelve un error.

- Autenticación multifactor (MFA)

La autenticación multifactor se implementa para ofrecer una capa adicional de seguridad al proceso de login. Si las credenciales son válidas, pero MFA es requerida, el sistema generará un código aleatorio de 6 dígitos, y lo envía al correo del usuario, como se puede observar en *Figura 92*.

```

async initiateMfa(metadata: LoginMetadataDto): Promise<void> {
    this [redacted] = metadata [redacted];
    this.fullname = metadata.full_name;

    const mfaCode = this.generateMfaCode();
    await this.saveMfaCode([redacted], mfaCode);

    await this.emailService.sendMfaCodeEmail(
        [redacted],
        this.fullname,
        mfaCode
    );
}

```

Figura 92. Método para enviar un código OTP generado.

El código MFA es validado comparando al valor ingresado por el usuario con el código almacenado en Redis, como se observa en Figura 93.

```
async validateMfaCode(otp: string, mfaCode: string): Promise<boolean> {  
  const redisClient = this.redisService.getClient();  
  const storedCode = await redisClient.get(otp);  
  if (storedCode === mfaCode) {  
    await redisClient.del(otp);  
    return true;  
  }  
  return false;  
}
```

Figura 93. Método para validar un código OTP

II. Resultados y conclusiones.

Los mecanismos de autenticación en la plataforma han demostrado que los controles implementados son efectivos. La implementación de MFA asegura que un atacante no podrá acceder sin completar el segundo factor de autenticación.

Además, se ha comprobado que el sistema permite manejar un número controlado de sesiones activas por usuario, y que se verifiquen de manera eficiente las sesiones en curso. La validación de las credenciales Microsoft y la generación de tokens JWT también siguen buenas prácticas de seguridad.

Sin embargo, se recomienda seguir monitorizando las sesiones y realizar auditorías periódicas de seguridad para garantizar que no existan vulnerabilidades. Además, se sugiere realizar pruebas regulares de MFA para verificar que los métodos de validación sigan funcionando correctamente, especialmente en escenarios de actualizaciones de la plataforma de los servicios externos como Microsoft.

3.2.8. A08 - Software and data integrity failures.

La categoría aborda las fallas relacionadas con la integridad del software y los datos. Se revisaron específicamente tres aspectos claves: la generación y validación de tokens, el control de acceso y el manejo de sesiones.

I. Desarrollo de la evaluación.

La plataforma utiliza tokens para la autenticación de acceso y para la renovación de sesiones. Se verifica que los tokens de acceso son firmados utilizando el algoritmo simétrico y una clave secreta. En Figura 94, se puede observar la implementación.

```

async newAccessToken(payload: any): Promise<string> {
  return jwt.sign(payload, this.accessSecret, {
    algorithm: [REDACTED],
    expiresIn: [REDACTED]
  });
}

```

Figura 94. Fragmento de código que muestra la generación de un token de acceso firmado.

El sistema también implementa la verificación de tokens de acceso. El token es recibido y el sistema valida su autenticidad utilizando la misma clave secreta. En *Figura 95*, se muestra el código que verifica la validez del token de acceso.

```

async verifyAccessToken(token: string): Promise<Payload> {
  try {
    return jwt.verify(token, this.accessSecret) as Payload;
  } catch (error) {
    Logger.warn(`⚠ Token inválido o expirado: ${error}`);
    throw new UnauthorizedException('Token inválido o expirado');
  }
}

```

Figura 95. Fragmento de código que muestra la verificación de un token de acceso.

El refresh token utiliza un enfoque de clave pública/privada con el algoritmo asíncrono. En *Figura 96*, se muestra cómo se genera el refresh token usando la clave privada para firmar el payload.

```

async newRefreshToken(payload: { [REDACTED] }): Promise<string> {
  return jwt.sign(payload, this.refreshPrivateKey, {
    algorithm: [REDACTED],
    expiresIn: [REDACTED]
  });
}

```

Figura 96. Fragmento de código que muestra la generación de un refresh token firmado.

El refresh token es recibido y el sistema lo valida usando la clave pública. El sistema se asegura de que el token no haya sido alterado o que no haya expirado. En *Figura 97*, se observa cómo se lleva a cabo la verificación de un refresh token.

```

async verifyRefreshToken(refreshToken: string): Promise<any> {
  try {
    return jwt.verify(refreshToken, this.refreshPublicKey, { [REDACTED] });
  } catch (error) {
    const err = error as { name: string; message: string };

    [REDACTED]

    Logger.error(`❌ Fallo al verificar refresh token: ${err.message}`, err.message);
    throw new UnauthorizedException('Fallo en la verificación del token de refresco');
  }
}

```

Figura 97. Fragmento de código que muestra la verificación de un refresh token utilizando la clave pública y el algoritmo asíncrono.

- Control de acceso y roles

El control de acceso está basado en roles y permisos definidos mediante el uso de guards y decoradores. La ruta GET 'historial', por ejemplo, requiere que el usuario tenga el permiso del módulo SEGURIDAD. La aplicación de este enfoque permite restringir el acceso a rutas sensibles solo a aquellos usuarios que tienen los permisos adecuados. En *Figura 98*, se muestra como el sistema verifica si un usuario tiene los permisos necesarios para acceder a los recursos protegidos.

```

@UseGuards(JwtAuthGuard, PermissionGuard)
@HasPermission('[REDACTED]', [REDACTED])
@Get('historial-page')
async getHistorialPaginado(
  @Query('page') page = 1,
  @Query('pageSize') pageSize = 20
): Promise<{ data: SesiónAppResponseDto[], total: number }> {
  return this.service.getHistorialPaginado(page, pageSize);
}

```

Figura 98. Código que valida los permisos de un usuario antes de permitirle acceder a una ruta.

- Manejo de sesiones activas y expiradas

El manejo de las sesiones activas se realiza utilizando Redis, lo que permite gestionar de manera eficiente las sesiones de los usuarios. Además, el sistema está diseñado para cerrar automáticamente las sesiones cuando el refresh token correspondiente ha expirado. Este enfoque asegura que no se mantengan sesiones activas con tokens inválidos. En *Figura 99*, se observa como el sistema cierra las sesiones cuando un refresh token caduca.

```
async closeRemoteSession( ): Promise<void> {  
  const redisClient = this.redisService.getClient();  
  const  
  await redisClient.hDel('');  
  const all = await this.redisService.getClient().hGetAll('');  
  await this.repository.closeSession('');  
}
```

Figura 99. Código que gestiona el cierre de sesiones.

II. Resultado.

La evaluación de la integridad del software y los datos en la plataforma mostró que se ha implementado mecanismos robustos para asegurar que los datos no sean manipulados. Los tokens están correctamente generados y validados, utilizando algoritmos seguros. El control de acceso basado en roles y permisos asegura que solo los usuarios autorizados puedan acceder a ciertas rutas. El manejo de sesiones activas y expiradas es eficiente. El sistema asegura que las sesiones caducadas sean cerradas automáticamente.

2.2.9. A09 – Security logging and monitoring failures.

La categoría abarca las fallas en los registros de seguridad y en el monitor de eventos críticos dentro de un sistema. La ausencia de registros detallados o alertas frente a eventos importantes puede mitigar que un atacante opere sin ser detectado.

