UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas
Carrera de Ingeniería en Sistemas Computacionales

ANÁLISIS COMPARATIVO DE LOS FRAMEWORKS JAVASCRIPT REACT Y VUE.JS MEDIANTE LAS NORMAS ISO/IEC 25010, PARA EL DESARROLLO DEL SISTEMA WEB DE CONTROL DE VENTAS PARA LA PANIFICADORA ROYAL.

Trabajo de grado previo a la obtención del título de Ingeniero en Sistemas Computacionales

Autor:

Javier Enrique Paucar Mayanquer

Directora:

MSc. Daisy Elizabeth Imbaquingo Esparza

Ibarra-Ecuador

2023

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:	0401544085		
APELLIDOS Y NOMBRES:	Paucar Mayanquer Javier Enrique		
DIRECCIÓN:	El Ángel, Av. Los Pastos Primera Transversal		
EMAIL:	jepaucarm@utn.edu.ec		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0990144484

DATOS DE LA OBRA	
TÍTULO:	ANÁLISIS COMPARATIVO DE LOS FRAMEWORKS JAVASCRIPT REACT Y VUE.JS MEDIANTE LAS NORMAS ISO/IEC 25010, PARA EL DESARROLLO DEL SISTEMA WEB DE CONTROL DE VENTAS PARA LA PANIFICADORA ROYAL
AUTOR (ES):	Paucar Mayanquer Javier Enrique
FECHA: DD/MM/AAAA	21/06/2023
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero En Sistemas Computacionales
ASESOR /DIRECTOR:	MSc. Daisy Elizabeth Imbaquingo Esparza

2. CONSTANCIAS

El autor (es) manifiesta (n) 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 (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 13 días del mes de julio de 2023

EL AUTOR:

Nombre: Javier Enrique Paucar Mayanquer
CI: 0401544085

CERTIFICACIÓN DIRECTOR DE TRABAJO DE TITULACIÓN

Ibarra, 21 de junio de 2023

MSc. Daisy Elizabeth Imbaquingo Esparza

DIRECTORA DE TRABAJO DE TITULACIÓN

CERTIFICA:

Que el trabajo de titulación "ANÁLISIS COMPARATIVO DE LOS FRAMEWORKS JAVASCRIPT REACT Y VUE.JS MEDIANTE LAS NORMAS ISO/IEC 25010, PARA EL DESARROLLO DEL SISTEMA WEB DE CONTROL DE VENTAS PARA LA PANIFICADORA ROYAL" previó a la obtención del título de Ingeniero en Sistemas Computacionales ha sido desarrollado y terminado en su totalidad por el Sr. Paucar Mayanquer Javier Enrique, con cédula de identidad número 0401544085, bajo mi supervisión.

Es todo cuanto puedo certificar en honor a la verdad.

Atentamente,

.....

MSc. Daisy Elizabeth Imbaquingo Esparza

C.I:1002873048

DEDICATORIA

Dedico este trabajo a mis padres por su apoyo incondicional y los sacrificios que hicieron para permitirme alcanzar este objetivo.

A Martha por haberme alentado durante todo el trayecto del desarrollo de este trabajo.

Javier Paucar.

AGRADECIMIENTOS

Agradezco a todas las personas que de una u otra forma fueron parte en el proceso de desarrollo de este trabajo.

Agradezco a los docentes de la carrera de Ingeniería en Sistemas Computacionales que me acompañaron y guiaron en el desarrollo de este trabajo. Gracias por su paciencia, su apoyo, sus concejos y enseñanzas. En especial a mi directora, MSc. Daisy Imbaquingo, sin su ayuda no hubiera podido cumplir este reto académico. Aprendí mucho de ustedes y valoro su dedicación y compromiso.

Javier Paucar.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Resumen

En los últimos años con la popularización de JavaScript han aparecido varios frameworks para el desarrollo de interfaces gráficas, estas herramientas permiten a los desarrolladores usar buenas prácticas y ser más eficientes. Sin embargo, debido a la gran cantidad de opciones existentes resulta un desafío seleccionar una. El objetivo de este trabajo es realizar el análisis comparativo entre dos frameworks de JavaScript para el desarrollo de interfaces gráficas para aplicaciones web: React y Vue.js. Para aplicar el análisis comparativo se empleó como instrumento una matriz de calidad basada en el modelo de calidad de producto expuesto en la norma ISO/IEC 25010, está matriz fue aplicada a cada uno de los frameworks en análisis donde se calificaron las diferentes características de calidad descritas en el modelo de calidad. Una vez obtenidos los resultados del análisis comparativo se determinó que el framework que obtuvo las mejores puntuaciones fue Vue.js. Con estos resultados se usó Vue.js para desarrollar el sistema de control de ventas para la panificadora Royal. El objetivo del sistema desarrollado es permitir a la panificadora Royal mejorar el control interno de sus ventas e inventario. Este sistema se los desarrollo usando la metodología XP (eXtreme Programming), el patrón de diseño MVC (Modelo Vista Controlador) y siguiendo la arquitectura Cliente - Servidor.

Abstract

In recent years, with the popularization of JavaScript, several frameworks for the development of graphical interfaces have appeared. These tools allow developers to use best practices and be more efficient. However, due to the large number of options available, it is challenging to select one. The objective of this work is to carry out the comparative analysis between two JavaScript frameworks for the development of graphical interfaces for web applications: React and Vue.js. To apply the comparative analysis, a quality matrix based on the product quality model exposed in the ISO/IEC 25010 standard was used as an instrument. This matrix was applied to each of the frameworks under analysis where the different quality characteristics described in the quality model were qualified. Once the results of the comparative analysis have been obtained it was determined that the framework that obtained the best scores was Vue.js. With these results, Vue.js was used to develop the sales control system for the Royal bakery. The objective of the developed system is to allow the Royal bakery to improve the internal control of its sales and inventory. This system was developed using the XP (eXtreme Programming) methodology, the MVC (Model View Controller) design pattern and following the Client - Server architecture.

Introducción

Antecedentes

En los inicios al crear aplicaciones web se tenía en mente que toda la interacción del usuario con la aplicación recaiga sobre el servidor, en el transcurso de los años el hardware ha mejorado y los equipos que ejecutan donde se ejecutan los clientes son cada vez más potentes, por lo tanto, ya no es necesario que todo el trabajo de procesamiento lo haga el servidor. De esta forma la tendencia cambia a trasladar más carga de trabajo al navegador web lo que reduce las interacciones con el servidor y suprimen el retardo en la red. encargados de mostrar la parte gráfica o donde el usuario final interactua con el sistema, para lograr una buena experiencia en esta interacción los navegadores web hacen uso de JavaScript lenguaje que permite agregar dinamismo a las interfaces de usuario permitiendo brindar retroamimentación instantanea al usuario mientras navega por la interfaz del sistema. (MDN Web Docs, 2022c). En años reicentes han ido apareciendo en el mercado varias herramientas que permiten el desarrollo rapido de interfaces de usuario para aplicaciones web usando JavaScript dos ejemplos son React y Vue.js. React es una biblioteca de JavaScript para construir interfaces de usuario. (React, 2022). Vue.js es un framework accesible eficaz y versátil para creara interfaces web de usuario. (Vue.js, 2022).

Sistema

La Panificadora Royal no cuenta con un sistema que le permita registrar las ventas que se realizan en sus locales de una forma precisa y sistematizada.

Situación actual

Actualmente JavaScript se ha convertido en el lenguaje de programación más popular en el mundo, ha logrado esta posición porque es fácil de usar y muy poderoso en sus capacidades. No siempre estuvo en esta posición, pero hoy en día permite realizar virtualmente cualquier tipo aplicación. Al integrarlo con HTML5 y CSS3, permite crear poderosos sitios web, con diseños complejos y animaciones sin hacer uso de otras tecnologías como por ejemplo Adobe Flash. (Etheredge, 2017).

Por la popularidad de JavaScript en estos momentos variedad de tecnologías de desarrollo para este lenguaje son muy variadas cada una con sus ventajas y desventajas, además estas tecnologías cambian con mucha frecuencia, lo que hace difícil la elección de una alternativa adecuada que es crucial cuando se tiene que implementar en un proyecto.

Sistema

Actualmente la Panificadora Royal no dispone de un software que permita controlar las ventas ni el inventario, todos los registros se los realiza de forma manual dando como resultado el desconocimiento de la situación real del negocio en cuanto a sus ventas.

Prospectiva

Con la realización de este estudio comparativo se pretende tener un mejor conocimiento de las dos herramientas a estudiar y dejar un documento con los resultados obtenidos que sirva como referencia para las personas que estén interesadas en este tipo de tecnologías.

Sistema

Con el aplicativo que se va a desarrollar se espera mejorar el control de ventas que realiza la Panificadora Royal, permitiéndole tener datos precisos además de facilitar el registro de información evitando o minimizando el registro manual.

Planteamiento del problema

¿Qué framework JavaScript posee las mejores características para el desarrollo de la aplicación web sistema de control de ventas? (Fig. 1)

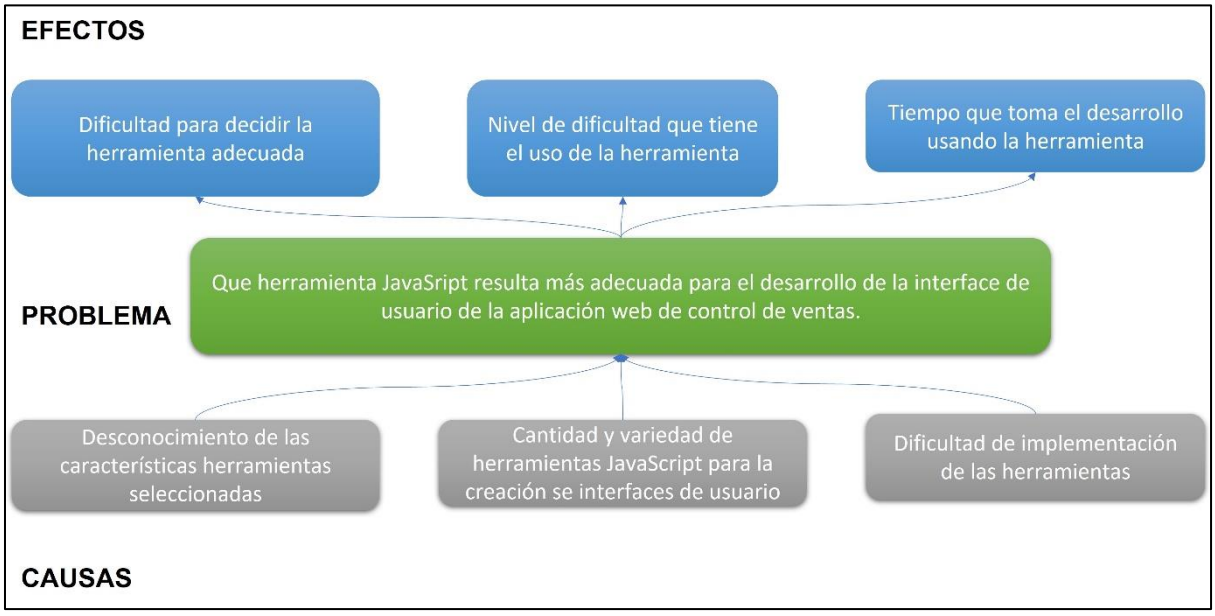


Fig. 1. Árbol de problema.

Objetivos

Objetivo general

Realizar el análisis comparativo de los frameworks JavaScript React y Vue.js mediante las normas ISO/IEC 25010, para el desarrollo del sistema web de control de ventas para la Panificadora Royal.

Objetivos específicos

- Determinar las características y funcionalidades principales de los dos frameworks.
- Realizar el análisis comparativo entre los frameworks React y Vue.js.
- Determinar cuál de las herramientas se adapta mejor para el desarrollo del sistema.
- Desarrollar el sistema utilizando el framework seleccionado utilizando la metodología XP.

Alcance

El presente estudio comparativo tiene como propósito analizar las características y funcionalidades de los frameworks React y Vue.js. Posteriormente realizar el sistema web de control de ventas para la Panificadora Royal seleccionando uno de los frameworks.

Sistema

El aplicativo para controlar las ventas en los locales de la Panificadora Royal hará que sea más fácil controlar cuantas ventas se realizan en cada uno de sus locales permitiéndole tener información que ayudará a tener mejor visión de cuantos ingresos generan estos locales.

En Fig. 2 se indica la arquitectura del sistema para control de ventas:

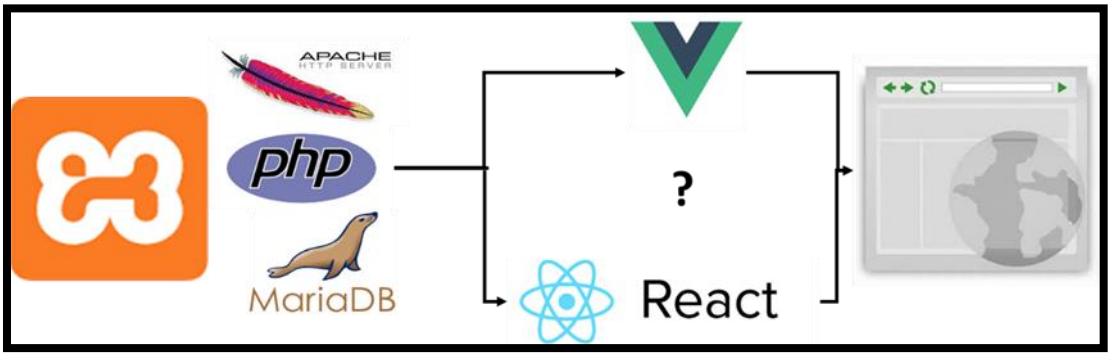


Fig. 2. Arquitectura de la aplicación.

La aplicación consta de los siguientes módulos:

- Registro de los locales que maneja la panificadora.
- Registro de usuarios del sistema.
- Manejo de inventario de cada local.
- Registro de ventas realizadas en cada local.
- Módulo de seguimiento técnico para el registro de ventas.
- Reportes de las ventas realizadas.

Justificación

Uno de los retos de los desarrolladores web es la elección de las tecnologías adecuadas para empezar con un proyecto, actualmente para el desarrollo de interfaces de usuario para sistemas web el lenguaje por excelencia es JavaScript, pero elegir entre la variedad de opciones que existe en cuanto a herramientas tales como librerías o frameworks es una ardua tarea.

Aún en la Universidad Técnica del Norte no se ha hecho un estudio sobre las herramientas propuestas, React y Vue.js, por lo que será un trabajo que quedará como fuente de consulta para la comunidad universitaria.

Sistema

Urdanegui (2019) indica que: "…Los retos más importantes que tienen las empresas son resguardar sus recursos y por medio de medidas de control evitar pérdidas…" (p. 13). Para hacer frente a estos retos se debe aplicar un control interno mediante mecanismos para prevenir la perdida de activos, incumplimiento de normas entre otros. (Urdanegui, 2019). Por esta razón la Panificadora Royal requiere llevar un control de las ventas para poder obtener información que le permita saber en qué situación se encuentra la empresa además de ayudarle a controla sus recursos, para cubrir este requerimiento se propone el desarrollo del sistema de control de ventas.

Capítulo I

1.1 Benchmarking

Según las definiciones de la TABLA 1.1 se puede definir al benchmarking como una serie de actividades realizadas con el objetivo de comparar procesos, entidades, sistemas, herramientas de desarrollo de software, entre otras. El objetivo de estas actividades es medir las características más relevantes de lo comparado y de esta manera formar un criterio que ayude a la toma de decisiones.

TABLA 1.1 Definiciones de benchmarking según varios autores.

Autor	Conceptos
(Spendolini, 1994)	Propone la siguiente definición: “Un proceso sistemático y continúo para evaluar los productos, servicios y procesos de trabajo de las organizaciones que son reconocidas como representantes de las mejores prácticas, con el propósito de realizar mejoras organizacionales”.
(Bouckaert et al., 2011)	Define al benchmarking en relación con sistemas computacionales como: “el acto de medir y evaluar el rendimiento computacional, dispositivos y redes bajo condiciones referenciales, relativas a una evaluación de referencia”.
(ISO/IEC, 2017)	Según esta organización el benchmarking consiste en: “comparar objetos de interés entre ellos, o usando otra comparativa como referencia, para evaluar sus características”.
(ISO/IEC/IEEE, 2010)	Define como: “un procedimiento, problema o prueba que puede ser usada para comparar sistemas o componentes entre ellos o usando un estándar”.
(Standard Performance Evaluation Corporation (SPEC), 2013)	Benchmarking es: “una prueba o conjunto de pruebas diseñadas para comparar el rendimiento de un sistema contra el rendimiento de otros sistemas”.

1.1.1 Benchmarking en el software.

El benchmarking es un proceso continuo que puede ser aplicado a cualquier área para comparar rendimiento o para aprender de otros. (Stapenhurst, 2009). El objetivo del benchmarking es facilitar una comparación justa entre dos diferentes soluciones haciendo uso de herramientas como pueden ser una aplicación o set de aplicaciones que ayuden a medir el rendimiento o cualquier otro aspecto de las soluciones a comparar. (Bouckaert et al., 2011). Si se refiere a la informática pueden poner bajo prueba elementos de hardware, una plataforma de software, un componente de software e incluso una operación. El benchmarking puede ser aplicado a cualquier plataforma o sistema sin importar su tamaño o complejidad. (Waller, 2014). El uso del benchmarking permite a la organización tomar en cuenta diferentes opciones, y no anclarse a una sola.

Para la realización de un benchmark se toman en cuenta diversos puntos de vista según los objetivos de la organización, por ejemplo, para quienes desarrollan el software su principal objetivo será enfocarse en como optimizar el rendimiento y procesos del software, mientras que para un usuario será seleccionar que opción le sirve mejor para un determinado proyecto. Daneva (1995) propone los siguientes criterios para realizar benchmarking a productos de software:

- Relevancia: debe ser relacionado estrechamente con el problema de negocio del cliente.
- Cobertura funcional: se deben seleccionar las funcionalidades que el usuario espera usar más en el sistema.
- Entendible: si es complejo de entender, dará poca credibilidad.
- Fácil de usar: el procedimiento de benchmarking debe ser relativamente simple, los costos no deben ser elevados incluyendo tiempo de diseño, recursos de hardware.
- Interpretación: el benchmark del producto debe proveer reglas de decisión para identificar el mejor producto de software de entre los competidores.
- Reproducibilidad: Cada vez que el benchmark es puesto en marcha los resultados deben permanecer inalterables.

1.2 Frameworks

De las definiciones de la Tabla 1.2 se puede definir un framework como un conjunto de elementos estructurales que proporcionan una plantilla de proyecto que ayuda a los desarrolladores a poder construir las aplicaciones de manera eficiente.

Tabla 1.2 Definiciones de framework según varios autores.

Autor	Conceptos
(Phillips, 2006)	Los define como: “Los frameworks pueden ser estructuras completas de directorios o un conjunto de funciones que ayudan a los desarrolladores a construir aplicaciones de forma rápida eficiente y organizada permitiendo generar productos de software de alta calidad”.
(Khan, 2021)	Lo define como: “Una colección de librerías que proveen un conjunto de funciones que cubren varias tareas rutinarias y proveen reglas, guías y patrones que permiten la construcción de aplicaciones de forma rápida, eficiente y organizada”.
(Olawanle Joel, 2022)	“El termino framework se puede referir a una estructura. Esta puede ser la estructura de un sistema, un edificio, un proyecto o cualquier otra cosa”.
(Cavaness, 2004)	“Un framework es un set de clases e interfaces que cooperan para resolver un tipo específico de problema de software”.
(Codecademy Team, 2021)	“Un framework es una estructura sobre la cual se puede construir software. Esta sirve como base, de esta manera no se empieza desde cero”.

1.3 JavaScript

JavaScript fue desarrollado por Netscape en colaboración con Sun Microsystems. Antes de JavaScript los navegadores web era una pieza muy básica de software con la capacidad de mostrar documentos de hipertexto. La primera versión de JavaScript 1.0 debuto en Netscape Navigator 2 en 1995. (Keith & Jeffrey Sambells, 2005).

JavaScript es un lenguaje ligero, interpretado y orientado a objetos con funciones de primera clase. Es un lenguaje de scripts dinámico, multiparadigma, basado en prototipos que

admite estilos de programación orientado a objetos, imperativos y funcionales. (MDN Web Docs, 2022a).

JavaScript es principalmente usado en aplicaciones web, pero no se limita a entornos web, en los últimos años los usos del lenguaje se han extendido a entornos como servidores, dispositivos móviles y sistemas embebidos, por esta versatilidad se ha convertido en uno de los lenguajes más usados a nivel mundial. (Meltzer, 2020).

1.4 React

React es una librería de JavaScript declarativa, eficiente y flexible para el desarrollo de interfaces de usuario. Permite construir interfaces de usuario complejas desde pequeñas y aisladas piezas de código llamadas “componentes”. Los componentes se usan para decirle a React que es lo que debe mostrar en pantalla. Cuando los datos cambian, vuelve a renderizar los componentes a los cuales están enlazados esos datos. (Facebook Inc, 2022).

1.4.1 DOM [1] Virtual (VDOM)

“El DOM es una API definida para representar e interactuar con cualquier documento HTML o XML. El DOM es un modelo de documento que se carga en el navegador web y representa el documento como un árbol de nodos, donde cada nodo representa una parte del documento” (MDN Web Docs, 2022b).

“El DOM virtual (VDOM) es un concepto de programación donde una representación virtual de la interfaz de usuario (UI) se mantiene en memoria y en sincronía con el DOM real, mediante una biblioteca. Este proceso se conoce como reconciliación” (React, 2022).

La reconciliación consiste en un algoritmo que diferencia un árbol DOM de otro para determinar que partes necesitan cambiar, su funcionamiento se muestra en la Fig. 3.(Clark, 2016).

[1] DOM: Document Object Model

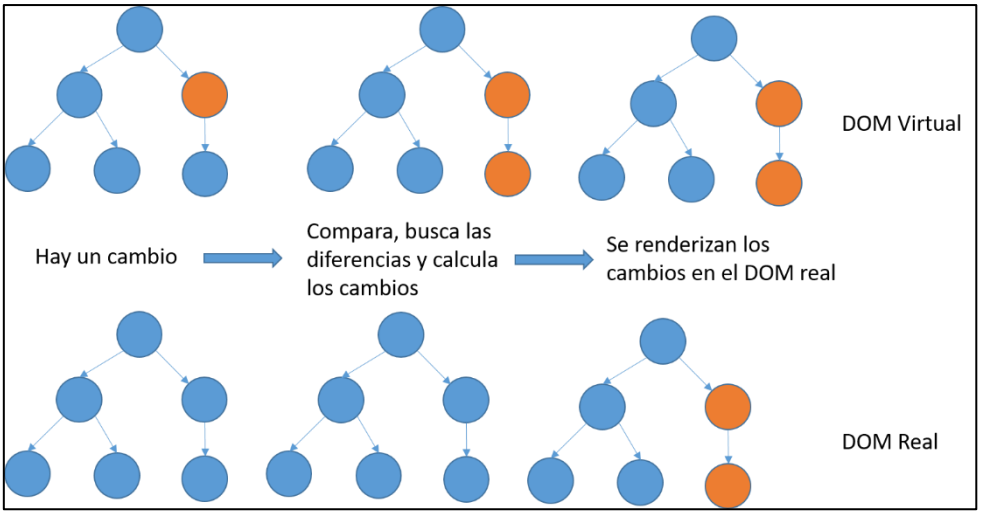


Fig. 3. Representación del funcionamiento del algoritmo de reconciliación.

Fuente: (Hamedani, 2018)

1.4.2 JSX

JSX es una sintaxis de extensión para JavaScript parecida a XML sin ninguna semántica definida. Está pensada para ser transformada en JavaScript valido mediante preprocesadores. (TypeScript, 2022).

React usa expresiones JSX para describir la interfaz de usuario que devuelven los componentes, JSX produce “elementos” de React haciendo uso del compilador Babel. React no requiere JSX, pero recomienda su uso como ayuda visual para trabajar con la interfaz de usuario. (React, 2022).

Una comparación de una expresión JSX y como babel compila esta expresión se muestra en la Fig. 4.

```
//JSX
const element = (
    <h1 className="greeting">
        Hello, world!
    </h1>
);

// llamada de React.createElement()
const element_ = React.createElement(
    'h1',
    { className: 'greeting' },
    'Hello, world!'
);
```

Fig. 4. Expresión JSX y su equivalente como llamada de React.createElement().