I. Desarrollo de la evaluación.

- Registro de solicitudes HTTPS con LoggerInterceptor

El módulo de seguridad para la plataforma implementa un interceptor global que registra información básica de cada solicitud HTTPS entrante. El interceptor es clave para la trazabilidad de las solicitudes y la medición del rendimiento. En *Figura 100*, se muestra el código de este logger que registra la actividad de las solicitudes entrantes.

```

@Injectables()
export class LoggerInterceptor implements NestInterceptor {
  private readonly logger = new Logger(LoggerInterceptor.name);

  /**
   * Intercepta la petición y mide el tiempo que toma en completarse, registrando
   * el método HTTP, la URL, el código de estado y la duración.
   *
   * @param context Contexto de ejecución HTTP.
   * @param next Handler que ejecuta la siguiente capa del pipeline.
   * @returns Observable con la respuesta original, tras aplicar el logger.
   */
  intercept(context: ExecutionContext, next: CallHandler): Observable<any> {
    const req = context.switchToHttp().getRequest();
    const { method, originalUrl } = req;
    const now = Date.now();

    return next.handle().pipe(
      tap(() => {
        const res = context.switchToHttp().getResponse();
        const statusCode = res.statusCode;
        const delay = Date.now() - now;

        this.logger.log(
          `${method} ${originalUrl} [${statusCode}] - ${delay}ms`
        );
      })
    );
  }
}

```

Figura 100. Fragmento de código que muestra el `LoggerInterceptor` para registrar información de cada solicitud HTTP.

- Manejo de excepciones con `CustomExceptionHandler`

El módulo de seguridad implementa un manejador de excepciones personalizadas para interceptar y registrar todos los errores generados por el sistema. En Figura 101, se observa como el `CustomExceptionHandler` captura las excepciones y las registra adecuadamente.

```

@Catch()
export class CustomExceptionHandler implements ExceptionFilter {
  private readonly logger = new Logger(CustomExceptionHandler.name);

  /**
   * Método principal que intercepta cualquier excepción lanzada durante la ejecución
   * de una ruta HTTP y genera una respuesta estandarizada.
   *
   * @param exception La excepción lanzada (puede ser de cualquier tipo).
   * @param host Contexto de ejecución, usado para acceder a `Request` y `Response`.
   */
  catch(exception: Error, host: ArgumentsHost): void {
    const ctx = host.switchToHttp();
    const response = ctx.getResponse<Response>();
    const request = ctx.getRequest<Request>();
    const path = request.path;

    if (exception instanceof ValidationException) {
      this.logger.warn(`[Validation] ${exception.message}`, exception.stack);
      response.status(exception.getStatus()).send(
        CustomErrorResponse.fromValidationException(exception, path),
      );
      return;
    }

    if (exception instanceof HttpException) {
      const status = exception.getStatus();
      this.logger.error(`[HttpException] ${exception.message}`, exception.stack);
      response.status(status).send(
        CustomErrorResponse.fromHttpException(exception, path),
      );
      return;
    }

    this.logger.error(`[UnhandledError] ${exception.message}`, exception.stack);
    response.status(500).send(CustomErrorResponse.fromError(exception, path));
  }
}

```

Figura 101. Fragmento de código que muestra el CustomExceptionHandler para la captura y registro de excepciones.

El sistema también utiliza Logger para registrar las excepciones con niveles adecuados de severidad.

- Validación de datos con CustimValidationPipe

El pipe valida los datos entrantes y lanza excepciones cuando los datos no cumplen con los requisitos esperados. En *Figura 102*, se muestra la validación.

```

export class CustomValidationPipe implements PipeTransform<any> {

  /**
   * Transforma y valida los datos de entrada antes de que lleguen al controlador.
   *
   * @param value Datos recibidos (generalmente desde el body).
   * @param metatype Tipo de clase DTO contra el cual se validan los datos.
   * @returns Instancia validada del objeto.
   *
   * @throws `ValidationException` si se detectan errores de validación.
   */
  async transform(value: any, { metatype }: ArgumentMetadata) {
    if (!metatype || !this.toValidate(metatype)) {
      return value;
    }
    const object = plainToInstance(metatype, value);
    await validateOrReject(object).catch((errors: any) => {
      throw new ValidationException(errors);
    });
    return object;
  }

  /**
   * Determina si el tipo recibido debe validarse o no.
   * Se ignoran tipos primitivos como `String`, `Boolean`, etc.
   *
   * @param metatype Tipo del dato a validar.
   * @returns `true` si debe validarse, `false` si es un tipo primitivo.
   */
  private toValidate(metatype: Function): boolean {
    const types: Function[] = [String, Boolean, Number, Array, Object];
    return !types.includes(metatype);
  }
}

```

Figura 102. Fragmento de código que muestra la CustomValidationPipe para la validación de datos entrantes.

- Proxy Service y registro de errores.

Este servicio también, perteneciente al API Gateway, tienen un componente importante en el registro de errores y el monitoreo de eventos críticos. Este servicio maneja las solicitudes de proxy y, en caso de que se produzca un error, registra un mensaje detallado sobre el fallo, junto con los datos relevantes como el método de solicitudes, la URL y los headers de la solicitud.

En Figura 103, se puede observar como el ProxyService maneja y registra los errores relacionados con las solicitudes a otros microservicios. Si ocurre un error al realizar la solicitud, el servicio registra

el error en los logs y proporciona una respuesta adecuada al cliente, asegurando que los errores sean manejados correctamente.

```
on: {  
  error: (err, req, res) => {  
    this.logger.error(`[Error] Error en la solicitud`, { message: err });  
    const expressRes = res as Response;  
    if (!expressRes.headersSent) {  
      expressRes.status(500).json({  
        error: 'Error inesperado',  
        detalles: 'No se pudo procesar la solicitud',  
      });  
    }  
  },  
}
```

Figura 103. Fragmento de código que muestra el registro de errores en el ProxyService.

- Configuración de logging con Winston

Se utiliza Winston para el registro centralizado y el monitoreo del API Gateway. Los logs son registrados tanto en la consola como en archivos específicos. En Figura 104, se muestra cómo se configura Winston para registrar eventos.

```

export const logger = winston.createLogger({
  level: 'info',
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.json(),
  ),
  transports: [
    new winston.transports.Console(), // Log en consola
    new winston.transports.File({ filename: 'logs/combined.log' }),
  ],
});

if (process.env.NODE_ENV === 'development') {
  logger.add(new winston.transports.Console({
    format: winston.format.combine(
      winston.format.colorize(),
      winston.format.simple(),
    ),
  }));
}

if (process.env.NODE_ENV === 'production') {
  logger.configure({
    level: 'error',
    transports: [
      new winston.transports.File({ filename: 'logs/errors.log' }),
    ],
  });
}

```

Figura 104. Fragmento de configuración de Winston para el registro de logs en diferentes entornos.

II. Resultado

La evaluación de los registros de seguridad y el monitoreo de eventos muestra que se han implementado prácticas adecuadas para garantizar la trazabilidad de las solicitudes y la captura de excepciones críticas.

El módulo de seguridad y API Gateway de la plataforma implementa una estrategia sólida de logging y monitoreo. Se ha asegurado que todos los eventos críticos sean registrados de manera adecuada y que los errores sean manejados de forma efectiva.

2.2.10. A10 – Server-side request failure

Esta categoría hace referencia a vulnerabilidades en las que un servidor puede ser inducido a realizar solicitudes arbitrarias a recursos humanos internos o externos, muchas veces ocultos para el

cliente. Estos ataques, conocidos como Server-Side Request Forgery (SSRF), pueden ser utilizados para acceder a servicios internos, explorar redes privadas, filtrar datos sensibles o incluso ejecutar código remoto si existe servicios mal configurados.

I. Desarrollo de la evaluación

En la plataforma UTN Móvil se identificó que el componente de API Gateway cumple el rol de redireccionar todas las solicitudes entrantes a los diferentes microservicios internos. Este comportamiento es gestionado mediante la clase ProxyService, donde se determina el servicio destinado a partir del primer segmento de la URL. En *Figura 105* se puede observar el bloque de código que establece dinámicamente el destino (target) utilizando la librería http-proxy-middleware.

```
const microserviceName = [REDACTED]
const service = services[microserviceName];
if (!service) {
  throw new HttpException(
    {
      error: 'No se encontró el servicio',
      detalles: `No se encontró el servicio con el nombre ${microserviceName}`,
    },
    HttpStatus.NOT_FOUND,
  );
}

const proxy = createProxyMiddleware({
  changeOrigin: true,
  target: [REDACTED],
  pathRewrite: (path: string) => {
    const segments = path.split('/').filter(segment => segment);
    return [REDACTED] + segments.slice(1).join('/');
  },
});
```

Figura 105. Fragmento de código que muestra la redirección dinámica mediante createProxyMiddleware.

La lógica permite que cualquiera petición sea redirigida a la dirección base definida en el objeto services para ese microservicio, concatenando luego la ruta interna. Esta funcionalidad se complementa con un reescrito de rutas que elimina el prefijo del microservicio, como se muestra en *Figura 106*.

```
pathRewrite: (path: string) => {
  const segments = path.split('/').filter(segment => segment);
  return [REDACTED] + segments.slice(1).join('/');
},
```

Figura 106. Fragmento de código que reescritura de rutas para redireccionamiento transparente en el Gateway.

II. Resultado.

La arquitectura actual del Gateway en la plataforma no presenta vulnerabilidades activas de tipo SSRF. También no expone directamente parámetros manipulables que controlen las rutas internas del servidor. La evaluación determinó que la lógica de redirección basada en configuración dinámica y reescritura de rutas constituye una superficie de riesgo latente.

Se concluye que, aunque la plataforma no es vulnerable en su estado actual, existe una condición técnica que podría facilitar ataques SSRF en entornos menos controlados o con malas prácticas de configuraciones.

2.2.11. Síntesis y representación gráfica de riesgos OWASP

Como cierre de la evaluación de seguridad en OWASP Top Ten 2021, se elaboró una representación gráfica tipo mapa de calor de riesgos que se puede observar en *Figura 107*, donde se cruzan dos dimensiones fundamentales:

- Probabilidad de ocurrencia del ataque
- Impacto potencial sobre la aplicación

Este análisis se realiza sobre la base de los hallazgos obtenidos durante las pruebas de penetración, revisión de código y comportamiento de la plataforma UTN Móvil. La posición de cada vulnerabilidad se indica con un número correspondiente a su categoría.

		Impacto				
Probabilidad		Insignificante	Menor	Crítica	Mayor	Catastrófica
	Constante		A06			
	Moderado			A09	A03	
	Ocasional		A05		A08	
	Posible	A02	A01	A04		A10
	Improbable			A07		

Figura 107. Mapa de Calor de Riesgos – Evaluación OWASP UTN Móvil

A continuación, en Tabla 34 se detalla la justificación de la ubicación de cada categoría.

Tabla 34. Argumentación técnica para la clasificación de vulnerabilidades OWASP según impacto y probabilidad

CATEGORÍA OWASP 2021	UBICACIÓN	JUSTIFICACIÓN
A01	Posible/Menor	Se detectaron accesos que requiere validación, pero ya están mitigados mediante RBAC y control por roles, por lo que el riesgo residual es bajo.

A02	Posible/Insignificante	La gestión de tokens y datos sensibles cumple con buenas prácticas. No hay fugas no claves expuestas, impacto bajo.
A03	Moderado/Mayor	Se identificó riesgos medio en pruebas de inyección. Aunque controlado, su impacto potencial es alto.
A04	Posible/Critica	Algunos flujos lógicos podrían ser abusados si no se controlan ciertos parámetros, pero se ha mitigado parcialmente.
A05	Ocasional/Menor	Se detectaron detalles menores como headers y servicios innecesarios abiertos.
A06	Constante/Menor	Se evidenció el uso continuo de dependencias externas. El riesgo puede ser constante por las actualizaciones constantes.
A07	Improbable/Critica	El inicio de sesión, validaciones de códigos OTP, la autenticación biométrica local y el control de sesiones se implementaron correctamente. El impacto en caso de fallo puede ser severo.
A08	Ocasional/Mayor	Se evidenció exposición potencial a manipulación de despliegues, aunque mitigado por políticas de CI/CD controlado.
A09	Moderado/Crítico	Aunque existen logs y trazabilidad, su cobertura puede expandirse aun más.
A10	Posible/Catastrófica	El uso del Gateway como proxy externo permite ciertos abusos si no se controlan rutas dinámicas. Aunque mitigado, el impacto sería crítico.

2.2.12. Resumen del cumplimiento general de la evaluación al módulo de seguridad para la plataforma UTN Móvil.

Con el fin de evidenciar el cumplimiento de todas las categorías OWASP se proporcionó un resumen en Tabla 35. Resumen del cumplimiento general de la evaluación al módulo de seguridad de la

plataforma UTN Móvil., utilizando una escala de Likert. Esta escala permite calificar el cumplimiento de categoría, donde:

- 1 - No cumple: Requiere mejoras significativas.
- 2 - Cumple parcialmente: Cumple parcialmente, pero necesita mejoras sustanciales.
- 3 – Cumple adecuadamente: Cumple con los requisitos, pero hay áreas de mejoras.
- 4 – Cumple bien: Implementado de manera adecuada, con pocas mejoras necesarias.
- 5 – Cumple completamente: Implementado de manera óptima, sin necesidad de mejora.

A continuación, se presenta los resultados:

Tabla 35. Resumen del cumplimiento general de la evaluación al módulo de seguridad de la plataforma UTN Móvil.

CATEGORÍA OWASP 2021	CUMPLIMIENTO	MOTIVO DE MEJORA (SI APLICA)
A01	5	El control de acceso fue implementado correctamente con RBCA. Se recomienda realizar revisión continua de los roles.
A02	5	La gestión de tokens y datos sensibles siguen buenas prácticas. Se recomienda siempre verificar la exposición de calves.
A03	4	Los riesgos de inyección están completamente controlados. Se recomienda realizar y mantener las pruebas de penetración actualizadas.
A04	5	El control de parámetros esta implementado. Se recomienda mantenerlo monitoreado.
A05	4	Detalles menores encontrados relacionado a su uso innecesarios, los cuales han sido desactivados. Se recomienda mantener todos los servicios disponibles y eliminar los innecesarios.
A06	5	La dependencia de paquetes externos es constante, por lo tanto, es recomendado mantener actualizado esto mediante herramientas de apoyo.
A07	5	El MFA se encuentra completamente implementado. Su impacto en caso de fallo sería severo.
A08	5	La exposición potencial está mitigada por políticas CI/CD, pero su vigilancia debe ser constante.