1.4.3 Componentes

Conceptualmente, los componentes son como funciones JavaScript. Aceptan entradas arbitrarias y retornan elementos de React que describen lo que debe aparecer en la pantalla. Los componentes permiten separar la interfaz de usuario en piezas independientes, reutilizables. (React, 2022).

Pavlutin (2017) describe algunas de las características de los componentes de React:

- Los componentes están encapsulados, el comportamiento del componente se controla mediante propiedades dejando la estructura interna del componente oculta.
- Los componentes cumplen con el principio de responsabilidad única, cada componente es responsable de una parte especifica de la interfaz de usuario.
- Son componibles, esto quiere decir que estos componentes pueden interactuar entre sí para formar estructuras más complejas, como si se tratara de piezas de lego.
- Son reusables, cada componente al ser componible se lo puede usar repetidamente en diferentes lugares de la interfaz de usuario.

En la siguiente imagen (Fig. 5) se muestra una interfaz gráfica compuesta de componentes y una representación de cómo los componentes se relacionan entre sí.

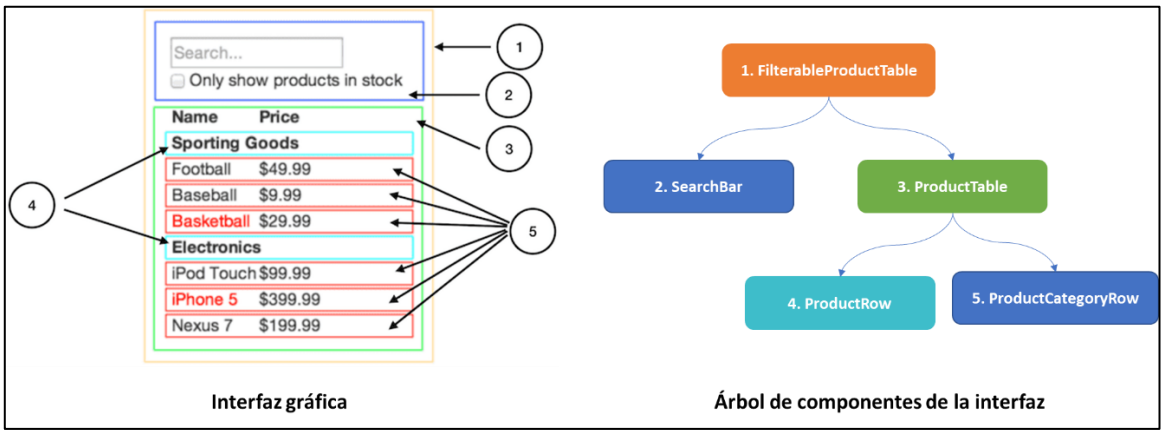


Fig. 5. Interfaz gráfica formada de componentes.

Fuente: https://reactjs.org/static/9381f09e609723a8bb6e4ba1a7713b46/90cbd/thinking-in-react-components.png

1.4.4 Renderizado

El renderizado de los componentes de una interfaz gráfica en React se compone de los siguientes tres pasos (Meta Open Source, 2023):

- **Triggering**: Lanzar la petición de la orden de renderizado.

  El lanzamiento de la petición de renderizado puede ocurrir por dos razones, una es el renderizado inicial del componente, otra es cuando el estado de un componente ha cambiado.

- **Rendering**: Preparar el componente que se va a renderizar.

  Luego de lanzar una petición de renderizado React llama a los componentes que se van a renderizar para saber qué es lo que se va a mostrar en pantalla. En el render inicial React llama al componente raíz, para los siguientes renders React llama a la función del componente cuyo estado ha cambiado.

- **Committing**: Colocar el componente en el DOM para mostrarlo.

  Luego de que React llama a los componentes modifica el DOM, para el render inicial React usa la función appendChild() para poner todos los nodos que ha creado en la pantalla. Para los componentes que vuelven a renderizarse React aplica las operaciones necesarias en el DOM, no actualiza todo el DOM solamente las partes que han cambiado.

1.4.5 Ciclo de vida de los componentes de React

Los componentes tienen diferentes fases de ejecución; el montaje, la actualización y el desmontaje, a estas fases se les llama el ciclo de vida del componente. (React, 2022). Cada de una de estas fases tiene distintos métodos que se ejecutan en determinado momento y que pueden llegar a ser útiles para ejecutar ciertos procesos como por ejemplo obtener datos de un servicio al renderizar por primera vez un componente. En la Fig. 6 se muestra el diagrama del flujo principal del ciclo de vida de un componente.

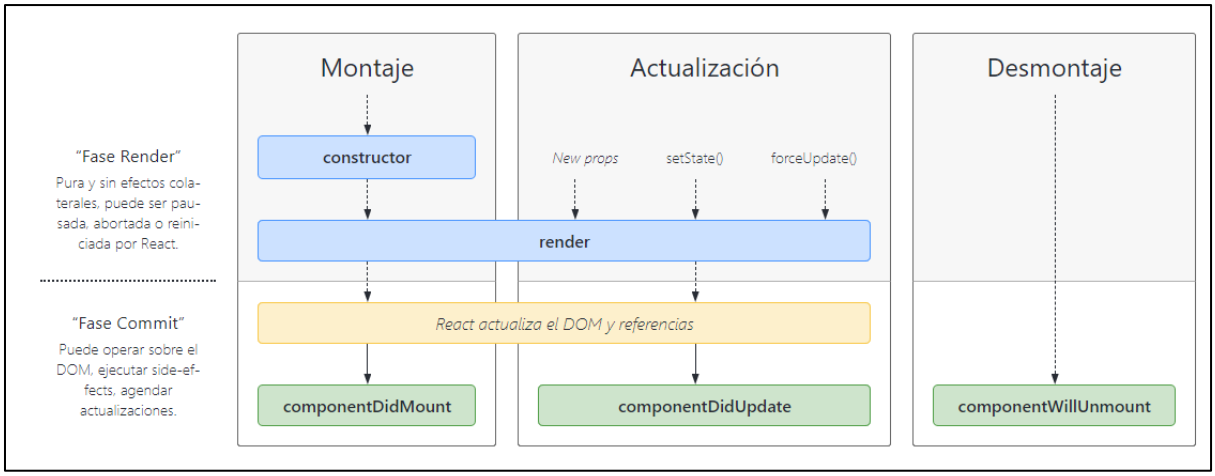


Fig. 6. Diagrama del ciclo de vida de un componente de React.
Fuente: https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

1.4.6 One Way Data Binding [2]

Data binding (enlace de datos) es una técnica que permite crear un enlace para sincronizar datos entre un proveedor y un consumidor de datos.(S. Gillis, 2022).

React hace uso de one way data binding, por lo que los datos fluyen en una sola dirección, desde los componentes padre hacia el componente hijo. El componente hijo puede leer los datos, pero no puede actualizar los datos directamente. En su lugar el componente hijo emite eventos hacia el padre y este es el responsable de actualizar los datos, como se representa en Fig. 7.(Roth, 2020).

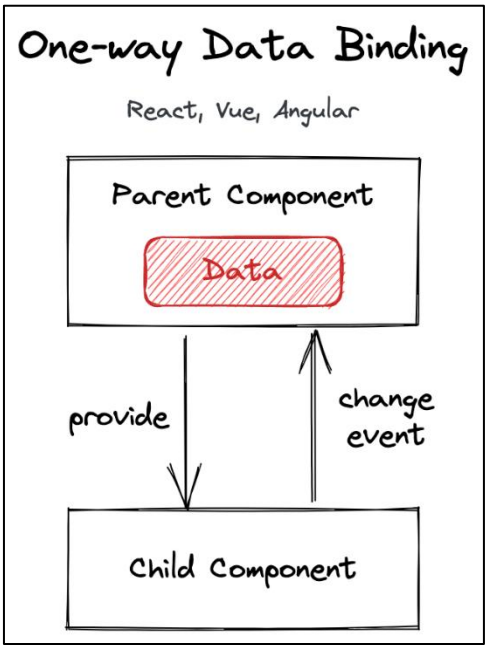


Fig. 7. Comunicación entre componentes usando One Way Data Binding.

[2] One Way Data Binding: Enlace de datos en una dirección

Fuente: https://sandroroth.com/assets/blog/react-two-way-data-binding/data-binding.png

1.4.7 Hooks

Los Hooks son funciones de javascript que permiten, crear y acceder al estado y métodos del ciclo de vida de los componentes, de esta manera con los Hooks es posible crear componentes funcionales que manejen estado, son una alternativa a los componentes de clase.(Huet, 2020).

Las principales ventajas de los Hooks según React (2022) son:

- Permiten reusar la lógica con su estado sin cambiar la jerarquía de los componentes.
- Permiten separar un componente es piezas más pequeñas basadas en la funcionalidad relacionada de cada pieza.
- Permiten usar la mayoría de las características de React si usar clases.

1.5 Vue.js

Vue es un framework de JavaScript que permite construir interfaces de usuario. Está construido sobre HTML, CSS y JavaScript y provee un modelo de programación declarativo basado en componentes que ayuda a construir interfaces de usuario de forma eficiente.(Vue.js, 2022).

1.5.1 Componentes

Vue al igual que React usa el concepto de componentes que permiten dividir la interfaz gráfica en piezas independientes y reusables. A diferencia de React en el que los componentes son funciones o clases, Vue hace uso de Single-File Components (SFC) que son un tipo especial de archivo que permite encapsular la lógica, vista y estilo en un solo archivo. Los bloques <template>, <script> y <style> del archivo encapsulan las diferentes partes del componente como se muestra en Fig. 8.(Vue.js, 2022).

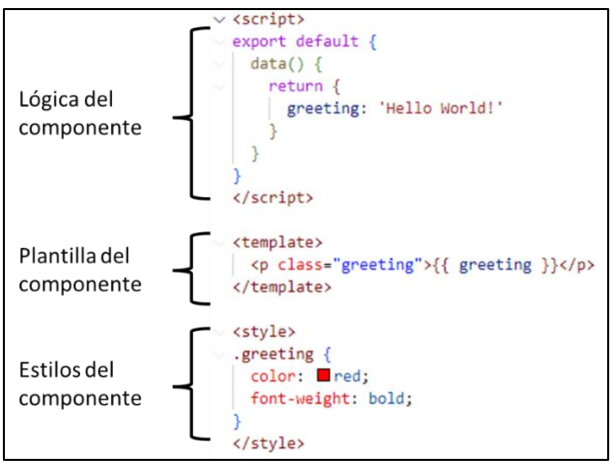


Fig. 8. Estructura de un Single-File Component de Vue.

En Vue cada instancia de un componente recorre una serie de pasos cuando es creado, montar la aplicación, compilar la plantilla, comprobara cambios en el estado de componente, etc. (Vue.js, 2022). Esta serie de pasos componen el ciclo de vida de un componente, en la siguiente imagen se muestra un diagrama de referencia.

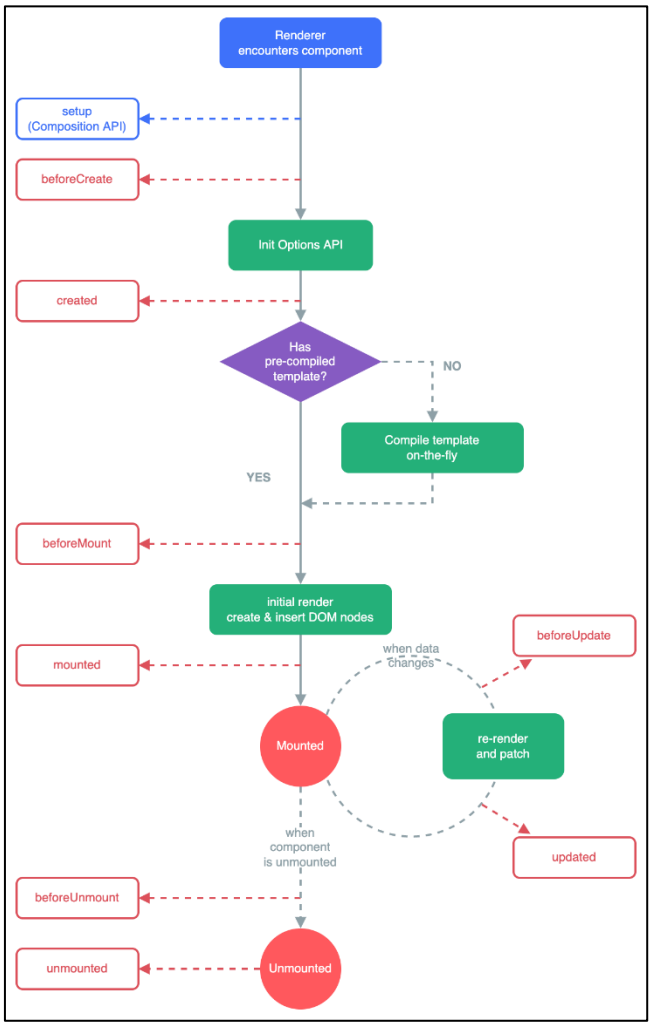


Fig. 9. Diagrama del ciclo de vida de un componente de Vue.
Fuente: https://vuejs.org/assets/lifecycle.16e4c08e.png

1.5.2 Renderizado

Vue usa una sintaxis de plantilla basada en HTML esto permite enlazar de forma declarativa el DOM a la instancia subyacente del componente. Todas las plantillas de Vue son HTML válido que puede ser procesado por navegadores.(Vue.js, 2022).

Cuando Vue compila las plantillas retorna funciones de renderizado, estas funciones retornan la representación de la plantilla como un nodo de DOM Virtual que luego Vue renderiza en el navegador. Vue también permite hacer uso directamente de las funciones de

renderizado sin hacer uso de plantillas, estas funciones también soportan JSX.(Ouellet, 2022).

El proceso de renderizado de Vue consta de las siguientes etapas (Fig. 10):

- Compilar: las plantillas de Vue son compiladas en render functions: estas funciones retornan arboles de DOM Virtual.
- Montado: El renderizador en tiempo de ejecución invoca las funciones de renderizado, recorre los árboles de DOM Virtual retornados por estas funciones y crea nodos en el DOM real basados en el DOM Virtual.
- Parchar: Cada vez que una dependencia usada cambia durante el montaje se repite el proceso de renderizado, creando un nuevo DOM Virtual.

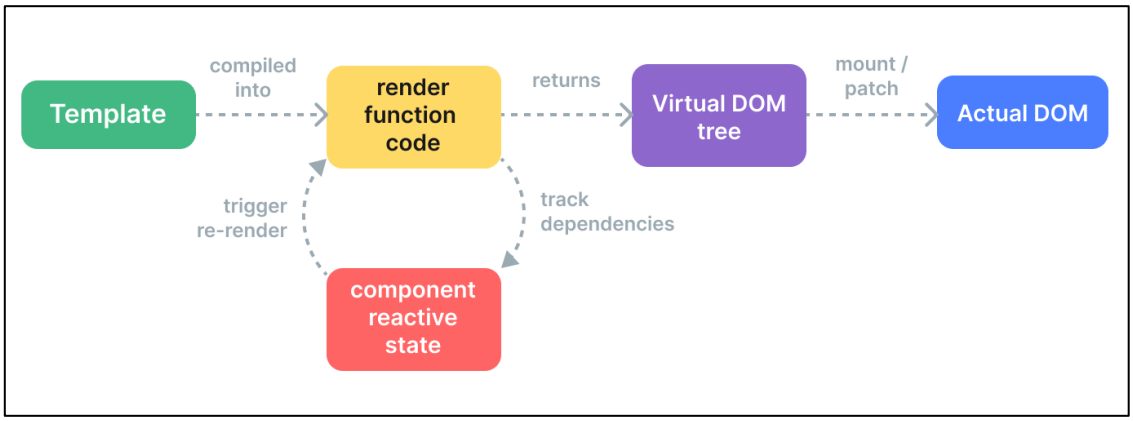


Fig. 10. Proceso de renderizado Vue.

Fuente: https://vuejs.org/assets/render-pipeline.03805016.png

1.5.3 Composition API

Composition API es un set de APIs que permiten crear componentes de Vue usando funciones importadas en lugar de usar las opciones del componente, este set de APIs encapsula (Vue.js, 2022):

- **Reactivity API**: permite crear directamente estados reactivos, estados computados y watchers.
- **Lifecycle Hooks**: permite acoplarse de forma programática en las diferentes fases de ciclo de vida del componente con funciones como onMounted(), onUnmounted, etc.
- **Dependency Injection**: mediante las funciones provide() y inject(), permiten aprovechar el sistema de inyección de dependencias mientras se hace uso de Reactivity API.

Las principales ventajas de Composition API según Vue.js (2022) son:

- Permite una mejorar la capacidad de reusabilidad de la lógica de un componente de forma clara y eficiente en la forma de funciones componibles.
- Más flexibilidad y mejor organización del código, permite a agrupar el código de un componente de manera que se puede separar la lógica relacionada en diferentes funciones.

1.5.4 Usar Vue sin herramientas de compilación.

Vue permite usar sus características para crear interfaces basadas en componentes sin necesidad de usar herramientas de compilación como Webpack[3]. Basta con insertar la librería en la página HTML usando la etiqueta <script> e indicar a Vue donde montar la aplicación. React también permite hacer está integración, pero para hacer uso de JSX es necesario tener instalado Node.js[4] y compilar el código de un pre-procesador JSX.

1.6 ISO/IEC 25000

La ISO/IEC 25000 es un conjunto de normas conocidas como Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE[5]). La base de estas normas son los estándares ISO/IEC 9126 e ISO/IEC 14598, que definen características, métricas y métodos de evaluación con el objetivo de crear un marco de trabajo que permite evaluar la calidad de productos de software.(INEN, 2015).

La familia de normas SQuaRE está compuesta por cinco divisiones como se ve en la Fig. 11.

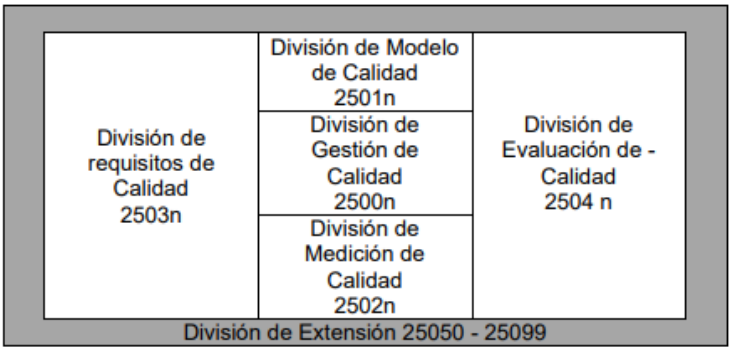


Fig. 11. Organización de las series de Normas Internacionales SQuaRE.
Fuente: (INEN, 2015).

La TABLA 1.3 describe en que consiste cada una de las divisiones de las que se componen las series SQuaRE.

[3] Webpack: https://webpack.js.org/
[4] Node.js: https://nodejs.org/es/
[5] SQuaRE: Systems and software Quality Requirements and Evaluation

TABLA 1.3 Descripción de las divisiones de las series SQUARE

División	Descripción
ISO/IEC 2500n: División de Gestión de Calidad	Las normas de esta división definen todos los modelos, términos y definiciones comunes referidos por otras normas de las series SQuaRE.
ISO/IEC 2501n: División de Modelo de Calidad	Las normas de esta división presentan modelos de calidad detallados para productos de software, calidad de uso y datos, en esta división se encuentra el estándar ISO/IEC 25010.
ISO/IEC 2502n: División de Medición de Calidad	Las normas de esta división incluyen un modelo de referencia de medición de calidad del producto de software.
ISO/IEC 2503n: División de Requisitos de calidad	Las normas de esta división ayudan a especificar requisitos de calidad, en base a modelos de calidad y medidas de calidad.
ISO/IEC 2504n: División de Evaluación de Calidad	Las normas que forman esta división proveen requisitos, recomendaciones y directrices para evaluación del producto de software.

Fuente: (INEN, 2015)

1.7 ISO/IEC 25010

En 2011 un grupo de trabajo de la organización ISO lanzo un modelo estándar de calidad de productos de software: ISO/IEC 25010. Este estándar se convirtió en el modelo de calidad de software más conocido. (Wagner, 2013).

El principal concepto de este modelo es separar la complejidad de un producto de software en piezas pequeñas y manejables. La idea es que la descomposición alcance un nivel con el cual se puedan medir las partes y usar estas medidas para evaluar la calidad del producto de software. (Wagner, 2013).

Esta norma define dos modelos de calidad según el (INEN, 2015): Un modelo de calidad de uso y un modelo de calidad de producto.

1.7.1 Calidad del software

De acuerdo con las definiciones expuestas en la Tabla 1.4 se puede decir que la calidad del software es asegurar que un producto, sistema o componente cumpla con los requerimientos establecidos y además cubra las necesidades y expectativas de los usuarios, siguiendo procesos y estándares para el desarrollo del software.

Tabla 1.4 Definiciones de calidad de software según varios autores.

Autor	Conceptos
(INEN, 2015)	Define la calidad del software como: “grado en el cual un producto de software satisface las necesidades establecidas e implícitas al ser usado bajo condiciones específicas”.
(ISO/IEC/IEEE, 2010)	Define la calidad como: "la habilidad de un producto, servicio, sistema, componente o proceso para satisfacer a un consumidor o las necesidades, expectativas o requerimientos de los usuarios".
(Rose, 2013)	Define la calidad aseguramiento de software como: "un proceso continuo que audita a otros procesos de software para asegurar que estos procesos se sigan".
(IEEE, 1990)	Define la calidad del software como: "el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario".
(Pressman, 2010)	Define la calidad del software como: "Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente".

1.7.2 Modelo de calidad

La norma ISO/IEC 2510:2011 presenta una estructura (Fig. 12) para el modelo de calidad, esta estructura consiste en el desglose de la calidad en características y subcaracterísticas para luego obtener las propiedades de calidad que tienen que ser medibles y relacionadas con la calidad.

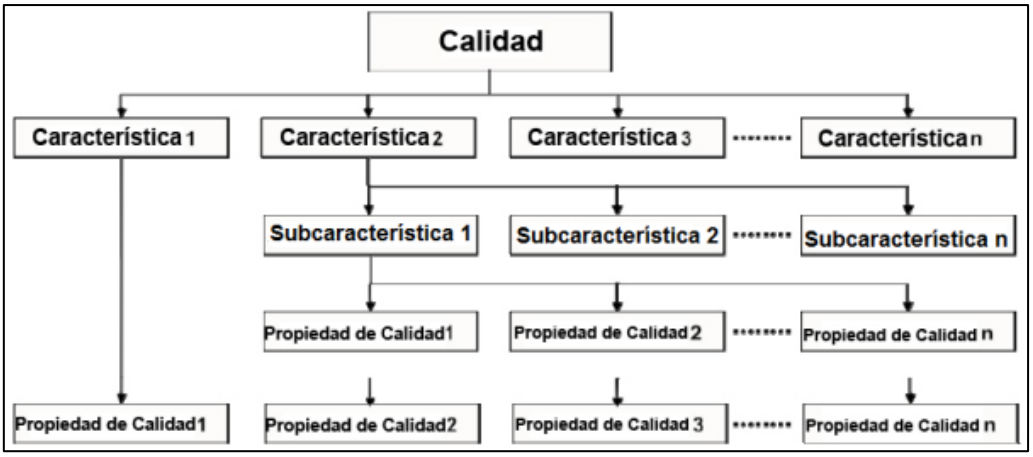


Fig. 12. Estructura usada para los modelos de calidad.

Fuente: (INEN, 2015).

1.7.3 Modelo de calidad de uso

INEN (2015) presenta cinco características relacionadas con los resultados de interacción con un sistema: eficacia, eficiencia, satisfacción, ausencia de riesgo y la cobertura de contexto (Fig. 13).