A09	4	La cobertura de logs y trazabilidad es adecuada, aunque podría expandirse para una mayor seguridad.
A10	5	El riesgo de abuso del Gateway esta mitigado. Se recomienda un control riguroso en el registro dinámico de las rutas.

3.3. Validación de principios CIA en la plataforma UTN Móvil.

En esta sección se evalúa el grado de cumplimiento de estos principios en la solución tecnológica de UTN Móvil mediante un enfoque deductivo-comparativo y apoyado en evidencia práctica recopilada durante las evaluaciones OWASP Top Ten.

3.3.1. Confidencialidad

El principio de confidencialidad busca asegurar que la información solo sea accesible por personas autorizados. La plataforma está fortalecida por lo siguiente:

- Autenticación basada en tokens con algoritmos síncronos y asíncronos.
- Separación de roles y políticas RBAC.
- Manejo de claves secretas dinámicas.
- Encriptación de datos sensibles.

Durante las pruebas de seguridad de las categorías A01, A02 y A08, no se detectaron fugas de información. Se estima un cumplimiento del 100% de confidencialidad.

3.3.2. Integridad

La integridad garantiza que los datos no han sido alterados de forma no autorizada. Este principio es particularmente importante en sistemas donde las decisiones se basan en registros almacenados, como el portafolio académico de la UTN.

La plataforma refuerza la integridad mediante:

- Validación de entrada de datos.
- Registros de eventos relevantes.
- Diseño modular separado en capas.

No obstante, durante la revisión técnica también se identificaron áreas de mejora, las cuales por política de seguridad no son mencionadas. Por estas razones, se estima que el principio de integridad está parcialmente implementado, con un grado de cumplimiento aproximado del 75%.

3.3.3. Disponibilidad

El principio de disponibilidad busca asegurar que los servicios estén siempre accesibles para los usuarios autorizados, incluso ante fallas o ataques. La plataforma cuenta con una arquitectura que prioriza este principio mediante:

- Desacoplamiento de servicios mediante microservicio.
- Uso de Redis para manejo de sesiones y tolerancia a fallos temporales.
- Implementación de un API Gateway con control centralizado, balanceo de carga y control de error.
- Mecanismos de logging e interceptores.

No se evidenciaron fallas estructurales que comprometieran la disponibilidad durante las pruebas realizadas. Se considera que el cumplimiento de este principio también alcanza un 100%.

3.3.4. Matriz de cumplimiento y análisis comparativo CIA.

Para sintetizar la evaluación realizada, se presenta en Tabla 36 una matriz de doble entrada que compara el cumplimiento de los tres principios frente a los mecanismos implementados. Se asigna un nivel cualitativo al cumplimiento de cada principio, en función de los controles técnicos detectados, la documentación del código fuente y los resultados obtenidos durante las pruebas de seguridad.

Tabla 36. Matriz de cumplimiento de principios CIA en la plataforma UTN Móvil.

PRINCIPIO	ELEMENTOS EVALUADOS	EVIDENCIA TECNICA	NIVEL DE CUMPLIMIENTO
Confidencialidad	Tokens de acceso, refresh tokens, uso de API Keys, control por roles	Firma síncrona y asíncrona, uso de headers seguros, enrutamiento protegido, MFA habilitado	100%
Integridad	Firmas, validación de payloads, registros, control de despliegue	Logger Interceptor, DTOs validados, hash de integridad en tokens, control de errores	75%
Disponibilidad	Gateway balanceado, arquitectura modular, manejo de errores, Redis	Proxy dinámico, fallback controlado, Redis activo, rutas críticas estable	100%

Además, se incluye un diagrama de pastel mostrado en Figura 108 que muestra la proporción relativa de cada principio dentro del total de medidas de seguridad aplicadas, normalizado para que la suma represente el 100% del cumplimiento general:

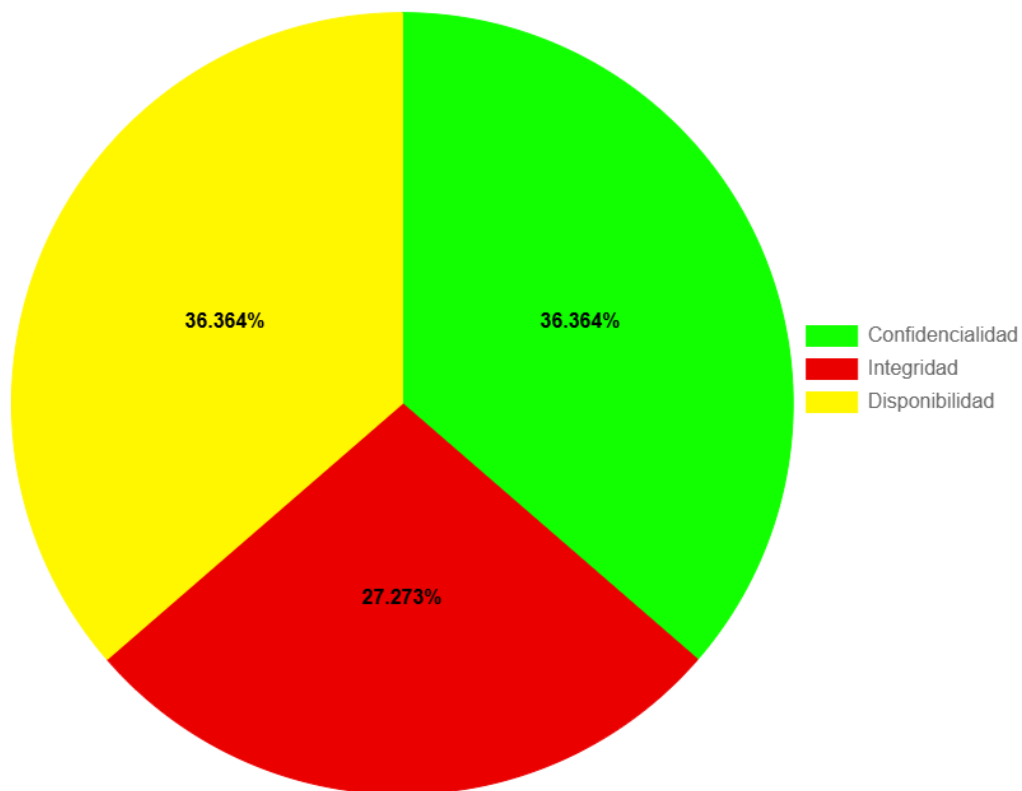


Figura 108. Distribución porcentual del cumplimiento de principios CIA en la plataforma UTN Móvil.

La representación visual permite observar que el sistema se ha centrado en proteger el acceso a la información (C) y asegurar la operatividad continua (D). La integridad (I) se puede fortalecerse mediante prácticas de verificación estructural.

CONCLUSIONES

La elaboración de un marco conceptual fue esencial para delimitar los componentes críticos de la arquitectura de seguridad en la plataforma UTN Móvil, permitiendo seleccionar metodologías, estándares y herramientas apropiadas para su análisis. Este enfoque facilitó la comprensión integral del sistema, tanto desde el punto de vista funcional como estructural, y evidenciando su alineación con las tendencias actuales de desarrollo seguro en aplicaciones móviles.

La plataforma permite que el desarrollo se mantenga desacoplado, escalable y adaptable. La organización permite la incorporación de nuevas funcionalidades sin comprometer la estabilidad de la plataforma.