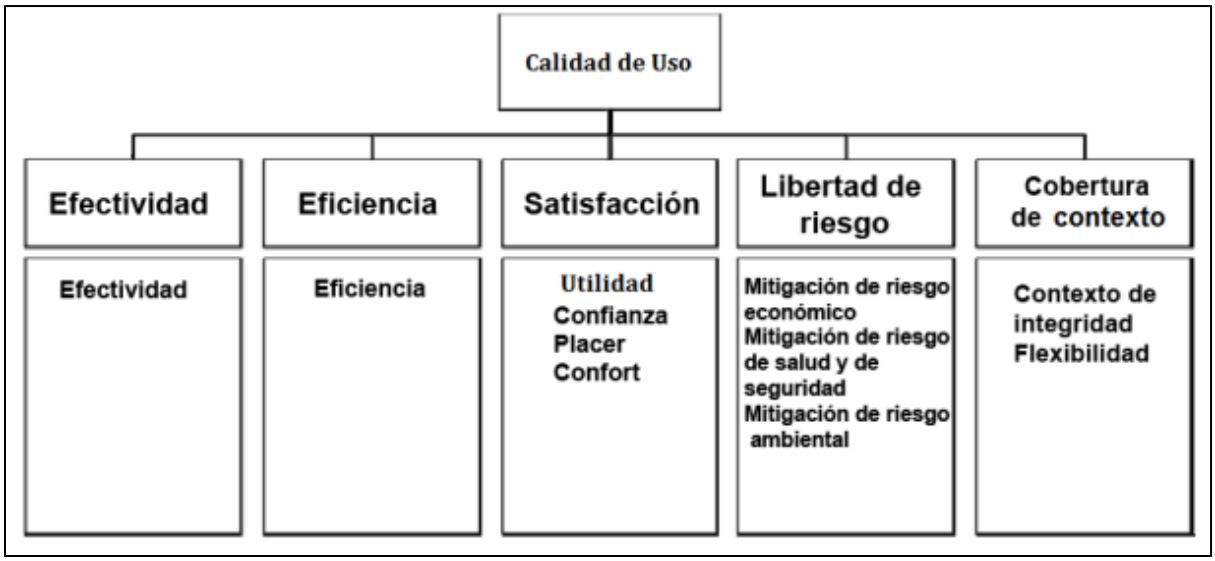


Fig. 13. Modelo de calidad de uso.

Fuente: (INEN, 2015).

Según INEN (2015) indica la siguiente lista de características y subcaracterísticas del modelo de calidad de uso:

a) **Efectividad. -** Precisión y cumplimiento con los cuales los usuarios alcanzan las metas específicas.

b) **Eficiencia. –** Recursos expandidos en relación con la precisión y el cumplimiento con los cuales los usuarios alcanzan las metas.

c) **Satisfacción. –** Grado en el cual las necesidades de los usuarios son satisfechas cuando el sistema es usado en un contexto específico.

- **Utilidad. –** Grado en el que el usuario está satisfecho con los resultados obtenidos por el producto.
- **Confianza. –** Grado en que las partes interesadas tienen confianza de que el producto funcionará como se espera.
- **Placer. –** Grado en el que los usuarios obtienen placer de satisfacer sus necesidades personales.
- **Confort. –** Grado en el que el usuario está satisfecho con el confort físico.

d) **Libertad de riesgo. –** Grado en el que el producto mitiga el riesgo potencial al estado económico, a la vida humana, la salud, o ambiente.

- **Mitigación de riesgo económico. –** Grado en el cual el producto mitiga el riesgo al estado financiero.
- **Salud y mitigación de riesgo de seguridad. –** Grado en el cual el producto mitiga el riesgo en los contextos de uso.
- **Mitigación de riesgo ambiental. –** Grado en el cual un producto mitiga el riesgo a la propiedad o al ambiente.

e) **Cobertura de contexto. –** Grado en el cual un producto puede ser usado para efectividad, eficiencia, libertad de riesgo y satisfacción tanto en contextos específicos y contextos más allá de los especificados.

- **Integridad de contexto. –** Grado en el que un producto puede ser usado en todos los contextos específicos de uso.
- **Flexibilidad. –** Grado en el cual un producto puede ser usado en contextos más allá que los especificados inicialmente en los requisitos. (págs. 9-11)

1.7.4 Modelo de calidad de producto

El modelo de calidad de producto específica y evalúa el cumplimiento de criterios del producto (Callejas Cuervo et al., 2017). Categoriza las propiedades de calidad de producto en varias características, cada característica compuesta de un conjunto de subcaracterísticas relacionadas. (INEN, 2015).

El modelo de calidad de producto de la ISO/IEC 25010 se encuentra compuesto por ocho características de calidad.(INEN, 2015). La intención del estándar es que las ocho

características describan la calidad de un producto de software. (Wagner, 2013). En Fig. 14 se muestra un diagrama con las ocho características que componen el modelo.

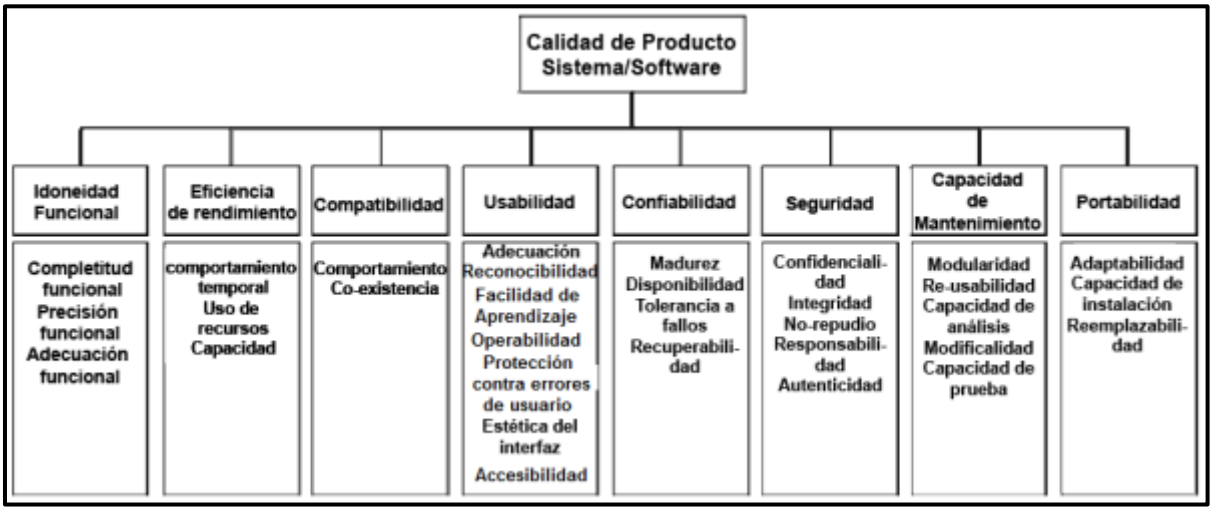


Fig. 14. Modelo de calidad de Producto.

Fuente: (INEN, 2015).

Según INEN (2015) presenta la siguiente lista de características y subcaracterísticas del modelo de calidad de producto:

a) **Idoneidad funcional. -** Grado en el que el software provee las funciones que satisfacen las necesidades cuando el software se usa en condiciones específicas.
   - **Completitud funcional. -** Grado en el cual el conjunto de funciones cubre todas las tareas y objetivos específicos del usuario.
   - **Exactitud funcional. -** Grado de exactitud en el cual un producto o un sistema provee los resultados correctos con el grado necesario de precisión.
   - **Oportunidad funcional. -** Grado en el cual las funciones facilitan el cumplimiento de las tareas y objetivos específicos.

b) **Eficiencia de rendimiento. -** El rendimiento según la cantidad de recursos que un software utiliza expuesto bajo determinadas condiciones.
   - **Comportamiento en el tiempo. -** Grado en el cual la respuesta, los tiempos de procesamiento y las tasas de rendimiento de un producto o sistema satisfacen los requisitos al desempeñar sus funciones.
   - **Utilización de recursos. -** Grado en el cual las cantidades y tipos de recursos usados por un producto o sistema satisfacen los requisitos al desempeñar sus funciones.

- **Capacidad. -** Grado en el cual los límites máximos de un parámetro de un producto o sistema satisface los requisitos.

c) Compatibilidad. - Es la capacidad de varios sistemas o componentes para interactuar entre ellos y llevar a cabo sus funciones requeridas cuando comparten el mismo entorno.

- **Coexistencia. -** Grado en el cual un producto puede realizar sus funciones requeridas eficientemente mientras comparte un ambiente y recursos comunes con otros productos, sin un impacto perjudicial en ningún otro producto.
- **Interoperabilidad. -** Grado en el cual dos o más sistemas, productos o componentes pueden intercambiar información y usar la información que ha sido intercambiada.

d) Usabilidad. - Grado en el que un software puede ser usado por usuarios específicos para alcanzar sus objetivos en un contexto específico de uso.

- **Reconocimiento de usabilidad. -** Grado en el cual los usuarios pueden reconocer si un producto o sistema es apropiado para sus necesidades.
- **Capacidad de aprendizaje. -** Grado en el cual un producto o sistema puede ser usado por usuarios específicos para alcanzar las metas especificadas y así aprender a usar el producto o sistema con efectividad, eficiencia, libertad de riesgo y satisfacción en un contexto de uso específico.
- **Operabilidad. -** Grado en el cual un producto o sistema tiene atributos que lo hacen más fácil de operar y controlar.
- **Protección contra error en el uso. -** Grado en el cual un sistema protege a los usuarios de cometer errores.
- **Estética de la interfaz. -** Grado en el cual una interfaz del usuario permite una interacción agradable y satisfactoria para el usuario.
- **Accesibilidad. -** Grado en el cual un producto o sistema puede ser usado por las personas con el rango más amplio de características y capacidades para alcanzar una meta específica en un contexto de uso específico.

e) Confiabilidad. - Capacidad de un sistema o componente para desempeñar sus funciones bajo condiciones específicas y durante periodos determinados de tiempo.

- **Madurez. -** Grado en el cual un sistema, producto o componente satisface las necesidades de confiabilidad bajo operación normal.
- **Disponibilidad. -** Grado en el cual un sistema, producto o componente es operativo y accesible cuando se lo requiere para el uso.
- **Tolerancia del error. -** Grado en el cual un sistema, producto o componente opera como se pretende a pesar de la presencia de errores del hardware o del software.
- **Capacidad de recuperación. -** Grado en el cual, en el caso de una interrupción o falla, un producto o sistema puede recuperar los datos directamente afectados y re-establecer el estado deseado del sistema.

f) Seguridad. - Capacidad de un software para proteger la información y datos de personas o sistemas no autorizados de manera que estos no puedan leerlos ni modificarlos.

- **Confidencialidad. -** Grado en el cual un producto o sistema asegura que los datos son accesibles solo para aquellos autorizados a tener acceso.
- **Integridad. -** Grado en el cual el sistema, producto o componente previene acceso no autorizado a, o modificación de los programas de computación o de los datos.
- **No-repudio. -** Grado en el cual se puede probar que las acciones o hechos sucedieron, de tal manera que los eventos o acciones que no pueden ser repudiadas más tarde.
- **Responsabilidad. -** Grado en el que las acciones de una entidad pueden ser rastreados de forma exclusiva a la entidad.
- **Autenticidad. -** Grado en el que la identidad de un objeto o recurso se puede probar en caso de reclamo.

g) Capacidad de mantenimiento. - Característica que permite a un software ser modificado de forma efectiva y eficiente, debido a necesidades evolutivas, correctivas o perfectivas.

- **Modularidad. -** Grado en el cual un sistema o programa de computación está compuesto de componentes discretos de tal manera que un cambio a un componente tiene un impacto mínimo en otros componentes.

- **Reusabilidad. -** Grado en el cual un activo puede ser usado en más de un sistema, o ser armado en otros activos.
- **Capacidad de análisis. -** Grado de efectividad y eficiencia con el cual es posible evaluar el impacto en un producto o sistema por un cambio previsto sobre uno o más de sus partes, o diagnosticar un producto por las deficiencias o causas de falla, o identificar las partes a ser modificadas.
- **Capacidad en modificación. -** Grado en el cual un producto o sistema puede ser efectiva y eficientemente modificado sin introducir defectos o sin degradar la calidad de producto existente.
- **Capacidad de prueba. -** Grado de efectividad y eficiencia con el cual los criterios de prueba pueden ser establecidos para un sistema, producto o componente y las pruebas pueden ser realizadas para determinar si aquellos criterios han sido satisfechos.

h) Portabilidad. - Capacidad de un software de ser transportado de forma efectiva y eficiente de manera que pueda ser transferido de un entorno a otro.

- **Adaptabilidad. -** Grado en cual un producto o sistema puede efectiva y eficientemente ser adaptado en un hardware, software, u otros ambientes operativos o de uso.
- **Capacidad de instalación. -** Grado de efectividad y eficiencia con el cual un producto o sistema puede ser instalado exitosamente y desinstalado en un ambiente específico.
- **Capacidad de reemplazo. -** Grado en el cual un producto puede reemplazar otro producto de software especificado para el mismo propósito en el mismo ambiente. (págs. 11-17)

1.8 Metodología XP

XP o Programación Extrema es una metodología ágil para el desarrollo de software, consiste en una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad. (Letelier & Penadés, 2006).

1.8.1 Fases de la metodología XP

En la Fig. 15 se pueden ver las fases de las cuales está compuesta la metodología XP.

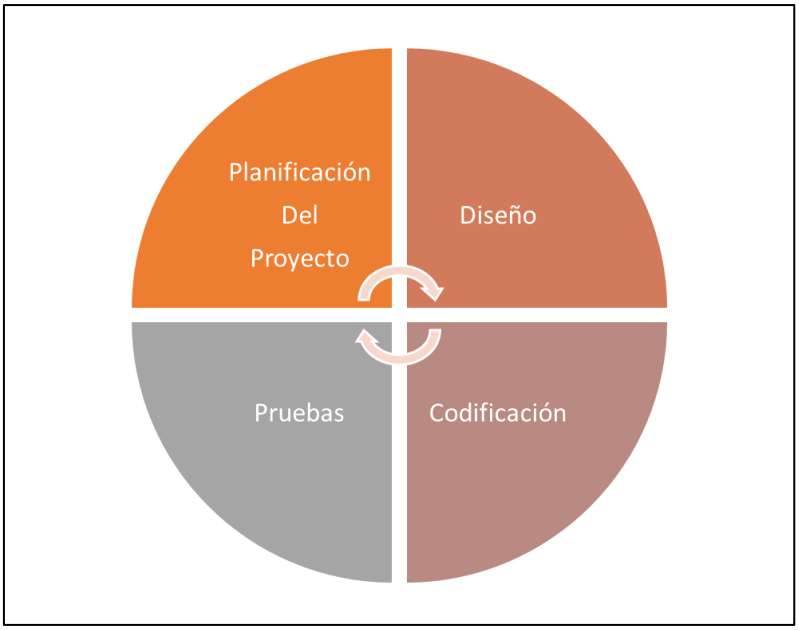


Fig. 15. Fases de la metodología XP.

Letelier & Penadés (2006) describen las siguientes fases:

a) Planificación del proyecto

En esa fase se hace una recopilación de todos los requerimientos del proyecto, también hay una interacción con el usuario, y se debe planificar entre los desarrolladores las necesidades del proyecto para lograr los objetivos finales.

b) Diseño

En esta fase debe predominar la simplicidad y sencillez se debe tomar solo entiempo necesario para el diseño de diagramas e interfaces de usuario, usar un glosario para nombres de métodos que facilite la comprensión y reutilización del código, no se debe añadir funcionalidades extra, mejorar la estructura y codificación de códigos ya creados.

c) Codificación

En esta ase de codificación los clientes y los desarrolladores del proyecto deben estar en comunicación para que los desarrolladores puedan codificar todo lo necesario para el proyecto, en esta fase está incluido todo lo referente a la programación.

d) Pruebas

Uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento de los códigos que se implementen.

Capítulo II

2.1 Requisitos de la evaluación

En esta sección se definen los requisitos de la evaluación.

2.1.1 Propósito de la evaluación

La evaluación tiene como propósito el medir la calidad de los productos de software React y Vue.js para determinar que producto es el más conveniente para desarrollar el sistema web de control de ventas para la Panificadora Royal.

2.1.2 Identificar el tipo de producto a evaluar

De a la Clasificación Central de Productos (CPC) versión 2.0 elaborada por INEC (2012), se presenta un catálogo de productos relacionado al sector Software, sintetizado en la TABLA 2.1. De esta tabla los productos que se van a evaluar caen en la categoría de software de aplicación, siendo herramientas de software de productividad.

TABLA 2.1 Tipos de producto de software

Productos	Tipos de productos
Página web	– Estática Animada Dinámica. – Portal Web Tienda Virtual o Comercio Electrónico. – Página Web con Gestor de Contenido. – Página Web 2.0.
Base de datos	– Base de datos jerárquica. – Base de red Base de datos transaccional. – Base de datos relacional Base de datos multidimensional. – Base de datos orientado a objetos Base de datos documental. – Base de datos deductiva.
Software de aplicación	– Software de Aplicación de productividad (editores de texto). – Software de Aplicación de entretenimiento (videojuegos) Software de Aplicación de negocios (ERP). – Software de Aplicación de educación (programas interactivos de aprendizaje). – Software de Aplicación de tecnología (aplicaciones de control de sistemas, aplicaciones médicas, etc.).

Fuente: (INEN, 2012)

2.1.3 Especificar el modelo de calidad

Para esta evaluación se usa el modelo de calidad de producto que proporcionan las normas ISO/IEC 25010.

2.2 Especificar la evaluación

En esta sección se describen las métricas y criterios de evaluación y calificación que se van a aplicar.

2.2.1 Selección de métricas

Para está evaluación se hace uso de las métricas presentadas en la norma ISO/IEC 25023:2016 para calidad interna y externa, se muestra en el Anexo A.

Para la métrica de rendimiento se hacen las mediciones con la herramienta Lighthouse que nos da una calificación en base al análisis y puntaje de varias métricas presentadas en la TABLA 2.2.

TABLA 2.2 Métricas de rendimiento Lighthouse

Métrica	Descripción	Peso de la métrica
First Contentful Paint	Marca el tiempo cuando el primer texto o imagen es pintada.	10%
Speed Index	Muestra que tan rápido los contenidos de una página son visiblemente poblados.	10%
Largest Contentful Paint	Marca el tiempo en el que un texto extenso o una imagen pesada son pintados.	25%
Time to Iteractive	Es la cantidad de tiempo que le toma a la página en ser totalmente interactiva.	10%
Total Blocking Time	Suma de todos los periodos entre FCP y Time to Interactive, cuando la duración de las tareas excede de los 50 ms.	30%
Cumulative Layout Shift	Mide el movimiento de los elementos visibles dentro de la ventana.	15%

Fuente: https://developer.chrome.com/docs/lighthouse/performance/

2.2.2 Niveles de calificación

En seguida se listan los diferentes criterios de calificación definidos para la evaluación:

a) Niveles de importancia

Los niveles de importancia representan la prioridad que se le debe dar a las características y subcaracterísticas en la evaluación.

TABLA 2.3 Niveles de importancia.

Nivel de importancia	Nomenclatura	Descripción
Alta	A	El nivel de importancia de la característica y subcaracterística obliga a realizar las mediciones.
Media	M	El nivel de importancia de la característica y subcaracterística indica que se sujeta a criterio del evaluador.
Baja	B	El nivel de importancia de la característica y subcaracterística indica que no es necesaria la medición.
No Aplica	NA	Significa de no se puede medir o aplicar.

Fuente: (Chávez, 2011)

b) Ponderación de características y subcaracterísticas de calidad

Con el fin de puntualizar resultados cuantitativos calculados jerárquicamente en la evaluación del producto, se asigna una ponderación a las características y subcaracterísticas de calidad de los modelos de calidad externa, interna y en uso. Las ponderaciones están sujetas al criterio del evaluador y al tipo de producto que se evalúa, la sumatoria deberá ser igual al 100%.(Vaca Sierra, 2017).

c) Definición de niveles de puntuación final de calidad interna, externa y en uso

La escala de medición que se presenta en la TABLA 2.4 se utiliza para analizar el resultado final de características de calidad interna, externa y en uso, valores que determinarán el nivel obtenido final y que se asignará al producto software después de su análisis.(Vaca Sierra, 2017).

TABLA 2.4 Niveles de puntuación final para calidad externa, interna y en uso.

Escala de medición	Nivel de puntuación	Grado de satisfacción
8,76 – 10,00	Cumple con requisitos	Muy satisfactorio
5,10 – 8,75	Aceptable	Satisfactorio
2,76 – 5,00	Mínimamente aceptable	Insatisfactorio
0,00 – 2,75	Inaceptable	

Fuente: (ISO/IEC, 2011)

2.3 Ejecución de la evaluación

En esta sección se describen las herramientas y en entorno en el que se van a ejecutar las pruebas para la evaluación.

2.3.1 Conduit, aplicación de referencia para la evaluación

RealWorld[6] es un proyecto pensado para ayudar a los desarrolladores en el aprendizaje de nuevos frameworks. El proyecto pone a disposición varias implementaciones de un blog llamado Conduit desarrollado por expertos usando diferentes frameworks back-end y front-end, la idea del proyecto es mostrar las características y la forma de uso de los diferentes frameworks en una aplicación del mundo real. Las diferentes implementaciones que el proyecto dispone siguen una guía para su creación donde se describe las características técnicas que deben tener las implementaciones ya sea de front-end o back-end por lo que las diferentes implementaciones ofrecen el mismo producto final.

Para realizar las diferentes pruebas y mediciones se hace uso las implementaciones de React[7] (versión 17.0.2) y Vue.js[8] (versión 3.2.45) del blog Conduit proporcionado por el proyecto RealWorld.

2.3.2 Chrome DevTools y Lighthouse

Chrome DevTools[9] es un conjunto de herramientas para desarrollo web disponibles en el navegador web Google Chrome.

Lighthouse[10] es una herramienta de código abierto proporcionada por goolgle, permite auditar el rendimiento, accesibilidad, entre otras cosas, de páginas web.

Estas herramientas se usan para hacer las mediciones correspondientes al uso de recursos y rendimiento de la aplicación de prueba.

2.3.3 Especificaciones técnicas del entorno de ejecución de pruebas

En este apartado se muestran las especificaciones técnicas del entorno de pruebas para la evaluación.

[6] Proyecto RealWorld: https://realworld-docs.netlify.app/
[7] Conduit React: https://github.com/romansndlr/react-vite-realworld-example-app.
[8] Conduit Vue.js: https://github.com/mutoe/vue3-realworld-example-app
[9] Chrome DevTools: https://developer.chrome.com/docs/devtools/
[10] Lighthouse: https://developer.chrome.com/docs/lighthouse/overview/

TABLA 2.5 Especificaciones técnicas del entorno de pruebas.

Procesador	Intel(R) Core (TM) i5-5200U CPU @ 2.20GHz 2.20 GHz
RAM instalada	8 GB
Sistema Operativo	Windows 10 Home versión 21H2 x64
Entorno de ejecución	Node v18.12.1
React	v17.0.2
Vue.js	v3.2.45
Navegador Web	Google Chrome v108.0.5359.125

2.4 Matriz de calidad

Para la realizar la evaluación de los frameworks se hace uso de la matriz de calidad diseñada por Vaca Sierra (2017), a continuación, se describe el instrumento.

2.4.1 Preliminares

Antes de iniciar con la aplicación de las métricas el evaluador debe indicar los datos preliminares del producto como el tipo de producto (TABLA 2.1), los niveles de importancia establecido (TABLA 2.3) para las características y subcaracterísticas, y como se van a ponderar cada una de estas de acuerdo con un porcentaje.

2.4.2 Componentes: Calidad Externa, Interna, en Uso

La matriz de calidad consta de tres componentes destinados a métricas: calidad interna, calidad externa y calidad en uso, en cada componente se detalla las métricas. La matriz hace uso de las métricas indicadas en la sección 2.2.1 Selección de métricas de este capítulo.

Los tres componentes trabajan bajo el mismo esquema, con los siguientes campos, descritos por Vaca Sierra (2017):

a. **Característica:** nombre de característica.
b. **Subcaracterística:** nombre de subcaracterística.
c. **Métrica:** nombre de métrica.
d. **Propósito-métrica:** indica el objetivo de medición de la métrica.
e. **Método de aplicación:** indica la acción que debe realizarse para medir la métrica.
f. **Fase ciclo de vida de calidad del producto**: indica si la métrica pertenece a fase interna, externa, interna/eterna o en uso.
g. **Fórmula / Variables:** fórmula de la métrica y variables que interactúan en la fórmula.

h. Peor caso: se refiere al valor mínimo luego de aplicar la fórmula.

i. Valor deseado: se refiere al valor máximo luego de aplicar la fórmula.

j. Aplica: campo para indicar si se aplica o no la métrica.

k. Variables: campos para valores de variables A, B o T.

l. Valor obtenido: valor X que se obtiene aplicando la fórmula.

m. Valor Métrica / 10: valor de la métrica sobre 10 luego de aplicar la fórmula.

n. Final Subcaracterística: promedio de valores obtenidos de las métricas que son parte de la subcaracterística multiplicado por el porcentaje asignado a la subcaracterística.

o. Total Característica: sumatoria de valores finales de subcaracterísticas que componen a la característica.

p. Final Característica: producto del campo "Total Característica" por el porcentaje de importancia asignada a cada característica.

q. Calidad Interna del Sistema: sumatoria de valores finales de características de calidad. (pág. 138)

2.4.3 Procedimiento para aplicar Matriz de Calidad

Para aplicar la matriz de calidad Vaca Sierra (2017) describe el siguiente procedimiento.