La verificación del cumplimiento de las recomendaciones de OWASP Top Ten 2021 permitió aplicar enfoque metodológico para validar la protección de la plataforma frente a amenazas reales. Todas las pruebas realizadas en el capítulo 3 evidenciaron que la plataforma incorpora múltiples capas de defensa.

Los hallazgos encontrados fueron organizados en una matriz de evaluación OWASP. Un mapa de calor de riesgo y una matriz de cumplimiento permite cuantificar el grado de cobertura de la plataforma. El resultado del análisis refleja el enfoque prioritario del sistema, desde la protección de los datos hasta la validación de identidad del usuario.

Todo el trabajo realizado ha permitido validar que la plataforma no solo responde a necesidades tecnológicas de la institución, sino que se encuentra también alineado con las buenas prácticas de seguridad.

RECOMENDACIONES

Se recomienda realizar actualizaciones periódicas de la aplicación UTN Móvil, tanto en el fronted como en los microservicios backend, para adaptarse a nuevas necesidades académicas, cambios tecnológicos y vulnerabilidades emergentes. Esto incluye la actualización constante de librerías y dependencias, como se evidenció en la evaluación de componentes vulnerables.

Con el crecimiento progresivo de la plataforma, se sugiere implementar una estrategia controlada de escalabilidad. Utilizar Kubernetes garantiza una alta disponibilidad y mantenibilidad eficiente.

En cuanto a la arquitectura, se recomienda continuar promoviendo el uso de una estructura modular en capas y la aplicación de buenas prácticas como la inyección de dependencias, que faciliten la mantenibilidad, trazabilidad y auditoría de seguridad, tal como se evidenció en el análisis de diseño seguro.

Se sugiere realizar evaluaciones periódicas de percepción y de experiencia de los usuarios. La retroalimentación funcional permitirá alinear el desarrollo de la plataforma con las expectativas institucionales.

REFERENCIAS

- [1] N. Castro and P. Hidalgo, "Educación Transmedia: Confinamiento, Aprendizaje y Nuevas Plataformas," p. 4, Apr. 2021, Accessed: Feb. 19, 2024. [Online]. Available: <https://revista.consejodecomunicacion.gob.ec/index.php/rec/article/view/26/73>
- [2] Symantec, "Executive Summary 2019 Internet Security Threat Report," vol. 24, Feb. 2019, Accessed: Feb. 19, 2024. [Online]. Available: <https://docs.broadcom.com/doc/istr-24-executive-summary-en>
- [3] OWASP, "OWASP Top 10:2021." Accessed: Feb. 19, 2024. [Online]. Available: <https://owasp.org/Top10/>
- [4] A. Bogantes, "El rol de la seguridad informática en el ámbito académico y los sistemas de información asociados," vol. 17, p. 28, 2020, Accessed: Mar. 25, 2024. [Online]. Available: <https://www.iiisci.org/journal/PDV/risci/pdfs/CB294NT20.pdf>
- [5] R. V. O'Connor, P. Elger, and P. M. Clarke, "Continuous software engineering—A microservices architecture perspective," *Journal of Software: Evolution and Process*, vol. 29, no. 11, Nov. 2019, doi: 10.1002/SMR.1866.
- [6] Programa De Las Naciones Unidas Para El Desarrollo, "Objetivos de Desarrollo Sostenible." Accessed: Mar. 25, 2024. [Online]. Available: <https://www.undp.org/es/sustainable-development-goals>
- [7] Z. M. Guitarra De La Cruz, "Creación de una aplicación Backend del módulo de transferencias (Cuentas y Montos), con microservicios GRAPHQL y REST en contenedores Docker, para fomentar el levantamiento de una arquitectura DevOps," Universidad Técnica del Norte, Ibarra, 2022. Accessed: Apr. 27, 2024. [Online]. Available: <https://repositorio.utn.edu.ec/bitstream/123456789/13293/2/04%20ISC%20665%20TRABAJO%20DE%20GRADO.pdf>
- [8] M. A. Escobar Quintana and R. O. Alarcón Pérez, "Desarrollo de un Sistema Web y un Módulo de Seguridad basado en la norma ISO 27000, para mitigar y controlar la integridad de la información," Escuela Superior Politécnica de Chimborazo, Riobamba, 2023. Accessed: Apr. 27, 2024. [Online]. Available: <http://dspace.esPOCH.edu.ec/bitstream/123456789/20158/1/20T01840.pdf>
- [9] O. Chaplia and H. Klym, "Designing a Node.js Project Architecture for RESTful Microservices," *Proceedings of the IEEE International Conference on Intelligent Data Acquisition and Advanced*

- Computing Systems: Technology and Applications*, IDAACS, pp. 808–811, 2023, doi: 10.1109/IDAACS58523.2023.10348681.
- [10] N. Ščnger and S. Abeck, “Authentication and Authorization in Microservice-Based Applications,” 2022, doi: 10.18420/inf2022_19.
 - [11] P. Andrés and R. Chiriboga, “Autenticación de Microservicios Autenticación de microservicios basados en OAuth,” 2022, Accessed: Mar. 25, 2024. [Online]. Available: <https://openaccess.uoc.edu/bitstream/10609/147247/4/prodriguezchiTFM0123memoria.pdf>
 - [12] Universidad Técnica del Norte, “Reglamento Portafolio Universitario,” 2014, Accessed: Jun. 06, 2024. [Online]. Available: https://legislacion.utn.edu.ec/wp-content/uploads/2017/12/2.REGLAMENTO_PORTAFOLIO_UNIVERSITARIO-1.pdf
 - [13] Universidad Técnica del Norte, “Plan Estratégico de Desarrollo Institucional (PEDI),” 2020, Accessed: Jun. 06, 2024. [Online]. Available: <https://transparencia.utn.edu.ec/wp-content/uploads/2024/01/PEDI-2021-2025.pdf>
 - [14] C. Sørensen, M. de Reuver, and R. C. Basole, “Mobile platforms and ecosystems,” *Journal of Information Technology*, vol. 30, no. 3, pp. 195–197, 2015, doi: 10.1057/jit.2015.22.
 - [15] L. Herazo, “¿Qué es una aplicación móvil?,” Anincubator. Accessed: Jul. 10, 2024. [Online]. Available: <https://anincubator.com/que-es-una-aplicacion-movil/>
 - [16] C. Latina Revista Científica Multidisciplinar *et al.*, “Desarrollo Híbrido con Flutter,” *Ciencia Latina Revista Científica Multidisciplinar*, vol. 6, no. 4, pp. 4594–4609, Sep. 2022, doi: 10.37811/CL_RCM.V6I4.2959.
 - [17] E. Apostolidou and P. A. Fokaides, “Enhancing Accessibility: A Comprehensive Study of Current Apps for Enabling Accessibility of Disabled Individuals in Buildings,” *Buildings* 2023, Vol. 13, Page 2085, vol. 13, no. 8, p. 2085, Aug. 2023, doi: 10.3390/BUILDINGS13082085.
 - [18] B. Zhang, G. Peng, F. Xing, and S. Chen, “Mobile Applications in China’s Smart Cities: State-of-the-Art and Lessons Learned,” <https://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/JGIM.20211101.0a26>, vol. 29, no. 6, pp. 1–18, Jan. 2021, doi: 10.4018/JGIM.20211101.OA26.
 - [19] P. Meirelles, C. S. R. Aguiar, F. Assis, R. Siqueira, and A. Goldman, “A Students’ Perspective of Native and Cross-Platform Approaches for Mobile Application Development,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11623 LNCS, pp. 586–601, 2019, doi: 10.1007/978-3-030-24308-1_47.

- [20] M. Martinez, "Two Datasets of Questions and Answers for Studying the Development of Cross-Platform Mobile Applications using Xamarin Framework," in *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, IEEE, May 2019, pp. 162–173. doi: 10.1109/MOBILESoft.2019.00032.
- [21] J. L. Moranga, "Apps más visuales y rápidas con Flutter," Deloitte España. Accessed: Jul. 08, 2024. [Online]. Available: <https://www2.deloitte.com/es/es/pages/technology/articles/apps-mas-visuales-y-rapidas-con-flutter.html>
- [22] M. Sabin, K. Buffardi, S. Larsen, and B. MacKellar, "Panel: Synergistic Perspectives on Collaborative Learning with Version Control Tools," *Proceedings - Frontiers in Education Conference, FIE*, vol. 2020-October, Oct. 2020, doi: 10.1109/FIE44824.2020.9273944.
- [23] S. S. Ragavan, M. Codoban, D. Piorkowski, D. Dig, and M. Burnett, "Version Control Systems: An Information Foraging Perspective," *IEEE Transactions on Software Engineering*, vol. 47, no. 8, pp. 1644–1655, Aug. 2021, doi: 10.1109/TSE.2019.2931296.
- [24] N. Deepa, B. Prabadevi, L. B. Krithika, and B. Deepa, "An analysis on Version Control Systems," *International Conference on Emerging Trends in Information Technology and Engineering, ic-ETITE 2020*, Feb. 2020, doi: 10.1109/IC-ETITE47903.2020.39.
- [25] A. Tyagi, "Tutorial 101: Git Push and Pull - Analytics Vidhya," Analytics Vidhya. Accessed: Sep. 01, 2024. [Online]. Available: <https://www.analyticsvidhya.com/blog/2024/01/heres-everything-about-git-push-and-pull/>
- [26] R. Majumdar, R. Jain, S. Barthwal, and C. Choudhary, "Source code management using version control system," *2017 6th International Conference on Reliability, Infocom Technologies and Optimization: Trends and Future Directions, ICRITO 2017*, vol. 2018-January, pp. 278–281, Apr. 2018, doi: 10.1109/ICRITO.2017.8342438.
- [27] A. Chernetsov, O. Ivanov, and D. Shevtsov, "Development and Implementation of FRL Language Interpreter and Integrated Development Environment," pp. 1–4, Jun. 2024, doi: 10.1109/INFORINO60363.2024.10551944.
- [28] G. A. Aye and G. E. Kaiser, "Sequence Model Design for Code Completion in the Modern IDE," Apr. 2020, doi: 10.1145/nnnnnnnn.nnnnnnnn.
- [29] F. F. Xu, B. Vasilescu, and G. Neubig, "In-IDE Code Generation from Natural Language: Promise and Challenges," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 2,

- Apr. 2022, doi: 10.1145/3487569/ASSET/638FDC26-1092-486A-909A-3A7E5C298858/ASSETS/GRAPHIC/TOSEM3102-29-F10.JPG.
- [30] H. Taniguchi and K. Nakasho, "Visual Studio Code Extension and Auto-completion for Mizar Language," *Proceedings - 2021 9th International Symposium on Computing and Networking, CANDAR 2021*, pp. 182–188, 2021, doi: 10.1109/CANDAR53791.2021.00033.
 - [31] M. G. Prakash, A. Ponmalar, S. Deeba, A. Akilandeswari, S. B. M. Rasool, and P. Lavanya, "VSCODE- Code With Voice Using Natural Language Processing(NLP)," *2022 1st International Conference on Computer, Power and Communications, ICCPC 2022 - Proceedings*, pp. 571–574, 2022, doi: 10.1109/ICCPC55978.2022.10072106.
 - [32] H. Bai, "VSC-WebGPU: A Selenium-based VS Code Extension for Local Edit and Cloud Compilation on WebGPU," *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer, ICFTIC 2021*, pp. 474–477, 2021, doi: 10.1109/ICFTIC54370.2021.9647189.
 - [33] Oracle México, "Oracle Database," Oracle. Accessed: Jul. 08, 2024. [Online]. Available: <https://www.oracle.com/mx/database/>
 - [34] M. Rouse, "Oracle Database," Techopedia. Accessed: Jul. 08, 2024. [Online]. Available: <https://www.techopedia.com/definition/8711/oracle-database>
 - [35] X. J. Hong, H. Sik Yang, and Y. H. Kim, "Performance Analysis of RESTful API and RabbitMQ for Microservice Web Application," *9th International Conference on Information and Communication Technology Convergence: ICT Convergence Powered by Smart Intelligence, ICTC 2018*, pp. 257–259, Nov. 2018, doi: 10.1109/ICTC.2018.8539409.
 - [36] I. Ahmad, E. Suwarni, R. I. Borman, Asmawati, F. Rossi, and Y. Jusman, "Implementation of RESTful API Web Services Architecture in Takeaway Application Development," *2021 1st International Conference on Electronic and Electrical Engineering and Intelligent System, ICE3IS 2021*, pp. 132–137, 2021, doi: 10.1109/ICE3IS54102.2021.9649679.
 - [37] G. Vega-Gorgojo, "CRAFTS: Configurable REST APIs for Triple Stores," *IEEE Access*, vol. 10, pp. 32426–32441, 2022, doi: 10.1109/ACCESS.2022.3160610.
 - [38] T. Johnson, "Documenting APIs: A guide for technical writers and engineers." Accessed: Jul. 10, 2024. [Online]. Available: <https://idratherbewriting.com/learnapidoc/>

- [39] G. Brito and M. T. Valente, "REST vs GraphQL: A controlled experiment," *Proceedings - IEEE 17th International Conference on Software Architecture, ICSA 2020*, pp. 81–91, Mar. 2020, doi: 10.1109/ICSA47634.2020.00016.
- [40] T. Brito *et al.*, "Study of JavaScript Static Analysis Tools for Vulnerability Detection in Node.js Packages," *IEEE Trans Reliab*, vol. 72, no. 4, pp. 1324–1339, Dec. 2023, doi: 10.1109/TR.2023.3286301.
- [41] Node.org, "About Node.js." Accessed: Jul. 10, 2024. [Online]. Available: <https://nodejs.org/en/about>
- [42] I. Kniazev and A. Fitiskin, "CHOOSING THE RIGHT JAVASCRIPT RUNTIME: AN IN-DEPTH COMPARISON OF NODE.JS AND BUN," *Norwegian Journal of development of the International Science*, vol. 108, 2023, doi: 10.5281/ZENODO.7945166.
- [43] M. Martínez Canelo, "Qué es Node.js: Introducción al JavaScript en el lado del servidor," Profile Software Services. Accessed: Sep. 01, 2024. [Online]. Available: <https://profile.es/blog/que-es-nodejs/>
- [44] T. Sufiyan, "Understanding Node.js Architecture," Simpli Learn. Accessed: Jul. 10, 2024. [Online]. Available: <https://www.simplilearn.com/understanding-node-js-architecture-article>
- [45] L. Trivedi, "Getting Started with NestJS: A Complete Guide with Coding Examples," Medium. Accessed: Apr. 22, 2025. [Online]. Available: <https://medium.com/zestgeek/getting-started-with-nestjs-a-complete-guide-with-coding-examples-d30c678b0053>
- [46] AJ Software Engineering, "NestJS: The Perfect Javascript Backend Framework for Structure, Clean Code, and Modularity," Medium. Accessed: Apr. 22, 2025. [Online]. Available: <https://medium.com/@ajonesb/nestjs-the-perfect-javascript-backend-framework-for-structure-clean-code-and-modularity-ae1b3a6e1418>
- [47] S. Al-Saqqa, S. Sawalha, and H. Abdelnabi, "Agile Software Development: Methodologies and Trends," *International Journal of Interactive Mobile Technologies (IJIM)*, vol. 14, no. 11, pp. 246–270, Jul. 2020, doi: 10.3991/IJIM.V14I11.13269.
- [48] Rajib. Mall and PHI Learning., *Fundamentals of software engineering*, 5th ed. PHI Learning Private Limited, 2018. Accessed: Jun. 16, 2024. [Online]. Available: https://books.google.com/books/about/FUNDAMENTALS_OF_SOFTWARE_ENGINEERING_FIF.html?id=-JNuDwAAQBAJ