1. **En Preliminares:**
    a. Ingresar datos informativos del producto software.
    b. Especificar el tipo de producto software a evaluar, según la TABLA 2.1.
    c. Asignar el nivel y porcentaje de importancia a características y subcaracterísticas de calidad interna, externa y en uso, el Nivel de Importancia (TABLA 2.3) y Ponderación de características y subcaracterísticas de calidad.
2. **En Calidad Interna, Calidad Externa y Calidad en Uso:**
    a. Seleccionar en columna "Aplica" Si o No se medirá determinada métrica.
    b. Ingresar valores de variables A, B o T (columna Variables A, B, T) de las fórmulas correspondientes a cada métrica seleccionada.
    c. El valor X es el resultado obtenido de la aplicación de la fórmula de acuerdo con las variables ingresadas, su cálculo es automático.

d. Obtenidos los resultados de las fórmulas, automáticamente se calcularán las siguientes columnas:

   - Valor Métrica / 10
   - Final Subcaracterística
   - Total Característica
   - Final Característica
   - Calidad Parcial del Sistema

3. **En Resultado Final:** se presentará el resultado final del análisis de calidad del producto software de acuerdo con lo indicado en el literal c) de la sección Niveles de calificación, determinando de esta manera el valor parcial y total de calidad por cada componente del producto software, nivel de puntuación y grado de satisfacción. (págs. 139-140)

2.4.4 Definición de características y subcaracterísticas de calidad

En la Fig. 16 se muestra cómo se establecieron los niveles de importancia para cada una de las características del modelo de calidad.

4. CARACTERÍSTICAS DE CALIDAD EXTERNA		
Nombre	**Nivel de Importancia**	**%**
C1 - Adecuación Funcional	Alta	30%
C2 - Fiabilidad	No Aplica	0%
C3 - Eficiencia en el desempeño	Alta	25%
C4 - Facilidad de Uso	Alta	15%
C5 - Seguridad	No Aplica	0%
C6 - Compatibilidad	Media	5%
C7 - Mantenibilidad	Alta	20%
C8 - Portabilidad	Media	5%
Total		**100%**

Fig. 16. Definición de características de calidad

Fuente: Matriz de calidad.

En la Fig. 17 se muestran las subcaracterísticas y los niveles de importancia establecidos para las mismas.

6. SUBCARACTERÍSTICAS DE CALIDAD EXTERNA				
Característica	**Subcaracterística**	**Nivel Importan**	**%**	**Total Característi**
C1 - Adecuación Funcional	Completitud funcional	Alta	100%	**100%**
	Exactitud funcional	No Aplica	0%	
C2 - Fiabilidad	Madurez	No Aplica	0%	**0%**
	Disponibilidad	No Aplica	0%	
	Tolerancia a fallos	No Aplica	0%	
	Recuperabilidad	No Aplica	0%	
C3 - Eficiencia en el desempeño	Comportamiento del tiempo	Alta	50%	**100%**
	Utilización de recursos	Alta	50%	
	Capacidad	No Aplica	0%	
C4 - Facilidad de Uso	Capacidad de reconocer su adecuación	No Aplica	0%	**100%**
	Capacidad para ser entendido	Alta	90%	
	Operatividad	No Aplica	0%	
	Protección contra errores del usuario	Baja	10%	
	Estética de la Interfaz del usuario	No Aplica	0%	
	Accesibilidad técnica	No Aplica	0%	
C5 - Seguridad	Confidencialidad	No Aplica	0%	**0%**
	Integridad	No Aplica	0%	
	No repudio	No Aplica	0%	
	Responsabilidad	No Aplica	0%	
	Autenticidad	No Aplica	0%	
C6 - Compatibilidad	Co – existencia	Media	100%	**100%**
	Interoperatividad	No Aplica	0%	
C7 - Mantenibilidad	Capacidad de ser analizado	Media	80%	**100%**
	Capacidad de ser modificado	No Aplica	0%	
	Capacidad de ser probado	Media	20%	
C8 - Portabilidad	Adaptabilidad	No Aplica	0%	**100%**
	Capacidad de ser Instalado	Media	100%	
	Capacidad de ser Reemplazado	No Aplica	0%	

Fig. 17. Definición de subcaracterísticas de calidad

Fuente: Matriz de calidad.

En la TABLA 2.6 se muestra las subcaracterísticas del modelo de calidad que se descartaron y el motivo de por qué se descartaron.

TABLA 2.6 Lista de subcaracterísticas descartadas.

Subcaracterística	**Motivo**
Madurez	Para poder medir está característica se necesita tener un conocimiento profundo de las herramientas.
Disponibilidad	Los frameworks en estudio no cuentan con recursos que satisfagan esta característica, la disponibilidad de la herramienta depende de factores externos.
Tolerancia del error	Los frameworks en estudio no cuentan con recursos que satisfagan esta característica.
Capacidad de recuperación	Los frameworks en estudio no guardan el estado de los datos, solo se encargan de la interfaz de usuario.
Confidencialidad	Los frameworks en estudio no manejan la autorización o autenticación en los sistemas
Integridad	Los frameworks en estudio no se encargan del estado de los datos en la base de datos.
No repudio	Los frameworks en estudio no registran las acciones realizadas por el usuario.
Responsabilidad	Los frameworks en estudio no registran las acciones realizadas por el usuario.
Autenticidad	Los frameworks en estudio no disponen de funciones para comprobar la autenticidad de la información

Capítulo III

3.1 Planificación del proyecto

En esta sección se describen las historias de usuario y tareas de estas, así como el plan que se va a seguir para ejecutar cada iteración.

3.1.1 Equipo de trabajo

Se define el equipo de trabajo que está involucrado en el proyecto, además de los roles que ejercerá cada miembro del equipo, ver *TABLA 3.1.*

TABLA 3.1 Equipo de trabajo

Nombre	Descripción	Rol
MSc. Daisy Elizabeth Imbaquingo Esparza	Encargada de revisar avances del sistema	Consultor
Miguel Paucar	Encargado de pruebas funcionales	Cliente
Javier Paucar	Encargado del desarrollo del sistema	Programador

3.1.2 Tipos de usuarios del sistema

a) Usuario Administrador. - Usuario con permisos para gestionar registros de usuarios, sucursales.

b) Usuario Supervisor. - Usuario con permisos para gestionar usuarios de sucursales, registros de productos, categorías de productos, inventarios, registros de caja y reportes.

c) Usuario Vendedor. - Usuario con permisos para registrar ventas y realizar movimientos en caja.

3.1.3 Historias de usuario

Como parte de la fase de planificación se elaboraron las historias de usuario, a cada una de estas historias de usuario se le ha asignado un nivel de prioridad y el riesgo en el desarrollo, los niveles pueden ser alto, medio y bajo, además se indica también el número de iteración a la que corresponde cada historia. En la TABLA 3.2 se muestra la lista de historias de usuario.

TABLA 3.2 Lista de historias de usuario.

Nro.	Nombre	Prioridad	Riesgo	Iteración
H1	Control de acceso de usuarios	Alta	Medio	1
H2	Gestión de usuarios, sucursales y clientes	Alta	Medio	1
H3	Manejo de varias sucursales	Media	Medio	1
H4	Manejo independiente de existencias en cada sucursal	Media	Medio	1
H5	Gestión de catálogo de productos	Alta	Medio	2
H6	Ajuste de cantidades en productos por sucursal	Media	Medio	2
H7	Ver estado del inventario por sucursal	Alta	Medio	2
H8	Registro de ventas	Alta	Alto	2
H9	Registro de movimientos de caja	Alta	Alto	3
H10	Ver detalles de movimientos de caja	Alta	Medio	3
H11	Autorización de usuarios	Alta	Medio	3

3.1.4 Planificación de lanzamiento

Una vez establecidas las historias de usuario se realiza la planificación de lanzamientos o entregas que se muestra en la TABLA 3.3.

TABLA 3.3 Plan de lanzamientos

Nro.	Nombre de Historia	Iteración		
		1	2	3
1	Control de acceso de usuarios	x		
2	Gestión de usuarios, sucursales y clientes	x		
3	Manejo de varias sucursales	x		
4	Gestión de catálogo de productos	x		
5	Manejo independiente de existencias en cada sucursal		x	
6	Ajuste de cantidades en productos por sucursal		x	
7	Ver estado del inventario por sucursal		x	
8	Registro de ventas		x	
9	Registro de movimientos de caja			x
10	Ver detalles de movimientos de caja			x
11	Autorización de usuarios			x

3.1.5 Iteraciones

En la

TABLA 3.4 se muestra la planificación de iteraciones que consta de un tiempo estimado de 11 semanas, en el transcurso de las cuales se distribuye el desarrollo de las tareas de las historias de usuario.

TABLA 3.4 Plan de iteraciones

Iteración	Nro. Historia	Semanas										
		1	2	3	4	5	6	7	8	9	10	11
1	1											
	2											
	3											
	4											
2	5											
	6											
	7											
	8											
3	9											
	10											
	11											

a) Historias de usuario por iteración

- **Iteración 1. –** En las siguientes tablas se describen las historias de usuarios planificadas para la iteración 1.

TABLA 3.5 Historia de usuario 1

	Historia de usuario
Número: 01	**Usuario:** Todos
Nombre historia: Control de acceso de usuarios	
Prioridad: Alta	**Riesgo de desarrollo:** Media
Programador: Javier Paucar	**Iteración:** 1
Descripción: Para que los usuarios tengan acceso a la aplicación se solicita una identificación y una clave.	

TABLA 3.6 Historia de usuario 2

	Historia de usuario
Número: 02	**Usuario:** Administrador
Nombre historia: Gestión de usuarios, sucursales y clientes	
Prioridad: Alta	**Riesgo de desarrollo:** Media
Programador: Javier Paucar	**Iteración:** 1

Descripción: El usuario podrá gestionar registros de usuarios, sucursales y clientes.

TABLA 3.7 Historia de usuario 3

Historia de usuario	
Número: 03	**Usuario:** Administrador
Nombre historia: Manejo de varias sucursales	
Prioridad: Media	**Riesgo de desarrollo:** Media
Programador: Javier Paucar	**Iteración:** 1
Descripción: El sistema deberá permitir registrar varias sucursales del negocio y sus respectivos usuarios.	

TABLA 3.8 Historia de usuario 4

Historia de usuario	
Número: 04	**Usuario:** Supervisor
Nombre historia: Manejo independiente de existencias en cada sucursal	
Prioridad: Media	**Riesgo de desarrollo:** Media
Programador: Javier Paucar	**Iteración:** 1
Descripción: El sistema debe permitir manejar las existencias de los productos independientemente en cada sucursal registrada.	

- **Iteración 2. -** En las siguientes tablas se describen las historias de usuarios planificadas para la iteración 2.

TABLA 3.9 Historia de usuario 5

Historia de usuario	
Número: 05	**Usuario:** Supervisor
Nombre historia: Gestión de catálogo de productos	
Prioridad: Alta	**Riesgo de desarrollo:** Media
Programador: Javier Paucar	**Iteración:** 2
Descripción: El usuario podrá gestionar productos y categorías de producto.	

TABLA 3.10 Historia de usuario 6

Historia de usuario	
Número: 06	**Usuario:** Supervisor
Nombre historia: Ajuste de cantidades en productos por sucursal	
Prioridad: Media	**Riesgo de desarrollo:** Media
Programador: Javier Paucar	**Iteración:** 2
Descripción: Permitir reemplazar, aumentar o disminuir la cantidad de un producto en una sucursal determinada.	

TABLA 3.11 Historia de usuario 7

Historia de usuario	
Número: 07	**Usuario:** Supervisor
Nombre historia: Ver estado del inventario por sucursal	
Prioridad: Alta	**Riesgo de desarrollo:** Media
Programador: Javier Paucar	**Iteración:** 2
Descripción: Mostrar un reporte con el estado del inventario en una sucursal determinada.	

TABLA 3.12 Historia de usuario 8

Historia de usuario	
Número: 08	**Usuario:** Vendedor
Nombre historia: Registro de ventas	
Prioridad: Alta	**Riesgo de desarrollo:** Alta
Programador: Javier Paucar	**Iteración:** 2
Descripción: El sistema debe permitir al vendedor registrar las ventas que realice durante su jornada.	

- **Iteración 3. -** En las siguientes tablas se describen las historias de usuarios planificadas para la iteración 3.

TABLA 3.13 Historia de usuario 9

Historia de usuario	
Número: 9	**Usuario:** Vendedor
Nombre historia: Registro de movimientos de caja	
Prioridad: Alta	**Riesgo de desarrollo:** Alta
Programador: Javier Paucar	**Iteración:** 3
Descripción: El sistema debe permitir al vendedor registrar movimientos en caja, retiros y depósitos de dinero.	

TABLA 3.14 Historia de usuario 10

Historia de usuario	
Número: 10	**Usuario:** Supervisor
Nombre historia: Ver detalles de movimientos de caja	
Prioridad: Alta	**Riesgo de desarrollo:** Media
Programador: Javier Paucar	**Iteración:** 3
Descripción: Mostrar los registros de las transacciones hechas por un vendedor en caja.	

TABLA 3.15 Historia de usuario 11

Historia de usuario	
Número: 11	**Usuario:** Administrador
Nombre historia: Autorización de usuarios	
Prioridad: Alta	**Riesgo de desarrollo:** Media
Programador: Javier Paucar	**Iteración:** 3
Descripción: Controlar la autorización de los usuarios a zonas del sistema según el tipo de rol que cumplan en el mismo.	

b) Tareas ejecutadas en cada iteración

- **Iteración 1. –** En las siguientes tablas se describen las tareas a ejecutar en cada una de las historias de usuario de la iteración 1.

TABLA 3.16 Tarea 1 de historia de usuario 1

Tarea	
Número tarea: 1	**Número historia:** 1
Nombre tarea: Configuración de entorno de desarrollo	
Tipo de tarea: General	**Tiempo Estimado:** 12 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Instalar y configurar las herramientas necesarias para el inicio del desarrollo. **Herramientas para implementar** Xampp (Apache, PHP, MariaDB), Visual Studio Code, Composer, Node.js, Vue.js, Laravel	

TABLA 3.17 Tarea 2 de historia de usuario 1

Tarea	
Número tarea: 2	**Número historia:** 1
Nombre tarea: Configuración e implementación del tema Inspinia y vue.js en el proyecto	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 8 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Configurar los archivos que manejan los recursos de las vistas de Laravel para poder hacer uso de Inspinia y Vue.js	

TABLA 3.18 Tarea 3 de historia de usuario 1

Tarea	
Número tarea: 3	**Número historia:** 1
Nombre tarea: Desarrollo del login para el sistema	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 12 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Implementar el sistema de login que proporciona Laravel, modificar los modelos y controladores para que se ajusten a las necesidades del desarrollo, crear la interfaz de usuario para el login haciendo uso del motor de platillas Blade.	

TABLA 3.19 Tarea 1 de historia de usuario 2

Tarea	
Número tarea: 1	**Número historia:** 2
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 36 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Modificar el modelo para el usuario generado por Laravel según las necesidades del proyecto, crear el modelo para las sucursales, crear el modelo para clientes, crear los controladores que se encargan de hacer las operaciones CRUD para usuarios, sucursales y clientes, crear las interfaces de usuario para las operaciones CRUD de usuarios, sucursales y clientes haciendo uso de Blade y Vue.js	

TABLA 3.20 Tarea 1 de historia de usuario 3

Tarea	
Número tarea: 1	**Número historia:** 3
Nombre tarea: Desarrollo de la funcionalidad que permita registrar varios usuarios cada sucursal	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 24 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Modificar modelos de usuarios y sucursales, añadir las funciones necesarias a los controladores de usuarios y sucursales para poder registrar varios usuarios cada sucursal, crear componentes de Vue.js para implementar la función de registro de usuarios en la interfaz de usuario de sucursales.	

TABLA 3.21 Tarea 1 de historia de usuario 4

	Tarea
Número tarea: 1	**Número historia:** 4
Nombre tarea: Desarrollo de operaciones CRUD para categorías	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 12 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Desarrollar el modelo para categorías, crear el controlador donde se definen las operaciones CRUD, crear la interfaz de usuario para las operaciones CRUD haciendo uso de Blade y Vue.js.	

TABLA 3.22 Tarea 2 de historia de usuario 4

	Tarea
Número tarea: 2	**Número historia:** 4
Nombre tarea: Desarrollo de operaciones CRUD para productos	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 24 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Desarrollar el modelo para productos, crear el controlador donde se definen las operaciones CRUD, crear la interfaz de usuario para las operaciones CRUD haciendo uso de Blade y Vue.js.	

- **Iteración 2. –** En las siguientes tablas se describen las tareas a ejecutar en cada una de las historias de usuario de la iteración 2.

TABLA 3.23 Tarea 1 de historia de usuario 5

	Tarea
Número tarea: 1	**Número historia:** 5
Nombre tarea: Desarrollo de la funcionalidad para registrar productos y existencias en cada sucursal	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 24 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Modificar modelos de productos y sucursal ,añadir las funciones necesarias a los controladores de productos y sucursales para poder registrar productos y existencias en cada sucursal.	

TABLA 3.24 Tarea 1 de historia de usuario 6

Tarea	
Número tarea: 1	**Número historia:** 6
Nombre tarea: Desarrollo de la funcionalidad para registrar todas las transacciones relacionadas con las existencias de productos	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 20 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Desarrollar el modelo para la tabla donde se registran las transacciones relacionadas con las existencias, crear un Trait con las funciones que registran las transacciones, implementar el Trait en todos los controladores que hagan operaciones con las existencias de los productos.	

TABLA 3.25 Tarea 2 de historia de usuario 6

Tarea	
Número tarea: 2	**Número historia:** 6
Nombre tarea: Desarrollo de funciones e interfaz para hacer ajustes en las cantidades de existencias de productos en cada sucursal.	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 20 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Crear controlador de inventario con funciones de ajuste de cantidades de existencias, adaptar Trait de registro de transacciones de existencias, crear interfaz de usuario para el ajuste de cantidades de existencias usando Vue.js.	

TABLA 3.26 Tarea 1 de historia de usuario 7

Tarea	
Número tarea: 1	**Número historia:** 7
Nombre tarea: Crear un reporte que permita visualizar las existencias actuales de productos en cada sucursal	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 16 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Añadir funciones de consulta de existencias al controlador de inventario, crear interfaz de usuario para mostrar el estado de existencias de los productos.	

TABLA 3.27 Tarea 1 de historia de usuario 8

	Tarea
Número tarea: 1	**Número historia:** 8
Nombre tarea: Desarrollo de funcionalidad para registrar ventas	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 20 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Crear modelo para caja, crear controlador con funciones para el registro de ventas en caja, crear Trait para registrar las transacciones de caja, implementar Trait en controlador de caja.	

TABLA 3.28 Tarea 2 de historia de usuario 8

	Tarea
Número tarea: 2	**Número historia:** 8
Nombre tarea: Desarrollo de interfaz para vendedor	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 20 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Crear interfaz de usuario para que el vendedor pueda registrar las ventas que realice durante su jornada usando Blade y Vue.js.	

- **Iteración 3. –** En las siguientes tablas se describen las tareas a ejecutar en cada una de las historias de usuario de la iteración 3.

TABLA 3.29 Tarea 1 de historia de usuario 9

	Tarea
Número tarea: 1	**Número historia:** 9
Nombre tarea: Desarrollo funcionalidad para registrar depósitos y retiros en caja	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 20 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Añadir funciones para depósitos y retiros en controlador de caja, adaptar Trait de transacciones de caja, crear componentes de Vue.js para las funcionalidades de depósito y retiro e incorporarlos a la vista del vendedor	

TABLA 3.30 Tarea 1 de historia de usuario 10

Tarea	
Número tarea: 1	**Número historia:** 10
Nombre tarea: Crear reportes y vistas con el detalle de transacciones de cada caja	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 20 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Añadir al controlador de caja funciones con consultas de las transacciones de caja, crear reportes e interfaces que muestren la información de las transacciones de caja así como cálculos que permitan verificar las transacciones.	

TABLA 3.31 Tarea 1 de historia de usuario 11

Tarea	
Número tarea: 1	**Número historia:** 11
Nombre tarea: Desarrollar la asignación de roles a usuarios	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 20 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Crear modelo para roles de usuario, modificar modelo de usuario, crear controlador para roles, hacer las modificaciones necesarias en controlador de usuarios, modificar interfaz de registro de usuarios para permitir asignar roles.	

TABLA 3.32 Tarea 2 de historia de usuario 11

Tarea	
Número tarea: 2	**Número historia:** 11
Nombre tarea: Restringir secciones del sistema según el rol del usuario	
Tipo de tarea: Desarrollo	**Tiempo Estimado:** 20 horas
Fecha inicio:	**Fecha fin:**
Programador Responsable: Javier Paucar	
Descripción: Crear un middleware que se encargue de interceptar las peticiones del usuario y autorizar o restringir el acceso según el rol del usuario, controlar la visibilidad de las interfaces por rol de usuario.	

3.2 Diseño

En esta sección se describe el diseño del sistema por medio de diagramas de diferentes tipos que muestran cómo se estructura y comporta el sistema desarrollado.

3.2.1 Diagrama de procesos

En la Fig. 18 se muestra el proceso principal del cual se encarga el sistema de control de ventas, el registro de las ventas realizadas por el vendedor. El proceso comienza con el cliente haciendo la oren de los productos que desea comprar al vendedor, el vendedor se encarga de registrar los productos seleccionados por el cliente para finalmente registrar el pago. El sistema de forma automática en una sola transacción registra el movimiento de caja con los detalles de la venta como: el valor que productos, quien vende y quien es el cliente, además registra el movimiento de inventario que consiste en actualizar las existencias de los productos vendidos.

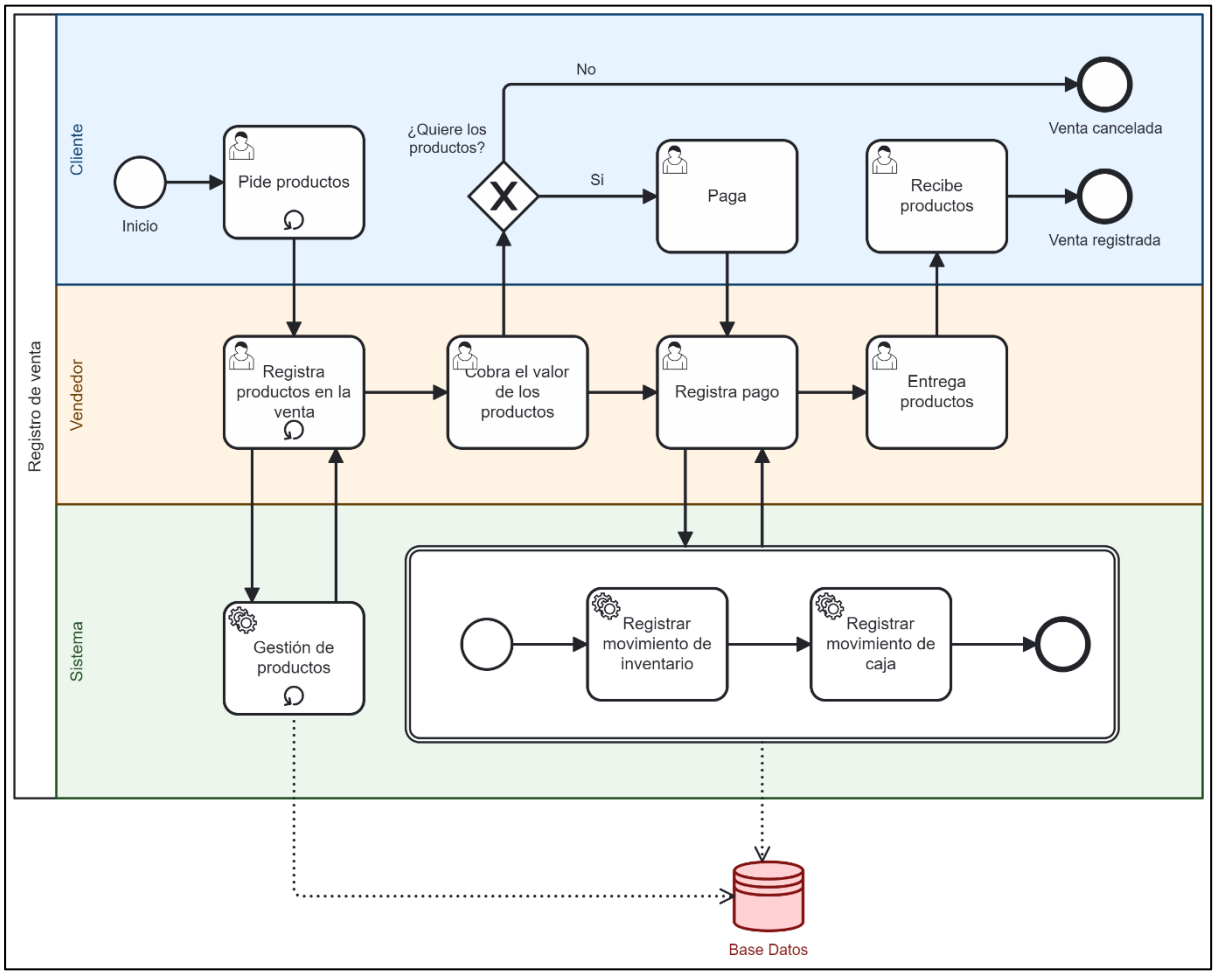


Fig. 18. Diagrama de proceso del registro de ventas.

3.2.2 Arquitectura del sistema

El sistema de control de ventas está diseñado basando en el estilo de arquitectura en capas, de tal manera que el sistema se divide en tres capas que son: la capa de datos, la capa de negocio y la capa de presentación, en la Fig. 19 se muestra una representación de la arquitectura del sistema.

Para el montaje de la aplicación se eligió usar el paquete de software XAMPP, este paquete incluye el servidor web Apache, el lenguaje de programación PHP y el servidor de base de datos MariaDB, se optó trabajar con este paquete por la facilidad que ofrece al integrar todo lo necesario para levantar una aplicación web y su facilidad de instalación y administración.

Para la construcción del sistema se usó el framework de PHP Laravel, se eligió este framework por la experiencia previa en el mismo y por la gran cantidad de recursos que se pueden encontrar y que facilitan el desarrollo de software.

Para la construcción de las interfaces gráficas del sistema se usó Vue.js, framework de JavaScript para la construcción de interfaces, la selección de este framework se la hizo de acuerdo con los resultados obtenidos en el análisis comparativo, detallados en el capítulo 2 de este documento.

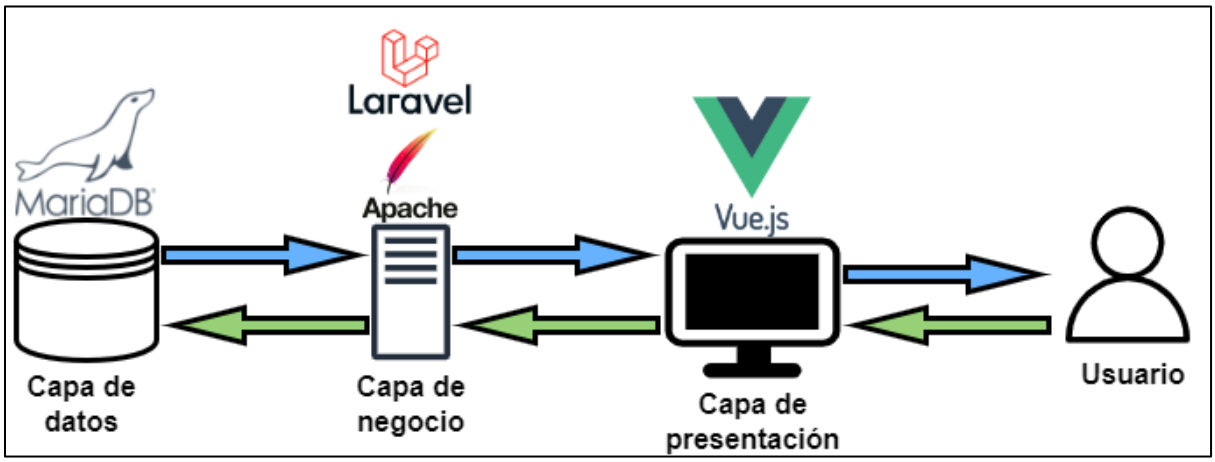


Fig. 19 Arquitectura del sistema.

3.2.3 Diagrama de base de datos

En la Fig. 20 se muestra el diagrama de la base de datos para el sistema de control de ventas. En el diagrama de base de datos permite tener una visión general del flujo de información que maneja el sistema facilitando el trabajo de implementación de los diferentes módulos del sistema en la fase de codificación.

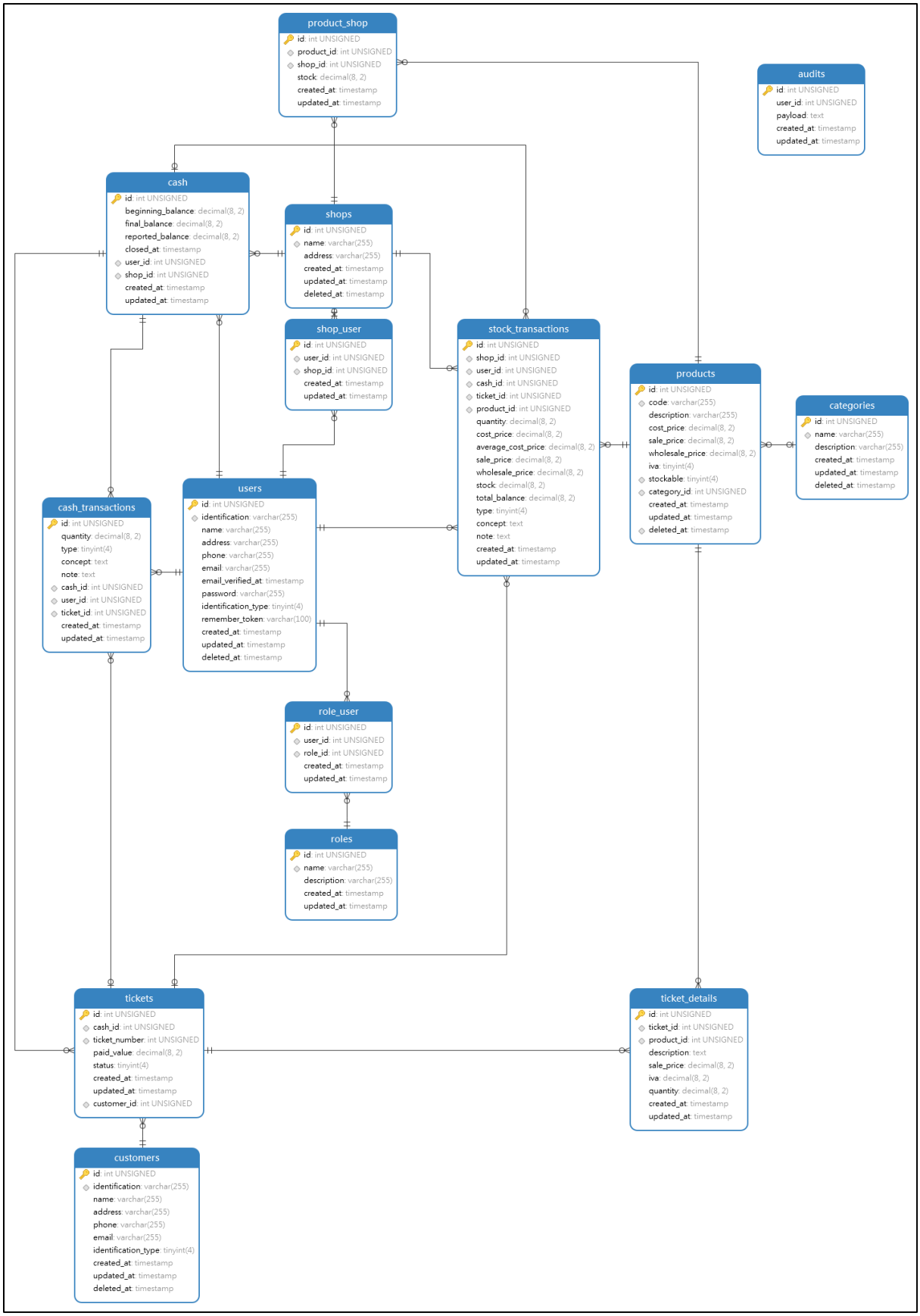


Fig. 20. Diagrama de base de datos del sistema

3.2.4 Diagramas de casos de uso

En esta sección se muestran los diagramas de casos de uso para los diferentes tipos de usuario del sistema de control de ventas.

- **Administrador**

  En la Fig. 21 se muestra el diagrama de casos de uso para el usuario administrador, en este diagrama se muestran las diferentes actividades que el usuario administrador puede realizar en el sistema, las actividades del administrador se centran en la gestión de usuarios y sucursales del sistema.

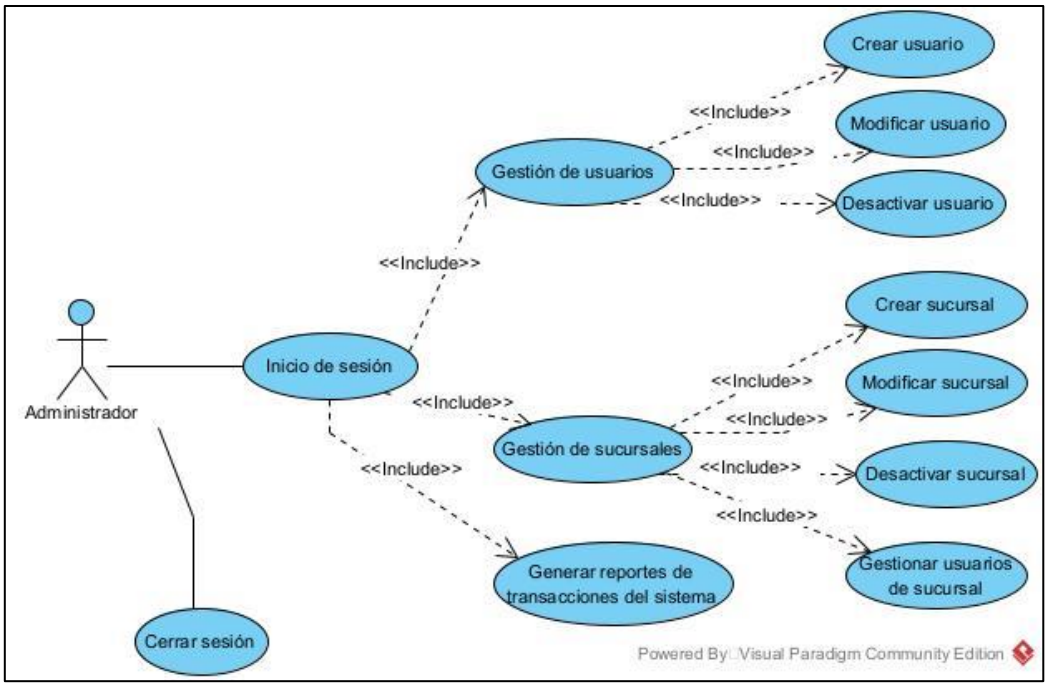


Fig. 21. Diagrama de casos de uso usuario Administrador.

- **Supervisor**

  En la Fig. 22 se muestra el diagrama de casos de uso para el usuario supervisor, en este diagrama se puede ver las actividades que puede realizar este usuario en el sistema, las actividades que realiza el usuario supervisor se centran en la gestión de productos, inventario del sistema, gestión de clientes además del control de las ventas.

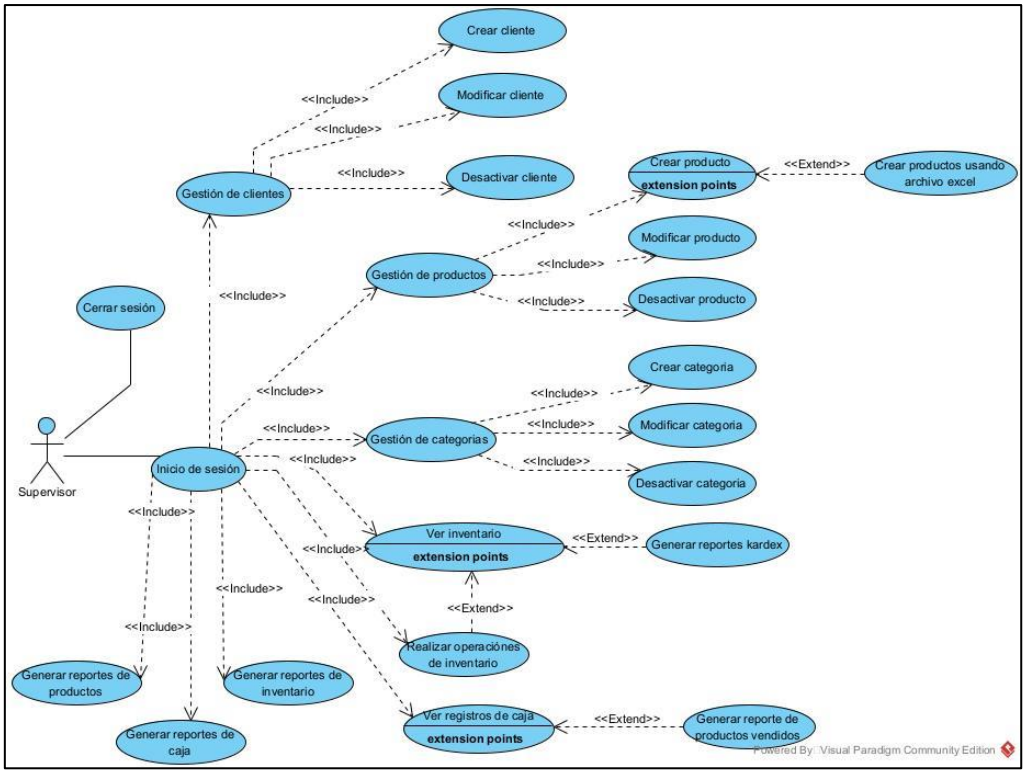


Fig. 22. Diagrama de casos de uso usuario Supervisor.

- **Vendedor**

  En la Fig. 23 se muestra el diagrama de casos de uso para el usuario supervisor, en este diagrama se puede ver las actividades que puede realizar este usuario en el sistema, el usuario vendedor es el encargado de registrar todos los movimientos de caja, también gestiona los clientes.

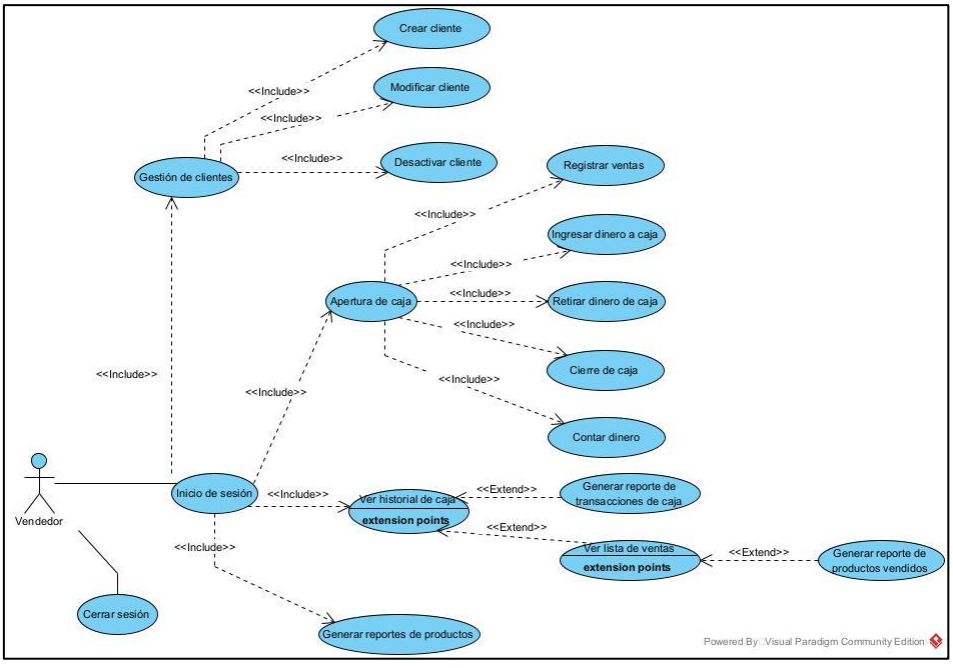


Fig. 23. Diagrama de casos de uso usuario Vendedor.

3.2.5 Diseño de interfaz de usuario

La interfaz de usuario tiene un diseño estándar donde se divide a la página en tres secciones principales que son:

- **Barra superior. –** En esta barra, muestra información básica del sistema como fecha, hora y el nombre del usuario actual, además se muestra un botón para cerrar la sesión del sistema.
- **Barra lateral. –** En esta barra se encuentra el menú principal del sistema, donde se listan todas las secciones a las que el usuario actual tiene acceso.
- **Área principal. –** Es el área principal del usuario en esta área se muestra el contenido principal de cada una de las secciones del sistema.

En la Fig. 24 se puede ver una representación gráfica del esquema general de la interfaz de usuario.

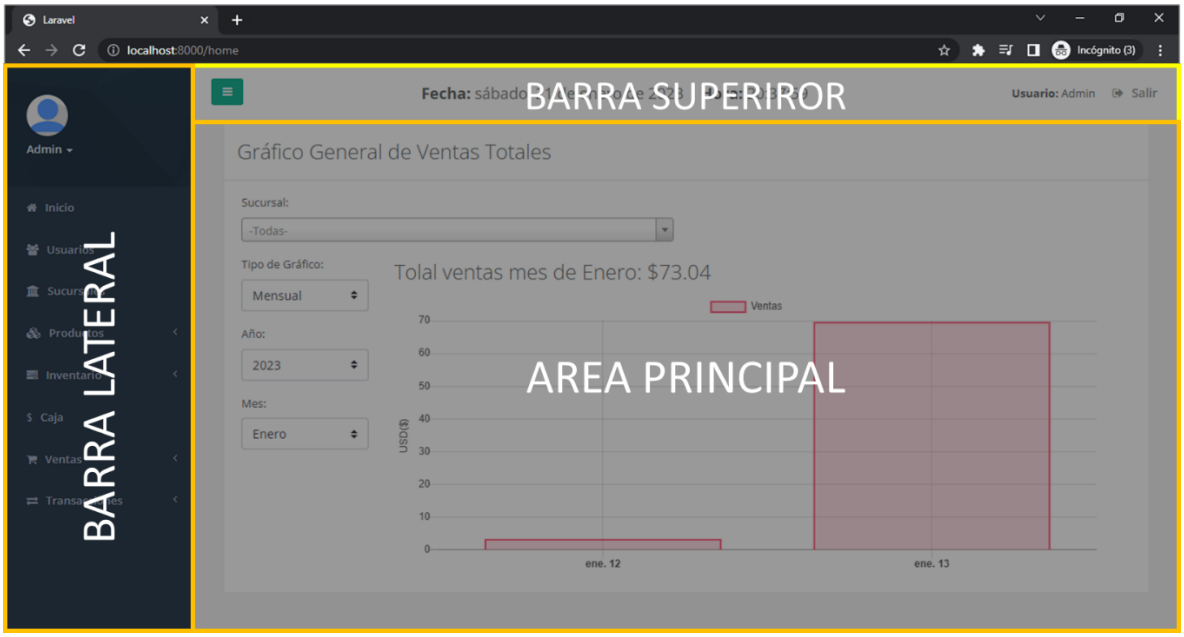


Fig. 24. Esquema general de la interfaz de usuario del sistema.

En la Fig. 25 se puede ver el diseño de la interfaz de usuario que el vendedor usa para el registro de ventas, cuenta con una barra en la parte superior donde se seleccionan los productos y la cantidad. En la parte central se listan los productos seleccionados para la venta. En la parte derecha cuenta con un panel donde se muestra el total de la venta y los accesos a las diferentes acciones como el pago de la venta y el cierre de caja.

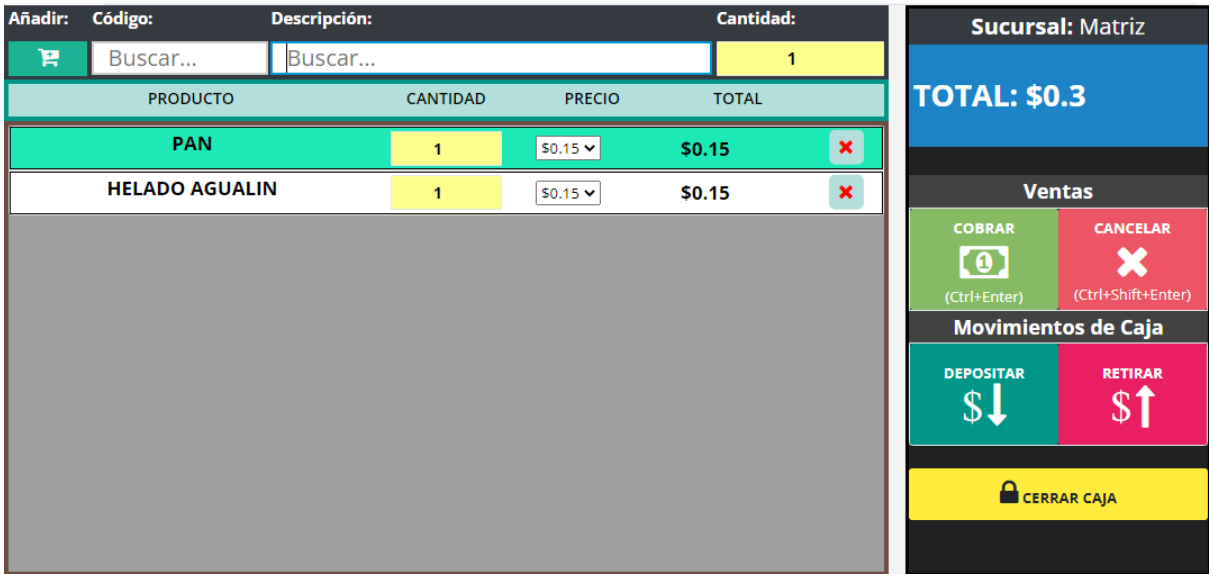


Fig. 25. Interfaz de usuario para el registro de ventas.

En la Fig. 26 se muestra la pantalla principal para la gestión de usuarios, consta de un botón que muestra un formulario para el ingreso de la información del usuario, la lista de los usuarios registrados, también en la parte superior de la lista se muestra una serie de filtros y controles que interactúan con la lista de usuarios. Además, tiene un menú con las acciones de editar y desactivar el usuario. Esta disposición de pantalla es la misma para la gestión de las diferentes entidades del sistema como sucursales, productos, categorías y clientes.

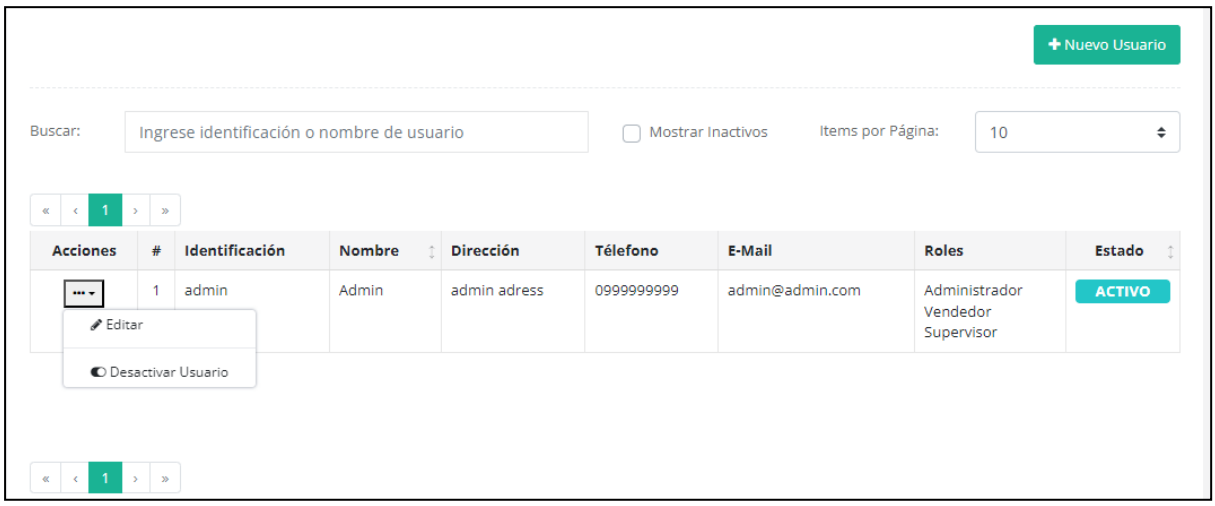


Fig. 26. Pantalla principal del módulo de gestión de usuarios.