- [49] G. S. Matharu, A. Mishra, H. Singh, and P. Upadhyay, "Empirical Study of Agile Software Development Methodologies," *ACM SIGSOFT Software Engineering Notes*, vol. 40, no. 1, pp. 1–6, Feb. 2015, doi: 10.1145/2693208.2693233.
- [50] N. Govil and A. Sharma, "Validation of agile methodology as ideal software development process using Fuzzy-TOPSIS method," *Advances in Engineering Software*, vol. 168, p. 103125, Jun. 2022, doi: 10.1016/J.ADVENGSOFT.2022.103125.
- [51] C. Verwijs and D. Russo, "Do Agile scaling approaches make a difference? an empirical comparison of team effectiveness across popular scaling approaches," *Volume 29, Issue 4*, vol. 29, no. 4, Jan. 1975, doi: 10.1007/s10664-024-10481-5.
- [52] J. N. Meckenstock, "Shedding light on the dark side – A systematic literature review of the issues in agile software development methodology use," *Journal of Systems and Software*, vol. 211, p. 111966, May 2024, doi: 10.1016/J.JSS.2024.111966.
- [53] T. Natarajan and S. Pichai, "Transition from Waterfall to Agile Methodology: An Action Research Study," *Volume 12, Pages 49341 - 49362*, vol. 12, pp. 49341–49362, 2024, doi: 10.1109/ACCESS.2024.3384097.
- [54] B. Shahzad, W. N. Awan, Fazal-E-Amin, A. Abro, M. Shoaib, and S. Alyahya, "Framework for Effective Utilization of Distributed Scrum in Software Projects," *Volume 44, Issue 1, Pages 407 - 422*, vol. 44, no. 1, pp. 407–422, 2022, doi: 10.32604/csse.2023.022601.
- [55] W. Omar Figueroa Escudero, M. Pérez Vasconcelos, F. Castillo Romero, R. Gómez Domínguez, and E. Sosa Silva, "Metodología SCRUM aplicada en el desarrollo del módulo de Consulta Externa del Sistema Integral Hospital Roviroso (SIHR)," 2021.
- [56] Scrum.org, "What is Scrum? ." Accessed: Jun. 17, 2024. [Online]. Available: <https://www.scrum.org/learning-series/what-is-scrum/>
- [57] F. Anwer, S. Aftab, S. S. Muhammad Shah, and U. Waheed, "Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum," *International Journal of Computer Science and Telecommunications*, vol. 8, no. 2, Mar. 2017, Accessed: Mar. 12, 2025. [Online]. Available: https://www.ijcst.org/Volume8/Issue2/p1_8_2.pdf
- [58] H. De Chandler, "Artefactos del scrum ágil," Atlassian. Accessed: Mar. 12, 2025. [Online]. Available: <https://www.atlassian.com/es/agile/scrum/artifacts>
- [59] Sybven, "Scrum vs Agile - Sybven." Accessed: Mar. 12, 2025. [Online]. Available: <https://sybven.com/que-es-scrum/>

- [60] H. Kong, T. Li, J. Ge, L. Zhang, and L. Li, "Enhancing fault localization in microservices systems through span-level using graph convolutional networks," *Automated Software Engineering*, vol. 31, no. 2, pp. 1–26, Nov. 2024, doi: 10.1007/S10515-024-00445-W/TABLES/3.
- [61] K. Gos and W. Zabierowski, "The Comparison of Microservice and Monolithic Architecture," *International Conference on Perspective Technologies and Methods in MEMS Design*, pp. 150–153, 2020, doi: 10.1109/MEMSTECH49584.2020.9109514.
- [62] I. A. Kautsar, M. R. Maika, A. N. Budiman, A. B. Setyawan, and J. Y. Awali, "Microservice Based Architecture: The Development of Rapid Prototyping Supportive Tools for Project Based Learning," *EDUNINE 2023 - 7th IEEE World Engineering Education Conference: Reimaging Engineering - Toward the Next Generation of Engineering Education, Merging Technologies in a Connected World, Proceedings*, 2023, doi: 10.1109/EDUNINE57531.2023.10102884.
- [63] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *Journal of Systems and Software*, vol. 182, p. 111061, Dec. 2021, doi: 10.1016/J.JSS.2021.111061.
- [64] H. Michael Ayas, R. Hebig, and P. Leitner, "An empirical investigation on the competences and roles of practitioners in Microservices-based Architectures," *Journal of Systems and Software*, vol. 213, p. 112055, Jul. 2024, doi: 10.1016/J.JSS.2024.112055.
- [65] M. I. Josélyne, D. Tuheirwe-Mukasa, B. Kanagwa, and J. Balikuddembe, "Partitioning microservices," in *Proceedings of the 2018 International Conference on Software Engineering in Africa*, New York, NY, USA: ACM, May 2018, pp. 43–49. doi: 10.1145/3195528.3195535.
- [66] P. Di Francesco, P. Lago, and I. Malavolta, "Architecting with microservices: A systematic mapping study," *Journal of Systems and Software*, vol. 150, pp. 77–97, Apr. 2019, doi: 10.1016/j.jss.2019.01.001.
- [67] N. Dragoni *et al.*, "Microservices: Yesterday, Today, and Tomorrow," in *Present and Ulterior Software Engineering*, Cham: Springer International Publishing, 2017, pp. 195–216. doi: 10.1007/978-3-319-67425-4_12.
- [68] A. M. Alashqar and Z. Kurdya, "Examining the Design Patterns of Microservices for Achieving Performance Quality Tactics," vol. 6, pp. 4–13, 2022, Accessed: Jun. 28, 2024. [Online]. Available: <https://www.researchgate.net/publication/366780406>