3.3 Codificación

En esta sección se describe el patrón de diseño que se usó para desarrollar el sistema mediante el diagrama de componentes. También se muestra el diagrama de despliegue que muestra la como se dispone cada elemento de la infraestructura del sistema.

3.3.1 Diagrama de componentes

En al Fig. 27 se muestra el diagrama de componentes del sistema, en este diagrama se describe la estructura y relaciones entre los principales componentes del sistema, en seguida se describe brevemente cada uno de los paquetes:

- **Modelo. –** aquí se encuentran los modelos del ORM Eloquent[11], ORM usado por Laravel, estos modelos son los encargados de comunicarse y manipular la información de la base de datos.
- **Controlador. –** en este paquete se encuentran los controladores en estos se encapsula la funcionalidad del sistema, estos controladores hacen uso de middlewares que son componentes que validan y filtran la información que llega desde las rutas antes de comunicarla al controlador, los controladores también se encargan de mostrar las vistas según la petición obtenida.
- **Vista. –** en este paquete se encuentra todos los elementos de la interfaz de usuario, que permiten al usuario interactuar con el sistema. Aquí se integran las vistas estáticas de PHP y los componentes de Vue.js.

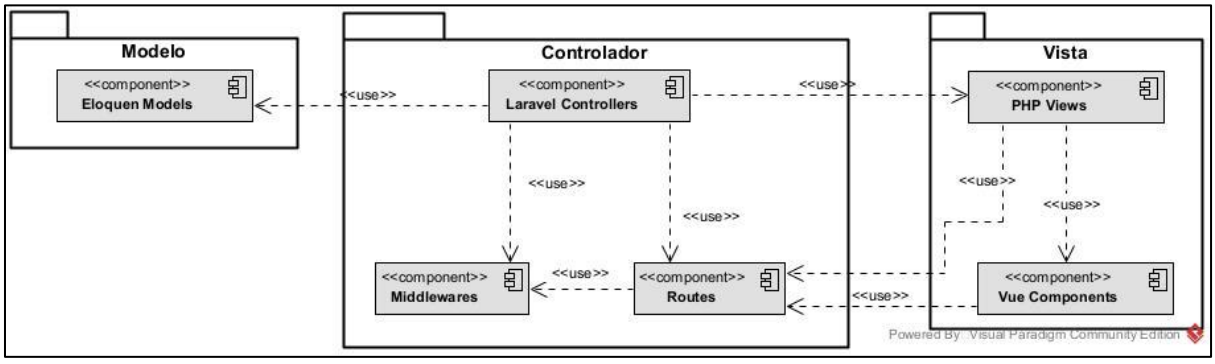


Fig. 27. Diagrama de componentes.

3.3.2 Diagrama de despliegue

En la Fig. 28 se muestra el diagrama de despliegue del sistema, este diagrama representa como los diferentes componentes y dispositivos van a estar montados en la estación de

[11] ORM Eloquent: https://laravel.com/docs/9.x/eloquent

trabajo del usuario final. Para el caso particular de este sistema todo se monta en la misma estación de trabajo.

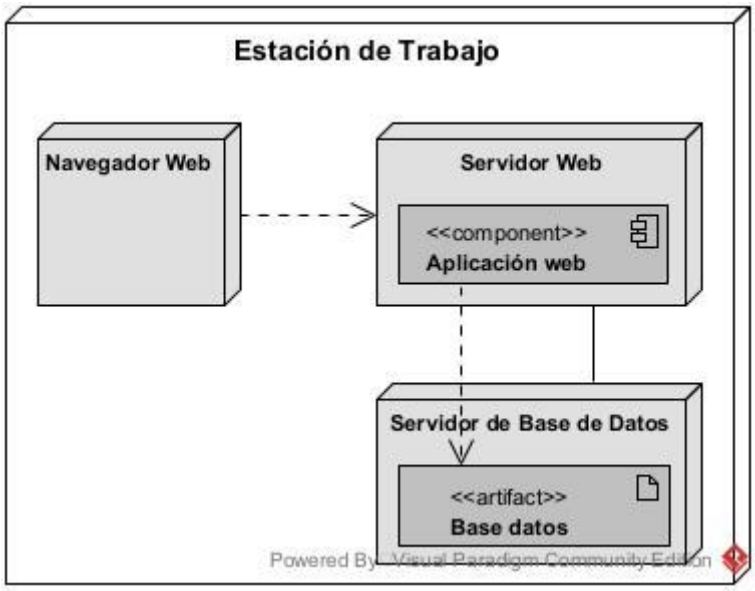


Fig. 28. Diagrama de despliegue.

3.4 Fase de pruebas

En esta sección se muestra los resultados de las pruebas hechas al software.

3.4.1 Pruebas de caja negra

Con las pruebas de caja negra se analiza el correcto funcionamiento del sistema, en estas pruebas no se toma en cuenta el funcionamiento interno de los procesos se comprueba que los resultados de cada proceso sean los correctos y cumplan con los requerimientos. En la TABLA 3.33 se muestra las pruebas de caja negra realizadas.

TABLA 3.33 Pruebas de caja negra.

Requisito	Funcionalidad	Descripción	Resultado
Control de acceso	Autenticación	Verificar que la identificación de usuario y clave de acceso sean válidos para el ingreso	✓
	Autorización	Verificar que se muestren al usuario solamente las secciones autorizadas para su rol	✓
Gestión de usuario	Registrar	Verificar al modificar el usuario los datos se guarden	✓
	Editar	Verificar que los cambios hechos a los datos del usuario se guarden	✓
	Desactivar	Verificar que el usuario desactivado no tenga acceso a ninguna función del sistema	✓
Gestión de sucursales	Registrar	Verificar que la sucursal se guarde con la información ingresada	✓

	Editar	Verificar que al modificar la sucursal los cambios se guarden	✓
	Desactivar	Verificar que la sucursal no sea accesible para los usuarios	✓
	Asignar usuarios	Verificar que los usuarios seleccionados para la sucursal tengan acceso a los datos de la sucursal	✓
Gestión de productos	Registrar	Verificar que el producto se guarde con la información ingresada	✓
	Editar	Verificar que al modificar el producto los cambios se guarden	✓
	Desactivar	Verificar que el producto no sea accesible para los usuarios	✓
Gestión de categorías	Registrar	Verificar que la categoría se guarde con la información ingresada	✓
	Editar	Verificar que al modificar la categoría los cambios se guarden	✓
	Desactivar	Verificar que la categoría no sea accesible para los usuarios	✓
Inventario	Agregar productos	Verificar que al agregar un producto al inventario se registre el movimiento y cantidades correctas	✓
	Quitar productos	Verificar que al agregar un producto al inventario se registre el movimiento y cantidades correctas	✓
	Llevar inventario por sucursal	Verificar la información de inventario de cada sucursal sea independiente	✓
Caja	Apertura de caja	Verificar que se cree un registro de caja con los valores iniciales correctos	✓
	Registrar venta	Verificar que los valores registrados en caja sean los correctos y verificar que el inventario haya variado correctamente	✓
	Ingreso	Verificar que los valores registrados en la caja sean correctos	✓
	Retiro	Verificar que los valores registrados en la caja sean correctos	✓
	Cierre de caja	Verificar que los valores de apertura, ventas, ingresos, retiros coincidan con los valores físicos	✓

Capítulo IV

Los resultados mostrados en esta sección se obtuvieron siguiendo lo descrito en la sección 2.2 donde se especifican las herramientas y características del entorno de pruebas.

4.1 Análisis Comparativo

A continuación, se exponen los diferentes resultados obtenidos con el análisis comparativo realizado.

4.1.1 Adecuación funcional

Para calificar este aspecto en las herramientas se elaboró una lista con los requerimientos funcionales necesarios con los que deben cumplir las herramientas para el desarrollo del sistema propuesto. Como se puede ver en la TABLA 4.1 React como Vue.js cumplen con el 100% de los requerimientos funcionales.

TABLA 4.1 Resultados Adecuación Funcional

Requisitos funcionales	React	Vue.js
Permite dividir la interfaz de usuario en componentes	1	1
Permite aplicar estilos en cada componente de forma independiente	1	1
Permite renderizar contenido dinámico dentro de los componentes	1	1
Tiene mecanismo para manejar el "prop drilling[12]"	1	1
Totales requisitos	**4**	**4**

1: cumple con el requerimiento
0: no cumple con el requerimiento

4.1.2 Eficiencia en el desempeño

En seguida se muestran los resultados del análisis de la característica eficiencia del desempeño del modelo de calidad.

- **Rendimiento**

  En el análisis de rendimiento realizado, usando Lighthouse, arrojo los siguientes resultados; la aplicación desarrollada con React obtuvo una calificación de 79 mientras que la aplicación desarrollada con Vue.js obtuvo 86, como se puede ver en la Fig. 29 y en la Fig. 30 la aplicación desarrollada con Vue.js destaca en las métricas de First Content Paint y Largest Contentful Paint con respecto a la aplicación desarrollada en React, lo que indica que a la aplicación hecha con React

[12] Prop drilling: Transportar información de un componente de nivel superior hasta un componente de nivel inferior usando propiedades de los componentes intermedios.

le toma más tiempo renderizar textos e imágenes, además si nos fijamos en las imágenes Vue.js obtiene mejores puntuaciones en la mayoría de las métricas de rendimiento.

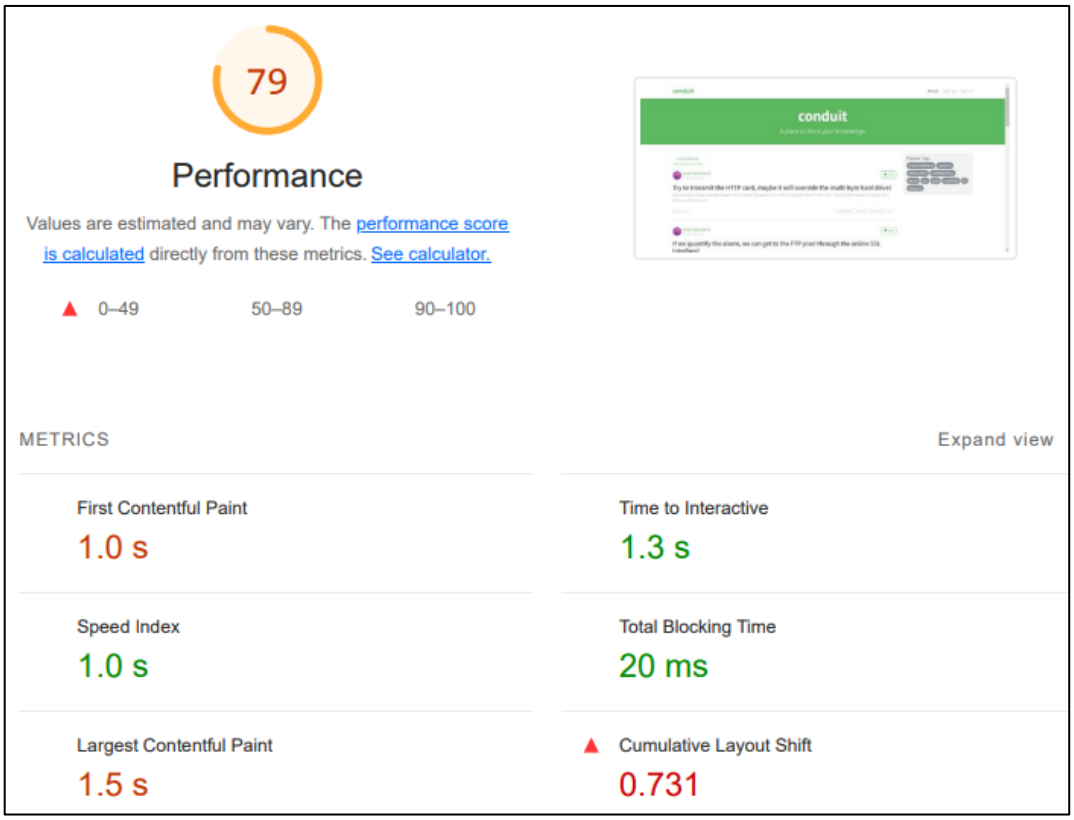


Fig. 29. Resultados de rendimiento obtenidos con Lighthouse para React.

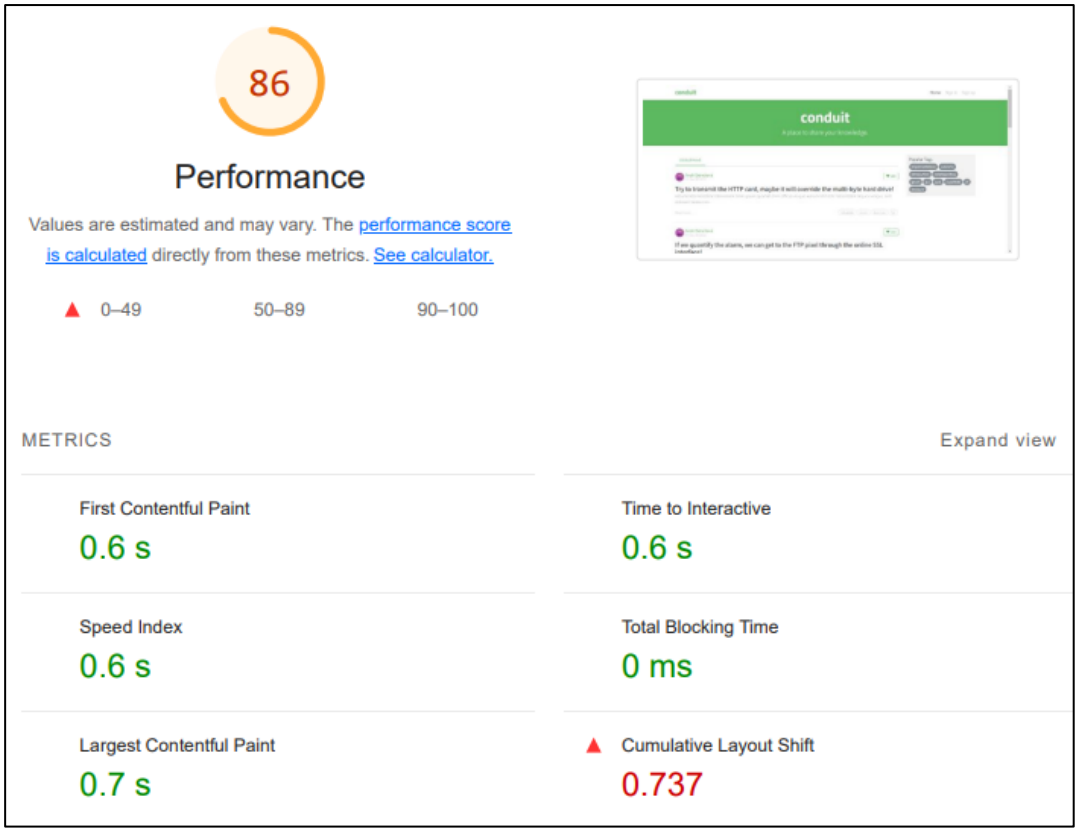


Fig. 30. Resultados de rendimiento obtenidos con Lighthouse para Vue.js.

- **Utilización de CPU**

  Los resultados obtenidos luego de generar perfiles de rendimiento, usando la herramienta Performance de Chrome DevTools, se muestran en las tablas x e y donde se pueden ver los tiempos de CPU empleados en las diferentes tareas de rendimiento analizadas para las dos aplicaciones de prueba, estas tareas son:

  – **Loading**: Mide el tiempo de tareas como las peticiones de red y el análisis de HTML.

  – **Scripting**: Mide el tiempo de análisis, compilación y ejecución del código JavaScript, además de la ejecución del recolector de basura.

  – **Rendering**: Cuanto tiempo le toma calcular estilos y diseño.

  – **Painting**: Tiempo que toma el pintado, composición, redimensionado y decodificación de imágenes.

TABLA 4.2 Tiempo de CPU usado por la aplicación de React para completar las tareas analizadas.

Tarea	Subtarea	Milisegundos (ms)
Scripting	Evaluate Module	135,5
Scripting	Event: pagehide	70,4
Scripting	XHR Ready State Change	49,1
Scripting	Function Call	23
Scripting	Run Microtasks	10,3
Scripting	Compile Module	6,8
Scripting	(anonymous)	0,3
Scripting	(anonymous)	0,1
Scripting	Cache Module Code	0,1
Scripting	XHR Load	0
Scripting	(anonymous)	0
Rendering	Layout	58,7
Rendering	Recalculate Style	26,5
Rendering	Pre-Paint	8,3
Painting	Paint	17,8
Painting	Composite Layers	1,2
Loading	Parse Stylesheet	11,6
Loading	Parse HTML	2,4

TABLA 4.3 Tiempo de CPU usado por la aplicación de React para completar las tareas analizadas.

Tarea	Subtarea	Milisegundos (ms)
Scripting	Run Microtasks	91,9
Scripting	Evaluate Module	70,7
Scripting	Event: beforeuload	38,7
Scripting	Compile Module	1
Scripting	(anonymous)	0,3
Scripting	Cache Module Code	0
Rendering	Layout	67,8
Rendering	Recalculate Style	25,5
Rendering	Pre-Paint	8,6
Painting	Paint	18,8
Painting	Composite Layers	1,3
Painting	Image Decode	0,1
Loading	Parse Stylesheet	14,1
Loading	Parse HTML	1,2

En la TABLA 4.4 se muestra la suma total de los tiempos de CPU empleados para cada una de las aplicaciones de prueba, como se ve la aplicación de Vue.js hace menos consumo de tiempo de CPU comparada con la aplicación desarrollada con React.

TABLA 4.4 Tiempo total neto usado por las aplicaciones de prueba en ejecución.

React		**Vue. js**	
Actividades	**ms**	**Actividades**	**ms**
Loading	14	Loading	15
Scripting	296	Scripting	203
Rendering	94	Rendering	102
Painting	19	Painting	20
Tiempo total de uso de CPU	**423**	**Tiempo total de uso de CPU**	**340**

- **Utilización de memoria**

Usando la herramienta Memory de Chrome DevTools se generaron perfiles de memoria para las dos aplicaciones de prueba, estos perfiles muestran cuanta memoria usa una página cargada de la aplicación, en las Figuras Fig. 31 y Fig. 32 se muestran los perfiles obtenidos para cada una de las aplicaciones, los gráficos muestran la cantidad de memoria usada por los diferentes objetos que componen la página analizada como son: el código fuente, cadenas de texto, arreglos de JavaScript, arreglos tipados de JavaScript, y otros objetos usado por el sistema donde corre el navegador.

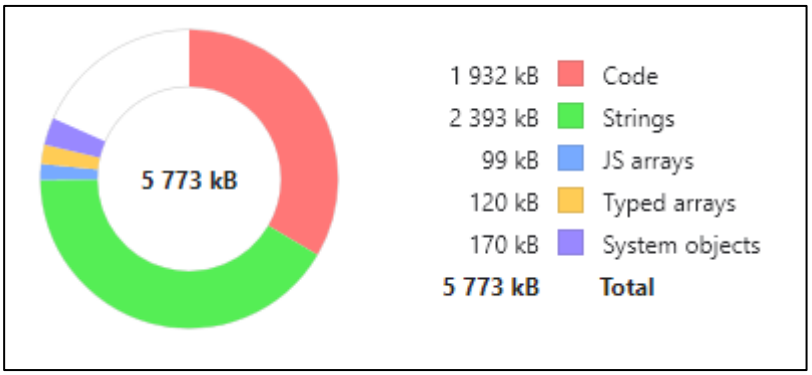


Fig. 31 Perfil de uso de memoria aplicación React.

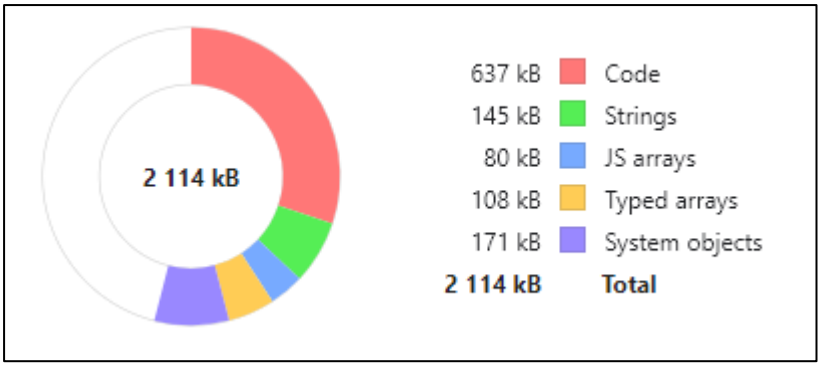


Fig. 32 Perfil de uso de memoria aplicación Vue.js.

Como se muestra en la TABLA 4.5 la aplicación desarrollada con Vue.js tiene un menor consumo de memoria comparada con la aplicación desarrollada en React.

TABLA 4.5 Memoria total usada por las aplicaciones de prueba

React	
Objetos	**kB**
Código	1932
Cadenas de texto	2393
Arreglos JS	99
Arreglos tipados	120
Objetos del sistema	170
Memoria total usada	**4714**

Vue.js	
Objetos	**kB**
Código	637
Cadenas de texto	145
Arreglos JS	80
Arreglos tipados	108
Objetos del sistema	171
Memoria total usada	**1141**

4.1.3 Facilidad de uso

En seguida se muestran los resultados del análisis de la característica facilidad de uso del modelo de calidad.

- **Documentación**

  Para obtener la puntuación en este apartado se hace un recuento de los recursos que ofrecen los dos framewokrs para que los desarrolladores puedan hacer uso de la herramienta y de sus funciones. Toda la información obtenida para realizar esta calificación se la obtuvo de la documentación oficial de cada herramienta[13] [14].

  En la TABLA 4.6 se muestra el resultado obtenido por las dos herramientas.

[13] Documentación React: https://es.reactjs.org/
[14] Documentación Vue.js: https://vuejs.org/

TABLA 4.6 Resultados puntuación de documentación

Recurso documentación	React	Vue.js
Introducción	1	1
Instalación	1	1
Conceptos básicos	1	1
Información detallada	1	1
Referencia de la API	1	1
Pruebas (Testing)	1	1
Mejores prácticas	1	1
Seguridad	0	1
Tutorial	1	1
Sección con ejemplos implementados	0	1
Traducción al español	1	0
Total recursos	**9**	**10**

- **Prevención del uso incorrecto**

  Basado en la documentación oficial de cada herramienta se obtuvieron las funciones que proporcionan un mecanismo para evitar su uso incorrecto, en este caso de los componentes creados con cada uno de los frameworks, en la TABLA 4.7 se muestra el resultado mostrando el número de funciones que proporciona cada uno.

TABLA 4.7 Resultado número de funciones para prevención de uso incorrecto

Funciones	React	Vue.js
Validar propiedades del componente	1	1
Validar los eventos que emite el componente	0	1
Total funciones	**1**	**2**

4.1.4 Mantenibilidad

En seguida se muestran los resultados del análisis de la característica mantenibilidad del modelo de calidad.

- **Capacidad de ser analizado**

  Las dos herramientas proporcionan extensiones para los diferentes navegadores que permiten probar y analizar los componentes de las aplicaciones mientras se desarrollan, para React existe React Developer Tools[15] y Vue.js dispone de Vue.js devtools[16].

[15] https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi
[16] https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnnanhbledajbpd

- **Capacidad de ser probado**

  En este apartado se indican cuantos tipos de prueba permite realizar cada una de las herramientas, con la información obtenida de la documentación oficial se muestran los resultados en la TABLA 4.8.

TABLA 4.8 Tipos de pruebas que permiten realizar los frameworks

Tipo de prueba	React	Vue.js
Pruebas Unitarias	1	1
Probar Componentes	1	1
End-to-End	1	1
Totales	**3**	**3**

4.1.5 Portabilidad

En este apartado se cuentan cuantos métodos de instalación permite cada una de las herramientas, esta información fue obtenida de la documentación oficial de las herramientas. En la TABLA 4.9 se muestran los resultados obtenidos.

TABLA 4.9 Resultado número de métodos de instalación

Método de instalación	React	Vue.js
Herramienta de creación de proyectos	1	1
Enlace CDN	1	1
Totales	**2**	**2**

4.1.6 Resultado final

En esta sección se muestra en resultado final de aplicar la matriz de calidad a las dos herramientas en estudio, React y Vue.js. Con la matriz se analizó la calidad externa de las dos herramientas según el modelo de calidad de producto presentada por la norma ISO/IEC 25010:2011.

- **React**

  Una vez aplicada la matriz de calidad al framework React este obtuvo una calificación de 7.86 en cuanto a la calidad externa. Esto significa que, para el caso de este estudio, la herramienta cumple con el 78.6% de nivel de calidad necesario para ser usada en el desarrollo del sistema web para la Panificadora Royal.

- **Vue.js**

  Una vez aplicada la matriz de calidad al framework Vue.js este obtuvo una calificación de 8.24 en cuanto a la calidad externa. Esto significa que, para el caso

de este estudio, la herramienta cumple con el 82.4% de nivel de calidad necesario para ser usada en el desarrollo del sistema web para la Panificadora Royal.

En la Fig. 33 y Fig. 34 se muestra el resumen de los resultados de obtenidos por React y Vue.js, respectivamente, con los valores parciales según el nivel de importancia y el porcentaje de importancia asignado a cada característica medida usando el modelo de calidad externa de la norma ISO/IEC 25010.

RESULTADOS DE EVALUACIÓN DE CALIDAD EXTERNA							
	Características	**Valor Parcial Total (/10)**	**Nivel de Importancia**	**Porcentaje de Importancia**	**Valor Final**	**Calidad Parcial del Sistema (/10)**	**Calidad Total del Sistema (/10)**
CALIDAD EXTERNA	Adecuación Funcional	10,00	Alta	30%	3,00	**7,86**	**7,86**
	Fiabilidad	0,00	No Aplica	0%	0,00		
	Eficiencia en el desempeño	3,71	Alta	25%	0,93		
	Facilidad de Uso	7,86	Alta	15%	1,18		
	Seguridad	0,00	No Aplica	0%	0,00		
	Compatibilidad	10,00	Media	5%	0,50		
	Mantenibilidad	10,00	Alta	20%	2,00		
	Portabilidad	5,00	Media	5%	0,25		

Fig. 33. Resumen de resultados obtenidos por React luego de aplicar la matriz de calidad.

RESULTADOS DE EVALUACIÓN DE CALIDAD EXTERNA							
	Características	**Valor Parcial Total (/10)**	**Nivel de Importancia**	**Porcentaje de Importancia**	**Valor Final**	**Calidad Parcial del Sistema (/10)**	**Calidad Total del Sistema (/10)**
CALIDAD EXTERNA	Adecuación Funcional	10,00	Alta	30%	3,00	**8,24**	**8,24**
	Fiabilidad	0,00	No Aplica	0%	0,00		
	Eficiencia en el desempeño	4,44	Alta	25%	1,11		
	Facilidad de Uso	9,18	Alta	15%	1,38		
	Seguridad	0,00	No Aplica	0%	0,00		
	Compatibilidad	10,00	Media	5%	0,50		
	Mantenibilidad	10,00	Alta	20%	2,00		
	Portabilidad	5,00	Media	5%	0,25		

Fig. 34. Resumen de resultados obtenidos por React luego de aplicar la matriz de calidad.

En la Fig. 35 se muestran los resultados finales luego de evaluar la matriz de calidad a cada una de las herramientas. Como se puede ver en las dos herramientas obtuvieron un resultado aceptable. En este caso Vue.js saca una ventaja de 0.38 puntos sobre React esta ventaja, como se puede ver en los resultados mostrados anteriormente, Vue.js obtiene ventaja en los apartados de Facilidad de uso, Mantenibilidad y principalmente en Eficiencia en el desempeño.

Con los resultados obtenidos se eligió trabajar con Vue.js para la construcción de la aplicación de control de ventas para la Panificadora Royal.

EVALUACIÓN DE CALIDAD TOTAL DEL PRODUCTO DE SOFTWARE			
React			
Componente	**Calidad del Componente**	**Nivel de Puntuación**	**Grado de Satisfacción**
Interna			
Externa	7,86	Aceptable	Satisfactorio
Uso			
Total	7,86	**Aceptable**	**Satisfactorio**

EVALUACIÓN DE CALIDAD TOTAL DEL PRODUCTO DE SOFTWARE			
Vue.js			
Componente	**Calidad del Componente**	**Nivel de Puntuación**	**Grado de Satisfacción**
Interna			
Externa	8,24	Aceptable	Satisfactorio
Uso			
Total	8,24	**Aceptable**	**Satisfactorio**

Fig. 35. Resultado evaluación matriz de calidad React y Vue.js.

Fuente: Matriz de calidad.

Conclusiones

- Para la correcta aplicación de los modelos de calidad de la norma ISO/IEC 25010 es necesario apoyarse en las demás normas de la familia ISO/IEC 25000 porque las normas se complementan entre sí.
- Luego de aplicar la matriz de calidad a cada una de las herramientas en estudio se obtuvo como resultado que Vue.js es la herramienta más adecuada para la construcción sistema web de control de ventas para la Panificadora Royal
- Con respecto a las características de React y Vue.js los dos son muy similares los diferencia la forma en la que se construyen los componentes en React están escritos 100% en JavaScript mientras que Vue.js divide sus componentes como si se tratara de una página web normal usando HTML, CSS y JavaScript, en la práctica la forma en que separa los componentes Vue.js resulta mucho más fácil de asimilar.
- El uso de la metodología XP permitió el desarrollo del sistema web de forma rápida y precisa, al estar en constante comunicación con el usuario durante el proceso de desarrollo permitió tener una idea clara de las necesidades del cliente e ir corrigiendo a tiempo las falencias y de esta manera conseguir un producto con mayor calidad.
- Cuando se maneja un negocio de cualquier rubro es fundamental llevar un control interno que le permita controlar sus recursos y mitigar riesgos. Una forma de llevar este control es implementar sistemas que permitan mantener organizada la información.

Recomendaciones

- En la parte académica se recomienda a la carrera tomar en cuenta enseñar el uso y funcionamiento de herramientas como Angular, React o Vue.js, debido a que en la actualidad son herramientas que proporcionan conocimientos valiosos que en el ámbito laboral son muy demandados.
- Se recomienda que antes de usar normas como la ISO/IEC 25010 se tenga la precaución de saber que cubre y cuáles son sus requerimientos para evitar dificultades.
- Al utilizar Vue.js para desarrollar aplicaciones se recomienda aprender de forma adecuada cómo funcionan los componentes para poder tener desarrollos organizados y de mejor calidad.
- Se recomienda a la empresa invertir en equipos adecuados para el correcto funcionamiento de sistema además de llevar un control frecuente de la información que genere el sistema una vez implementado.

Referencias

Bouckaert, S., Vanhie-Van Gerwen, J., Moerman, I., Phillips, S., & Wilander, J. (2011). *BONFIRE: benchmarking computers and computer networks*. https://eprints.soton.ac.uk/374081/

Callejas Cuervo, M., Alarcón Aldana, A. C., & Álvarez Carreño, A. M. (2017). Modelos de calidad del software, un estado del arte*. *Entramado*, *13*.

Cavaness, C. (2004). *Programming Jakarta Struts* (O'Reilly Media, Ed.; Second Edition).

Chávez, M. M. (2011). *Variables y Escalas de Medición*. https://es.slideshare.net/SCSF2011/012-variables-medicion

Clark, A. (2016, octubre 18). *React Fiber Architecture*. https://github.com/acdlite/react-fiber-architecture

Codecademy Team. (2021, septiembre 23). *What Is a Framework?* https://www.codecademy.com/resources/blog/what-is-a-framework/.

Daneva, M. (1995). *Software benchmark design and use*. Springer Science+Business Media B.V. https://link.springer.com/content/pdf/10.1007/978-0-387-34876-6_3.pdf

Etheredge, R. C. (2017). *JavaScript: Optimizing Native JavaScript Designing, Programming, and Debugging Native JavaScript Applications*. MiraVista Press. https://books.google.com.ec/books?id=6SidDgAAQBAJ

Facebook Inc. (2022). *A JavaScript library for building user interfaces - React*. https://facebook.github.io/react/

Hamedani, M. (2018, diciembre 3). *React Virtual DOM Explained in Simple English - Programming with Mosh*. https://programmingwithmosh.com/react/react-virtual-dom-explained/

Huet, P. (2020, octubre 14). *React Hooks: Qué son y qué problemas solucionan*. OpenWebinars.

IEEE. (1990). *Standard Glossary of Software Engineering Terminology* (Patent Núm. Std 610.121990).

INEN. (2012). *Clasificación Nacional Central de Productos (CCP VER. 2.0)*. https://aplicaciones2.ecuadorencifras.gob.ec/SIN/metodologias/CPC%202.0.pdf

INEN. (2015). *SISTEMAS E INGENIERÍA DE SOFTWARE — REQUISITOS Y EVALUACIÓN DE SISTEMAS Y CALIDAD DE SOFTWARE (SQuaRE) — MODELOS DE CALIDAD DEL SISTEMA Y SOFTWARE (ISO/IEC 25010:2011, IDT)* (Patent Núm. NTE INEN-ISO/IEC 25010). Servicio Ecuatoriano de Normalización.

ISO/IEC. (2011). *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Evaluation process* (Patent Núm. ISO/IEC 25040:2011).

ISO/IEC. (2016). *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality* (Patent Núm. ISO/IEC 25023:2016).

ISO/IEC. (2017). *Systems and software engineering — Information technology project performance benchmarking framework —* (Patent Núm. ISO/IEC 29155-1).

ISO/IEC/IEEE. (2010). – *Systems and software engineering – Vocabulary* (Patent Núm. ISO/IEC/IEEE 24765:2010).

Keith, J., & Jeffrey Sambells. (2005). *DOM Scripting: Web Design with JavaScript and the Document Object Model*. Apress.

Khan, S. (2021). *What Is JavaScript Framework | General Assembly*. General Assembly Blog. https://generalassemb.ly/blog/what-is-a-javascript-framework/

Letelier, P., & Penadés, M. (2006). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica Administrativa*, *5*. http://www.cyta.com.ar/ta0502/v5n2a1.htm

MDN Web Docs. (2022a). *Acerca de JavaScript - JavaScript | MDN*. MDN Web Docs. https://developer.mozilla.org/es/docs/Web/JavaScript/About_JavaScript

MDN Web Docs. (2022b). *DOM - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN*. MDN Web Docs. https://developer.mozilla.org/es/docs/Glossary/DOM

MDN Web Docs. (2022c). *JavaScript | MDN*. MDN Web Docs. https://developer.mozilla.org/es/docs/Web/JavaScript

Meltzer, R. (2020, diciembre 3). *What is JavaScript Used For? - Lighthouse Labs*. LIGHTHOUSE LABS. https://www.lighthouselabs.ca/en/blog/what-is-javascript-used-for

Meta Open Source. (2023). *Render and Commit*. https://react.dev/learn/render-and-commit

Olawanle Joel. (2022, septiembre 6). *What is a Framework? Software Frameworks Definition*. https://www.freecodecamp.org/news/what-is-a-framework-software-frameworks-definition/.

Ouellet, C. (2022, abril 8). *Vue Render Functions: What (and How) to Use Them - Snipcart*. SNIPCART. https://snipcart.com/blog/vue-render-functions

Pavlutin, D. (2017, septiembre 25). *Dmitri Pavlutin*. 7 Architectural Attributes of a Reliable React Component. https://dmitripavlutin.com/7-architectural-attributes-of-a-reliable-react-component/

Phillips, M. H. (2006). *APPLICATION OF SOFTWARE ENGINEERING BEST PRACTICIES AND PRINCIPLES TO SMALL DEVELOPMENT TEAMS* [THE UNIVERSITY OF TEXAS AT ARLINGTON]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.614.3954&rep=rep1&type=pdf

Pressman, R. S. (2010). *Ingeniería del software. Un enfoque práctico.* (7 ma). McGRAW-HILL INTERAMERICANA EDITORES, S.A. DE C.V.

React. (2022). *React*. https://reactjs.org/

Rose, K. H. (2013). A Guide to the Project Management Body of Knowledge (PMBOK® Guide)-Fifth Edition. *Project Management Journal*, *44*(3), e1–e1. https://doi.org/10.1002/pmj.21345

Roth, S. (2020). *Two-way Data Binding in React | Articles | Sandro Roth*. sandroroth. https://sandroroth.com/blog/react-two-way-data-binding

S. Gillis, A. (2022). *What is Data Binding? - Definition from WhatIs.com*. TechTarget. https://www.techtarget.com/whatis/definition/data-binding

Spendolini, M. J. (1994). *Benchmarking*. Editorial Norma S. A.

Standard Performance Evaluation Corporation (SPEC). (2013). *SPEC Glossary*. http://spec.org/spec/glossary/.

Stapenhurst, T. (2009). *The Benchmarking Book: A how-to-guide to best practice for managers and practitioners* (1 ed). Elsevier.

TypeScript. (2022). *TypeScript: Documentation - JSX*. TypeScript. https://www.typescriptlang.org/docs/handbook/jsx.html

Urdanegui, R. (2019). El control interno en las empresas. *Review of Global Management*, *4*(1), 13. https://doi.org/10.19083/rgm.v4i1.911

Vaca Sierra, T. N. (2017). *Modelo de calidad de software aplicado al módulo de talento humano del sistema informático integrado universitario – UTN* [Tesis de maestría, Universidad Técnica del Norte]. http://repositorio.utn.edu.ec/handle/123456789/7457

Vue.js. (2022). *The Progressive JavaScript Framework*. https://vuejs.org/

Wagner, S. (2013). *Software Product Quality Control*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-38571-1

Waller, J. (2014). *Performance Benchmarking of Application Monitoring Frameworks*. Department of Computer Science, Kiel University.

Anexos

Anexo A: Métricas de calidad externa

Métricas de Calidad Interna/Externa – Característica: Adecuación Funcional.

Subcarac-teristica	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Completitud funcional	Completitud de la implementación funcional	Interna/Externa	¿Cuán completa es la implementación de acuerdo a la especificación de requerimientos?	Contar el número de las funciones indicadas en la especificación de requerimientos y el número de funciones que faltan o están incorrectas	X = A / B A = Número de funciones que están incorrectas o que no fueron implementadas B = Número de las funciones establecidas en la especificación de requisitos Dónde: B > 0	0<=X<=1 El más cercano a 0 es el mejor	X = Contable / Contable A = Contable B = Contable
Exactitud funcional	Exactitud	Interna/Externa	¿Cuánto del estándar requerido de exactitud se cumple?	Contar el número de elementos de datos implementados con el estándar especifico de exactitud y el número total de elementos de datos implementados	X = A/B A = Número de elementos de datos implementados con el estándar especifico de exactitud B = Número total de elementos de datos implementados Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable

Subcaracterística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Precisión computacional	Interna/Externa	¿Con qué frecuencia ocurren los resultados inexactos?	Contar el número de cálculos inexactos encontrados y tomar el tiempo de operación	X = A/T A = Número de cálculos inexactos encontrados T = Tiempo de operación Dónde: T > 0	X = A/T El más cercano a 0/t es el mejor. Donde el peor caso es >=10/t.	X = Contable / Tiempo A = Contable T = Tiempo

Fuente: (ISO/IEC, 2016)

Métricas de Calidad Interna/Externa – Característica: Fiabilidad.

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Madurez	Eliminación de errores	Interna/Externa	¿Cuantos errores detectados han sido corregidos?	Contar el número de fallas corregidas en la fase de diseño/codificación/pruebas y el número de fallas detectadas en las pruebas	X = A/B A = Número de fallas corregidas en la fase de diseño/codificación/pruebas B = Número de fallas detectadas en las pruebas Dónde: B > 0	0<=X<=1 Cuanto más se acerque a 1 es lo mejor	X = Contable / Contable A = Contable B = Contable
	Cobertura de pruebas	Interna/Externa	¿Cuántos casos de prueba requeridos han sido ejecutados durante la etapa de pruebas?	Contar el número de casos de pruebas realizados en un escenario de operación durante la prueba y el número de casos de prueba a ser realizados para cubrir los requerimientos	X = A/B A = Número de casos de pruebas realizados en un escenario de operación durante la prueba B = Número de casos de prueba a ser realizados para cubrir los requerimientos Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable
	Tiempo medio entre fallos	Externa	¿Cuál es la frecuencia en que el sistema falla en la operación?	Tomar el tiempo de operación y contar el número total de fallas detectadas actualmente	X = A/T A = Número total de fallas detectadas actualmente T = Tiempo de operación Donde T > 0	X = A/T El más cercano a 0/t es el mejor	X = Contable / Tiempo A = Contable T = Tiempo

Subcarac-teristica	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Disponibilidad	Tiempo de servicio	Externa	¿Cuál es el tiempo de servicio del sistema que proporciona realmente?	Tomar el tiempo de servicio del sistema que se proporciona actualmente y tomar el tiempo de servicio del sistema regulado en el cronograma operacional	X = A/B A = Tiempo de servicio del sistema que se proporciona actualmente B = Tiempo de servicio del sistema regulado en el cronograma operacional Dónde: B > 0	0<=X<=1 Cuanto más se acerque a 1 es lo mejor	X = Tiempo / Tiempo A = Tiempo B = Tiempo
	Tiempo medio de inactividad	Externa	¿Cuál es el tiempo promedio que el sistema está inactivo después de que ocurre un fallo?	Tomar el tiempo total de inactividad y contar el número de fallos observados	X = A/T A = Número de fallos observados T = Tiempo total de inactividad Dónde: T > 0	X = A/T El más cercano a 0/t es el mejor	X = Contable / Tiempo A = Contable T = Tiempo
Tolerancia a fallos	Prevención de fallas	Externa	¿Cuantas fallas iniciales estuvieron bajo control para evitar fallas serias y críticas?	Contar el número de ocurrencia de fallas serias y críticas evitadas contra los casos de pruebas de fallas iniciales y el número de casos de pruebas de fallas iniciales ejecutados durante las pruebas	X = A/B A = Número de ocurrencia de fallas evitadas contra los casos de pruebas de fallas iniciales B = Número de casos de pruebas de fallas iniciales ejecutados durante las pruebas Dónde: B > 0	0<=X<=1 Cuanto más se acerque a 1 es lo mejor	X = Contable / Contable A = Contable B = Contable

Subcaracterística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Redundancia	Interna/Externa	¿Cuántos tipos de componentes/ sistemas del son instalados de forma redundante para evitar un fallo en el sistema?	Contar el número total de tipos de componentes y el número de tipos de componentes instalados de forma redundante	X = A / B A= Número componentes/sistemas instalados de forma redundante B = Número total de componentes/sistemas instalados Dónde: B > 0	0<=X<=1 Cuanto más se acerque a 1 es lo mejor	X = Contable / Contable A = Contable B = Contable
	Anulación de operación incorrecta	Interna	¿Cuántas funciones son implementadas con capacidad de anular operaciones incorrectas?	Contar el número de funciones implementadas que evitan fallas críticas y serias causadas por operaciones incorrectas y contar el número operaciones incorrectas presentadas	X = A/B A = Número de operaciones incorrectas presentadas B = Número total de funciones implementadas para anular operaciones incorrectas Dónde: B > 0	0<=X<=1 Cuanto más se acerque a 0 es lo mejor	X = Contable / Contable A = Contable B = Contable
Recuperabilidad	Tiempo medio de recuperación	Interna/Externa	¿Cuál es el tiempo promedio que toma el sistema en recuperarse completamente después un fallo?	Tomar el tiempo que le tomó al sistema en recuperarse y contar el número de casos en los cuales se ha observado que el sistema entró en recuperación	X = A / T A = Número de casos en los cuales se ha observado que el sistema entró en recuperación T = Tiempo que le tomó al sistema en recuperarse Dónde: T > 0	X = A/T El más cercano a 0/t es el mejor. Donde el peor caso es >=10/t.	X = Contable / Tiempo A = Contable T = Tiempo

Fuente: (ISO/IEC, 2016)

Métricas de Calidad Interna/Externa – Característica: Eficiencia en el desempeño.

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Comportamiento del tiempo	Tiempo de respuesta	Interna/Externa	¿Cuál es el tiempo estimado para completar una tarea?	Tomar el tiempo desde que se envía la petición hasta obtener la respuesta	X = B - A A= Tiempo de envío de petición B = Tiempo en recibir la primera respuesta	0<=X<=1 El más cercano a 0 es el mejor. Donde el peor caso es >=15t.	X = Tiempo - Tiempo A = Tiempo B = Tiempo
	Tiempo de espera	Interna/Externa	¿Cuál es el tiempo desde que se envía una instrucción, para que inicie un trabajo, hasta que lo completa?	Tomar el tiempo cuando se inicia un trabajo y el tiempo en completar el trabajo	X = B - A A= Tiempo cuando se inicia un trabajo B = Tiempo en completar el trabajo	0<=X<=1 El más cercano a 0 es el mejor. Donde el peor caso es >=15t.	X = Tiempo - Tiempo A = Tiempo B = Tiempo
	Rendimiento	Interna/Externa	¿Cuántas tareas pueden ser procesadas por unidad de tiempo?	Contar el número de tareas completadas en un intervalo de tiempo	X = A/T A= Número de tareas completadas T = Intervalo de tiempo Dónde: T > 0	X = A/T El más lejano a 0/t es el mejor. Donde el mejor caso es >=10/t	X = Contable / Tiempo A = Contable T = Tiempo
Utilización de recursos	Líneas de código	Interna	¿Cuántas líneas de código existen por cada función implementada ?	Contar el número de líneas de código (sin tomar en cuenta espacios ni comentarios) que existen en una determinada función	X = A A = Número de líneas de código	1<=X<=50 El más cercano a 1 es el mejor. Donde el peor caso es >=50 líneas de código	X = Contable A = Contable

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Utilización de CPU	Interna/Externa	¿Cuánto tiempo de CPU es usado para realizar una tarea dada?	Tomar el tiempo de operación y la cantidad de tiempo de CPU que se usa para realizar una tarea	X = B-A A= La cantidad de tiempo de CPU que realmente es usado para realizar una tarea B = Tiempo de operación Dónde: B > 0	0<=X<=1 Cuanto más se acerque a 0 es lo mejor. Donde el peor caso es >=15t.	X = Tiempo - Tiempo A = Tiempo B = Tiempo
	Utilización de la memoria	Interna/Externa	¿Cuánto espacio de memoria es usado para realizar una tarea dada?	Medir la cantidad total de espacios de memoria y la cantidad de espacios de memoria que realmente es usado para realizar una tarea	X = B-A A = Cantidad de espacios de memoria que realmente es usado para realizar una tarea B = Cantidad total de espacios de memoria Dónde: B > 0	0<=X<=15 El más cercano a 0 es el mejor	X = Tamaño - Tamaño A = Tamaño B = Tamaño
	Utilización de los dispositivos de E/S	Interna/Externa	¿Cuánto tiempo los dispositivos de E/S utilizan para realizar una tarea?	Tomar el tiempo de operación y el tiempo que los dispositivos de E/S pasan ocupados para realizar la tarea	X =B-A A = Tiempo que los dispositivos de E/S pasan ocupados para realizar la tarea B = Tiempo de operación Dónde: B > 0	0<=X<=15 El más cercano a 0 es el mejor	X = Tiempo - Tiempo A = Tiempo B = Tiempo

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Capacidad	Número de peticiones online	Interna/Externa	¿Cuántas peticiones online pueden ser procesadas por unidad de tiempo?	Contar el número máximo de peticiones online procesadas y tomar el tiempo de operación	X = A/T A= Número máximo de peticiones online procesada T = Tiempo de operación Dónde: T > 0	X = A/T El más lejano a 0/t es el mejor. Donde el mejor caso es >=10/t.	X = Contable / Tiempo A = Contable T = Tiempo
	Número de accesos simultáneos	Interna/Externa	¿Cuántos usuarios pueden acceder al sistema simultáneamente en un cierto tiempo?	Contar el número máximo de accesos simultáneos y tomar el tiempo de operación	X = A/T A= Número máximo de accesos simultáneos T = Tiempo de operación Dónde: T > 0	X = A/T El más lejano a 0/t es el mejor. Donde el mejor caso es >=10/t	X = Contable / Tiempo A = Contable T = Tiempo
	Sistema de transmisión de ancho de banda	Externa	¿Cuánto es el valor límite absoluto de transmisión necesaria para cumplir con las funciones?	Contar la cantidad máxima de transmisión de datos y tomar el tiempo de operación	X = A/T A= Cantidad máxima de transmisión de datos T = Tiempo de operación Dónde: T > 0	X = A/T El más lejano a 0/t es el mejor. Donde el mejor caso es >=10/t	X = Tamaño / Tiempo A = Tamaño T = Tiempo

Fuente: (ISO/IEC, 2016)

Métricas de Calidad Interna/Externa – Característica: Facilidad de Uso.