- [69] R. Pincirolì, A. Aletì, and C. Trubiani, "Performance Modeling and Analysis of Design Patterns for Microservice Systems," *Proceedings - IEEE 20th International Conference on Software Architecture, ICSA 2023*, pp. 35–46, 2023, doi: 10.1109/ICSA56044.2023.00012.
- [70] G. Vale, F. F. Correia, E. M. Guerra, T. De Oliveira Rosa, J. Fritzsche, and J. Bogner, "Designing Microservice Systems Using Patterns: An Empirical Study on Quality Trade-Offs," *Proceedings - IEEE 19th International Conference on Software Architecture, ICSA 2022*, pp. 69–79, 2022, doi: 10.1109/ICSA53651.2022.00015.
- [71] P. D. Parente, "The role of an API gateway in a microservice architecture," Dec. 2022, Accessed: Jun. 28, 2024. [Online]. Available: <https://repositorium.sdum.uminho.pt/handle/1822/84396>
- [72] J. Arnedo-Moreno and V. Garcia-Font, "A Study on the Design and Application of Fictional Storytelling in Online Learning of Computer Security," *Applied Sciences* 2021, Vol. 11, Page 6185, vol. 11, no. 13, p. 6185, Jul. 2021, doi: 10.3390/APP11136185.
- [73] R. Villalón-Fonseca, "The nature of security: A conceptual framework for integral-comprehensive modeling of IT security and cybersecurity," *Comput Secur*, vol. 120, p. 102805, Sep. 2022, doi: 10.1016/J.COSE.2022.102805.
- [74] M. G. Cains, L. Flora, D. Taber, Z. King, and D. S. Henshel, "Defining Cyber Security and Cyber Security Risk within a Multidisciplinary Context using Expert Elicitation," *Risk Analysis*, vol. 42, no. 8, pp. 1643–1669, Aug. 2022, doi: 10.1111/RISA.13687.
- [75] International Telecommunications Union (ITU), "Overview of cybersecurity," *Series X: Data Networks, Open System Communications and Security – Telecommunication Security.*, 2008, Accessed: Jun. 13, 2024. [Online]. Available: <https://www.itu.int/rec/T-REC-X.1205-200804-I>
- [76] F. N. Solarte Solarte, E. R. Enriquez Rosero, and M. Del Carmen Benavides, "Metodología de análisis y evaluación de riesgos aplicados a la seguridad informática y de información bajo la norma ISO/IEC 27001," *Análisis de seguridad de bases de datos*, vol. 28, no. 5, 2015, Accessed: Jul. 08, 2024. [Online]. Available: <https://rte.espol.edu.ec/index.php/tecnologica/article/view/456>
- [77] IAPLIKADA, "Servicios de Seguridad de la Información," Informática Aplicada Consulting Services. Accessed: Sep. 01, 2024. [Online]. Available: <https://www.iaplikada.com/soluciones-seguridad-informatica/>

- [78] National Institute of Standards and Technology, "Information Assurance (IA) - Glossary | CSRC," U.S. Department of Commerce. Accessed: Jun. 13, 2024. [Online]. Available: https://csrc.nist.gov/glossary/term/information_assurance
- [79] M. M. Chowdhury, N. Rifat, M. Ahsan, S. Latif, R. Gomes, and M. S. Rahman, "ChatGPT: A Threat Against the CIA Triad of Cyber Security," *IEEE International Conference on Electro Information Technology*, vol. 2023-May, pp. 478–483, 2023, doi: 10.1109/EIT57321.2023.10187355.
- [80] M. Humayun, M. Niazi, N. Jhanjhi, M. Alshayeb, and S. Mahmood, "Cyber Security Threats and Vulnerabilities: A Systematic Mapping Study," *Arab J Sci Eng*, vol. 45, no. 4, pp. 3171–3189, Apr. 2020, doi: 10.1007/S13369-019-04319-2/TABLES/9.
- [81] M. G. de Almeida and E. D. Canedo, "Authentication and Authorization in Microservices Architecture: A Systematic Literature Review," *Applied Sciences*, vol. 12, no. 6, p. 3023, Mar. 2022, doi: 10.3390/app12063023.
- [82] A. Pereira-Vale, E. B. Fernandez, R. Monge, H. Astudillo, and G. Márquez, "Security in microservice-based systems: A Multivocal literature review," *Comput Secur*, vol. 103, p. 102200, Apr. 2021, doi: 10.1016/j.cose.2021.102200.
- [83] D. Berardi, S. Giallorenzo, J. Mauro, A. Melis, F. Montesi, and M. Prandini, "Microservice security: a systematic literature review," *PeerJ Comput Sci*, vol. 7, p. e779, Jan. 2022, doi: 10.7717/peerj-cs.779.
- [84] R. A. Al-Wadi and A. A. Maaita, "Authentication and Role-Based Authorization in Microservice Architecture: A Generic Performance-Centric Design," *Volume 14, Issue 4, Pages 758 - 768*, vol. 14, no. 4, pp. 758–768, 2023, doi: 10.12720/jait.14.4.758-768.
- [85] A. Barabanov and D. Makrushin, "Authentication and authorization in microservice-based systems: survey of architecture patterns," *Voprosy kiberbezopasnosti*, no. 4(38), pp. 32–43, 2020, doi: 10.21681/2311-3456-2020-04-32-43.
- [86] K. R. Rao, A. Nayak, I. G. Ray, Y. Rahulamathavan, and M. Rajarajan, "Role recommender-RBAC: Optimizing user-role assignments in RBAC," *Comput Commun*, vol. 166, pp. 140–153, Jan. 2021, doi: 10.1016/J.COMCOM.2020.12.006.
- [87] C. Urrea Stabile, "What's the Right Authorization Model for My Application?," AuthO by Okta. Accessed: Jun. 29, 2024. [Online]. Available: <https://auth0.com/blog/whats-the-right-authorization-model-for-my-application/>

- [88] Mohasina, "AWS Identity Center: Access Made Easy," Medium. Accessed: Jul. 10, 2024. [Online]. Available: <https://medium.com/@mohasina.clt/aws-identity-center-access-made-easy-900bf13eb7bd>
- [89] P. Kamboj, S. Khare, and S. Pal, "User authentication using Blockchain based smart contract in role-based access control," *Peer Peer Netw Appl*, vol. 14, no. 5, pp. 2961–2976, Sep. 2021, doi: 10.1007/S12083-021-01150-1/METRICS.
- [90] OAuth, "OAuth 2.0." Accessed: Jul. 08, 2024. [Online]. Available: <https://oauth.net/2/>
- [91] JWT, "JSON Web Token Introduction." Accessed: Jul. 09, 2024. [Online]. Available: <https://jwt.io/introduction>
- [92] IONOS, "JSON Web Token (JWT): presentación, explicación y ejemplos," IONOS España. Accessed: Jul. 10, 2024. [Online]. Available: <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/json-web-token-jwt/>
- [93] OWASP, "About the OWASP Foundation." Accessed: Jul. 09, 2024. [Online]. Available: <https://owasp.org/about/>
- [94] OWASP, "OWASP Top Ten," OWASP Foundation. Accessed: Jul. 09, 2024. [Online]. Available: <https://owasp.org/www-project-top-ten/>
- [95] M. Cohn, "Agile Estimating and Planning," Google Books. Accessed: Apr. 01, 2025. [Online]. Available: https://books.google.com.ec/books/about/Agile_Estimating_and_Planning.html?id=BuFWHffRJssC&redir_esc=y
- [96] Endo Flife, "Redis," endoflife.date. Accessed: Jun. 23, 2025. [Online]. Available: <https://endoflife.date/redis>
- [97] Endo Fifle, "Oracle Database," endofifle.date. Accessed: Jun. 23, 2025. [Online]. Available: <https://endoflife.date/oracle-database>