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Capacidad de reconocer su adecuación	Integridad de descripción	Interna/Externa	¿Qué cantidad de funciones (o tipos de funciones) son descritas como entendibles en la descripción del producto?	Contar el número de funciones (o tipos de funciones) descritas como entendibles en la descripción del producto y contar el número total de funciones (o tipos de funciones)	X = A/B A = Número de funciones (o tipos de funciones) descritas como entendibles en la descripción del producto B = Número total de funciones (o tipos de funciones) Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable
	Capacidad de demostración	Interna/Externa	¿Qué cantidad de funciones tienen la capacidad de demostración?	Contar el número de funciones implementadas con capacidad de demostración y contar el número total de funciones que requieren capacidad de demostración	X = A/B A = Número de funciones implementadas con capacidad de demostración B = Número total de funciones que requieren capacidad de demostración Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable
Capacidad para ser entendido	Funciones evidentes	Interna	¿Qué cantidad de funciones del producto son evidentes al usuario?	Contar el número de funciones que son evidentes al usuario y comparar con el número total de funciones.	X = A / B A= Número de funciones (o tipo de funciones) evidentes al usuario B = Número total de	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable

Subcaracterística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
					funciones (o tipo de funciones) Dónde: B > 0		B = Contable
	Efectividad de la documentación del usuario o ayuda del sistema	Interna/Externa	¿Qué cantidad de funciones están descritas correctamente en la documentación del usuario o ayuda en línea?	Contar el número de funciones descritas correctamente y contar el número total de funciones implementadas	X = A / B A= Número de funciones descritas correctamente B = Número total de funciones implementadas Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable
Operatividad	Recuperabilidad de error operacional	Interna	¿Qué cantidad de funciones pueden tolerar errores de usuario?	Contar el número de funciones implementadas con tolerancia de error de usuarios y el número total de funciones requeridas con capacidad de tolerancia.	X = A / B A= Número de funciones implementadas con tolerancia de error de usuarios B = Número total de funciones requeridas con capacidad de tolerancia. Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable
	Claridad del mensaje	Interna/Externa	¿Qué cantidad de mensajes son auto explicativo?	Contar el número de mensajes implementados con explicaciones claras y el número total de mensajes implementados	X = A / B A= Número de mensajes implementados con explicaciones claras B = Número total de mensajes implementados Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Consistencia operacional	Interna/Externa	¿Cuántas operaciones similares pueden llevarse a cabo consecuentemente?	Contar el número de operaciones que se comportan de manera incoherente y el número total de operaciones que se comportan de forma normal	X = A / B A= Número de operaciones que se comportan de manera incoherente B = Número total de operaciones que se comportan de forma normal Dónde: B > 0	0<=X<=1 El más cercano a 0 es el mejor	X = Contable / Contable A = Contable B = Contable
	Posibilidad de personalización	Interna/Externa	¿Cuántas funciones y procedimiento s operacionales puede un usuario modificar para su conveniencia?	Contar el número de funciones implementadas que pueden ser personalizados durante la operación y el número de funciones que requieran la capacidad de personalización	X = A / B A = Número de funciones implementadas que pueden ser personalizados durante la operación B = Número de funciones que requieran la capacidad de personalización Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable
Protección contra errores del usuario	Verificación de entradas válidas.	Interna/Externa	¿Qué cantidad de ítems de entrada son validados?	Contar el número de ítems de entrada que son validados y el número de ítems que necesitan ser validados	X = A/B A= Número de ítems de entrada que son validados B = Número de ítems que necesitan ser validados Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Prevención del uso incorrecto	Interna/Externa	¿Cuántas funciones tienen la capacidad de evitar operaciones incorrectas?	Contar el número de funciones implementadas para evitar fallos de funcionamiento provocados por un uso incorrecto y el número total de operaciones iniciales incorrectas	X = A/B A = Número operaciones iniciales incorrectas B = Número de funciones implementadas para evitar fallos de funcionamiento provocados por un uso incorrecto Dónde: B > 0	0<=X<=1 El más cercano a 1, el mejor	X = Contable / Contable A = Contable B = Contable
Estética de la Interfaz del usuario	Personalización de la apariencia de la interfaz del usuario	Interna/Externa	¿Qué cantidad de los elementos de la interfaz de usuario pueden ser personalizado s en apariencia?	Contar el número de tipos de elementos de interfaz que pueden ser personalizados y contar el número total de tipos de elementos de interfaz	X = A/B A= Número de elementos de interfaz que pueden ser personalizados B = Número total de elementos de interfaz Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable
Accesibilidad técnica	Accesibilidad física	Interna/Externa	¿A qué cantidad de funciones puede acceder un usuario con discapacidades físicas?	Contar el número de funciones a las que pueden acceder personas con discapacidad y contar el número total de funciones implementadas	X = A/B A= Número de funciones a las que pueden acceder personas con discapacidad B = Número total de elementos de interfaz Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable

Fuente: (ISO/IEC, 2016)

. Métricas de Calidad Interna/Externa – Característica: Seguridad

Subcarac-teristica	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Confidencialidad	Capacidad de control de acceso	Interna/Externa	¿Qué tan controlable son los accesos al sistema?	Contar el número de diferentes tipos de operaciones ilegales detectados y el número de tipos de operaciones ilegales en la especificación	X = A / B A = Número de diferentes tipos de operaciones ilegales detectados B = Número de tipos de operaciones ilegales en la especificación Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable
	Encriptación de datos	Interna/Externa	¿Qué tan correctamente es la implementación de encriptación / desencriptación de datos de acuerdo a la especificación de requerimientos?	Contar el número de elementos de datos encriptados/ desencriptados correctamente y el número de elementos de datos que requiere el encriptación/desencriptación	X = A / B A = Número de elementos de datos encriptados/ desencriptados correctamente B = Número de elementos de datos que requiere el encriptación/ desencriptación Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Integridad	Prevención de corrupción de datos	Interna/Externa	¿Hasta qué punto se puede prevenir la corrupción de datos?	Contar el número de casos de corrupción de datos ocurridos en la actualidad y el número de accesos donde se espera que ocurran daños de datos	X = A / B A = Número de casos de corrupción de datos ocurridos en la actualidad B = Número de accesos donde se espera que ocurran daños de datos Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X = Contable / Contable A = Contable B = Contable
No repudio	Utilización de firma digital	Interna/Externa	¿Qué proporción de eventos que requieran no - repudio se procesan utilizando la firma digital?	Contar el número de eventos procesados usando firma digital y el número de eventos que requieran la propiedad de no - repudio	X = A / B A = Número de eventos procesados usando firma digital B = Número de eventos que requieran la propiedad de no - repudio Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable
Responsabilidad	Capacidad de auditoría de acceso	Interna/Externa	¿Qué tan completa es la pista de auditoria en relación al acceso de los usuarios al sistema y a los datos?	Contar el número de accesos al sistema y los datos registrados en el log del sistema y el número de accesos ocurridos en la realidad	X = A / B A = Número de accesos ocurridos en la realidad B = Número de accesos al sistema y los datos registrados en el log del sistema Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Autenticidad	Métodos de autenticación	Interna/Externa	¿Qué tan bien el sistema autentica la identidad de un sujeto o recurso?	Contar el número de métodos de autenticación previstos	X = A A = Número de métodos de autenticación previstos	X>=0 Donde X es mayor a 0, siendo X el mejor igual o mayor a 2	X = Contable A = Contable

Fuente: (ISO/IEC, 2016)

Métricas de Calidad Interna/Externa – Característica: Compatibilidad

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Co – existencia	Co – Existencia disponible	Interna/Externa	¿Qué tan adaptable es el sistema en compartir su entorno con otros sistemas sin causar efectos adversos?	Contar el número de entidades con las que el producto puede coexistir y el número de entidades en el entorno de operación que requieren de coexistencia	X = A/B A = Número de entidades con las que el producto puede coexistir B = Número de entidades en el entorno de operación que requieren de coexistencia Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable
Interoperatividad	Conectividad con sistemas externos	Interna/Externa	¿Qué tan correctamente se ha implementado los protocolos de interfaz externa?	Contar el número de interfaces implementadas con otros sistemas y el número total de interfaces externas	X = A/B A= Número de interfaces implementadas con otros sistemas B = Número total de interfaces externas Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable
	Capacidad de intercambiar de datos	Interna/Externa	¿Qué tan exacto es el intercambio de datos entre el sistema otros sistemas de enlace?	Contar el número de datos que se han intercambiado sin problemas con otro sistema y el número total de datos que se intercambiarán	X = A/B A= Número de datos que se han intercambiado sin problemas con otro sistema B = Número total de datos que se intercambiarán Dónde: B > 0	0<=X<=1 El más cercano a 1 es el mejor	X = Contable / Contable A = Contable B = Contable

Fuente: (ISO/IEC, 2016)

Métricas de Calidad Interna/Externa – Característica: Mantenibilidad.

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Modularidad	Capacidad de condensación	Interna	¿Qué tan fuerte es la relación entre los componentes del sistema?	Contar el número de componentes que no son afectados por cambios de otros componentes y el número total de componentes específicos	X = A / B A = Número de componentes que no son afectados por cambios de otros componentes B = Número total de componentes específicos Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X = Contable / Contable A = Contable B = Contable
	Acoplamiento de clases	Interna	¿Qué tan fuerte es la relación entre una función del sistema con otras clases implementada s?	Contar el número de relaciones que tiene una función con respecto a otras clases	X = A A = Número de relaciones que tiene una función con respecto a otras clases	1<=X<=4 El más cercano a 1, es el mejor	X = Contable A = Contable
Reusabilidad	Ejecución de reusabilidad	Interna	¿Cuántos elementos pueden ser reutilizados?	Contar el número de elementos reutilizados y el número total de elementos de la biblioteca reutilizable	X = A / B A = Número de elementos reutilizados B = Número total de elementos de la biblioteca reutilizable Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Capacidad de ser analizado	Capacidad de pistas de auditoría	Interna/Externa	¿Los usuarios pueden identificar fácilmente la operación específica que causó el fallo?	Contar el número de datos realmente grabadas durante la operación y el número de datos previstos a grabarse para controlar el estado del sistema durante la operación	X = A / B A = Número de datos realmente grabadas durante la operación B = Número de datos previstos a grabarse para controlar el estado del sistema durante la operación Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable
	Diagnóstico de funciones suficientes	Interna/Externa	¿Hasta qué punto las funciones de diagnóstico están preparadas o hasta qué punto funcionan para el análisis causal?	Contar el número de funciones de diagnóstico implementadas y contar el número de funciones de diagnóstico requeridas en la especificación de requerimientos	X = A/B A = Número de funciones de diagnóstico implementadas B = Número de funciones de diagnóstico requeridas en la especificación de requerimientos Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable
Capacidad de ser modificado	Complejidad ciclomática	Interna	¿Cuál es la complejidad estructural de un código fuente?	Contar las instrucciones condicionales, bucles, salidas de métodos y clausulas AND y OR dentro de los condicionales.	X = A+1 A = Número de instrucciones condicionales que tiene una función	1<=X<15 El más cercano a 1, es el mejor	X = Contable A = Contable

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Profundidad de herencia	Interna	¿Qué tan profunda es la jerarquía de la herencia de las clases involucradas en una determinada función?	Contar las jerarquías empleadas en una determinada función o método.	X = A A = Número de jerarquías empleadas para una determinada función.	0<=X<=4 El más cercano a 0 es el mejor	X = Contable A = Contable
	Grado de localización de corrección de impacto	Interna/Externa	¿Hasta qué punto los problemas causados pueden tener como consecuencia un mantenimiento ?	Contar el número de fallas aparecidas después que se ha resuelto un fallo y contar el número de fallas resultas	X = A/B A = Número de fallas aparecidas después que se ha resuelto un fallo B = Número de fallas resueltas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X = Contable / Contable A = Contable B = Contable
	Complejidad de modificación	Externa	¿Con qué facilidad el desarrollador puede modificar el software para resolver problemas?	Tomar el tiempo de trabajo que le toma al desarrollador modificar y contar el número de modificaciones	X = A/T A = Número de modificaciones T = Tiempo de trabajo que le toma al desarrollador modificar Dónde: T > 0	X = A/T El más lejano a 0/t es el mejor	X = Contable / Tiempo A = Contable T = Tiempo

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Índice de éxito de modificación	Externa	¿Hasta qué punto puede el sistema ser operado sin fallas después del mantenimiento ?	Contar el número de problemas dentro de un determinado período antes de mantenimiento y contar el número de problemas en el mismo período después del mantenimiento	X = A/B A = Número de problemas dentro de un determinado período antes de mantenimiento B = Número de problemas en el mismo período después del mantenimiento Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X = Contable / Contable A = Contable B = Contable
Capacidad de ser probado	Completitud funcional de funciones de pruebas	Interna	¿Son las funciones de prueba, completas y fáciles de implementar?	Contar el número de funciones de prueba implementadas y contar el número de funciones de prueba requeridas	X = A/B A = Número de funciones de prueba implementadas B = Número de funciones de prueba requeridas Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable
	Capacidad de prueba autónoma	Interna	¿Qué tan independiente es el software al ser probado?	Contar el número de pruebas que están dependiendo de otros sistemas y contar el número total de pruebas dependientes con otros sistemas	X = A/B A = Número de pruebas que están dependiendo de otros sistemas B = Número total de pruebas dependientes con otros sistemas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X = Contable / Contable A = Contable B = Contable

Subcaracterística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Capacidad de reinicio de pruebas	Externa	¿Con qué facilidad se puede llevar a cabo las pruebas nuevamente después del mantenimiento ?	Contar el número de casos en los cuales el mantenedor puede pausar y restaurar las pruebas y contar el número de casos de pausa en la ejecución de pruebas	X = A/B A = Número de casos en los cuales el mantenedor puede pausar y restaurar las pruebas B = Número de casos de pausa en la ejecución de pruebas Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable

Fuente: (ISO/IEC, 2016)

Métricas de Calidad Interna/Externa – Característica: Portabilidad.

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
Adaptabilidad	Adaptabilidad en entorno hardware	Interna/Externa	¿Es el sistema lo suficientemente capaz de adaptarse al entorno hardware?	Contar el número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el entorno hardware y contar el número total de funciones las cuales han sido probadas	X = A/B A = Número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el entorno hardware B = Número total de funciones que han sido probadas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X = Contable / Contable A = Contable B = Contable
	Adaptabilidad en entorno de software	Interna/Externa	¿Es el sistema lo suficientemente capaz de adaptarse al entorno del sistema software?	Contar el número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el sistema y contar el número total de funciones las cuales han sido probadas	X = A/B A = Número de funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con el sistema B = Número total de funciones que han sido probadas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X = Contable / Contable A = Contable B = Contable

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Adaptabilidad en entorno empresarial	Interna/Externa	¿Es el sistema lo suficientemente capaz de adaptarse al entorno operacional?	Contar el número funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con usuarios del entorno empresarial y contar el número total de funciones las cuales han sido probadas	X = A/B A = Número de funciones operativas de las tareas que no se hayan completado durante las pruebas operativas con usuarios del entorno empresarial B = Número total de funciones que han sido probadas Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X = Contable / Contable A = Contable B = Contable
Capacidad de ser Instalado	Eficiencia en el tiempo de instalación	Externa	¿Cuánto tiempo es requerido para realizar una instalación?	Contar el tiempo total transcurrido al instalar el sistema y contar el número de reintentos al instalar el sistema	X = A/T A = Número de reintentos al instalar el sistema T = Tiempo total transcurrido al instalar el sistema Dónde: T > 0	X = A/T El más lejano a 0/t es el mejor	X = Contable / Tiempo A = Contable T = Tiempo

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Facilidad de instalación	Externa	¿Puede fácilmente el usuario o el desarrollador instalar el software en un entorno operacional?	Contar el número casos en que los usuarios tuvieron éxito al instalar el sistema cambiando proceso de instalación para su conveniencia y contar el número total de casos en que los usuarios han intentado cambiar el proceso de instalación para su conveniencia	X = A/B A = Número casos en que los usuarios tuvieron éxito al instalar el sistema cambiando proceso de instalación para su conveniencia B = Número total de casos en que los usuarios han intentado cambiar el proceso de instalación para su conveniencia Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable
Capacidad de ser Reemplazado	Consistencia en la función de soporte al usuario	Interna/Externa	¿Cuán consistente es el nuevo componente con la interfaz de usuario existente?	Contar el número de nuevas funciones que son consideradas como no consistentes por el usuario y contar el número de nuevas funciones	X = A/B A = Número de nuevas funciones que son consideradas como no consistentes por el usuario B = Número de nuevas funciones Dónde: B > 0	0<=X<=1 El más cercano a 0, es el mejor	X = Contable / Contable A = Contable B = Contable

Subcarac-terística	Métrica	Fase ciclo de vida	Propósito-métrica	Método de aplicación	Fórmula / Variables	Valor esperado	Tipo de medida
	Inclusividad funcional	Externa	¿Pueden fácilmente las funciones ser utilizadas después de ser cambiadas a por otras similares?	Contar el número de funciones que producen resultados similares con anterioridad y que no se han exigido cambios y contar el número de funciones probadas que son similares a las funciones proporcionadas por otro software para ser reemplazado	X = A/B A = Número de funciones que producen resultados similares con anterioridad y que no se han exigido cambios B = Número de funciones probadas que son similares a las funciones proporcionadas por otro software para ser reemplazado Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable
	Uso continuo de datos	Externa	¿Pueden los datos fácilmente ser utilizados después de reemplazar el software por otro similar?	Contar el número de datos que son continuamente utilizables por el software a ser reemplazado y contar el número de datos que son continuamente reutilizables por el software a ser reemplazado	X = A/B A = número de datos que son continuamente solo utilizables por el software a ser reemplazado B = Número de datos que son reutilizables por el software a ser reemplazado Dónde: B > 0	0<=X<=1 El más cercano a 1, es el mejor	X = Contable / Contable A = Contable B = Contable

Fuente: (ISO/IEC, 2016)

Anexo B: Preliminares y niveles de importancia de las características y subcaracterísticas del modelo de calidad de cada herramienta en estudio

MATRIZ DE CALIDAD DE SOFTWARE	
1. DATOS INFORMATIVOS:	
Fecha:	06/04/2022
Institución:	Facebook
Nombre del Software:	React
OBJETIVOS GENERALES DEL SOFTWARE	
Crear interfaces gráficas para aplicaciones web	
OBJETIVOS ESPECÍFICOS DEL SOFTWARE	
Crear interfaces gráficas para aplicaciones web	

2. TIPO DE PRODUCTO SOFTWARE		
Producto	**Clasificación de producto**	**Selección**
Página Web (PW)	Estática	
	Animada	
	Dinámica	
	Portal Web	
	Tienda Virtual o Comercio Electrónico	
	Página Web con Gestor de Contenido	
	Página Web 2.0	
Base de Datos (BDD)	BDD jerárquica	
	BDD de red	
	BDD transaccional	
	BDD relacional	
	BDD multidimensional	
	BDD orientado a objetos	
	BDD documental	
	BDD deductiva	
Software de Aplicación (SA)	SA de productividad (editores de texto)	x
	SA de entretenimiento (videojuegos)	
	SA de negocios (ERP)	
	SA de educación (programas interactivos de aprendizaje)	
	SA de tecnología (control de sistemas, médicas, etc.)	

4. CARACTERÍSTICAS DE CALIDAD EXTERNA		
Nombre	**Nivel de Importancia**	**%**
C1 - Adecuación Funcional	Alta	30%
C2 - Fiabilidad	No Aplica	0%
C3 - Eficiencia en el desempeño	Alta	25%
C4 - Facilidad de Uso	Alta	15%
C5 - Seguridad	No Aplica	0%
C6 - Compatibilidad	Media	5%
C7 - Mantenibilidad	Alta	20%
C8 - Portabilidad	Media	5%
Total		**100%**

6. SUBCARACTERÍSTICAS DE CALIDAD EXTERNA				
Característica	**Subcaracterística**	**Nivel Importan**	**%**	**Total Característi**
C1 - Adecuación Funcional	Completitud funcional	Alta	100%	**100%**
	Exactitud funcional	No Aplica	0%	
C2 - Fiabilidad	Madurez	No Aplica	0%	**0%**
	Disponibilidad	No Aplica	0%	
	Tolerancia a fallos	No Aplica	0%	
	Recuperabilidad	No Aplica	0%	
C3 - Eficiencia en el desempeño	Comportamiento del tiempo	Alta	50%	**100%**
	Utilización de recursos	Alta	50%	
	Capacidad	No Aplica	0%	
C4 - Facilidad de Uso	Capacidad de reconocer su adecuación	No Aplica	0%	**100%**
	Capacidad para ser entendido	Alta	90%	
	Operatividad	No Aplica	0%	
	Protección contra errores del usuario	Baja	10%	
	Estética de la Interfaz del usuario	No Aplica	0%	
	Accesibilidad técnica	No Aplica	0%	
C5 - Seguridad	Confidencialidad	No Aplica	0%	**0%**
	Integridad	No Aplica	0%	
	No repudio	No Aplica	0%	
	Responsabilidad	No Aplica	0%	
	Autenticidad	No Aplica	0%	
C6 - Compatibilidad	Co – existencia	Media	100%	**100%**
	Interoperatividad	No Aplica	0%	
C7 - Mantenibilidad	Capacidad de ser analizado	Media	80%	**100%**
	Capacidad de ser modificado	No Aplica	0%	
	Capacidad de ser probado	Media	20%	
C8 - Portabilidad	Adaptabilidad	No Aplica	0%	**100%**
	Capacidad de ser Instalado	Media	100%	
	Capacidad de ser Reemplazado	No Aplica	0%	

MATRIZ DE CALIDAD DE SOFTWARE

1. DATOS INFORMATIVOS:	
Fecha:	01/03/2022
Institución:	Ninguna
Nombre del Software:	Vue.js

OBJETIVOS GENERALES DEL SOFTWARE

Construir Interfaces de Usuario

OBJETIVOS ESPECÍFICOS DEL SOFTWARE

1. Crear interfaces de usuario usando JavaScript

2. TIPO DE PRODUCTO SOFTWARE		
Producto	**Clasificación de producto**	**Selección**
Página Web (PW)	Estática	
	Animada	
	Dinámica	
	Portal Web	
	Tienda Virtual o Comercio Electrónico	
	Página Web con Gestor de Contenido	
	Página Web 2.0	
Base de Datos (BDD)	BDD jerárquica	
	BDD de red	
	BDD transaccional	
	BDD relacional	
	BDD multidimensional	
	BDD orientado a objetos	
	BDD documental	
	BDD deductiva	
Software de Aplicación (SA)	SA de productividad (editores de texto)	x
	SA de entretenimiento (videojuegos)	
	SA de negocios (ERP)	
	SA de educación (programas interactivos de aprendizaje)	
	SA de tecnología (control de sistemas, médicas, etc.)	

4. CARACTERÍSTICAS DE CALIDAD EXTERNA		
Nombre	**Nivel de Importancia**	**%**
C1 - Adecuación Funcional	Alta	30%
C2 - Fiabilidad	No Aplica	0%
C3 - Eficiencia en el desempeño	Alta	25%
C4 - Facilidad de Uso	Alta	15%
C5 - Seguridad	No Aplica	0%
C6 - Compatibilidad	Media	5%
C7 - Mantenibilidad	Alta	20%
C8 - Portabilidad	Media	5%
Total		**100%**

6. SUBCARACTERÍSTICAS DE CALIDAD EXTERNA				
Característica	**Subcaracterística**	**Nivel Importan**	**%**	**Total Característi**
C1 - Adecuación Funcional	Completitud funcional	Alta	100%	**100%**
	Exactitud funcional	No Aplica	0%	
C2 - Fiabilidad	Madurez	No Aplica	0%	**0%**
	Disponibilidad	No Aplica	0%	
	Tolerancia a fallos	No Aplica	0%	
	Recuperabilidad	No Aplica	0%	
C3 - Eficiencia en el desempeño	Comportamiento del tiempo	Alta	50%	**100%**
	Utilización de recursos	Alta	50%	
	Capacidad	No Aplica	0%	
C4 - Facilidad de Uso	Capacidad de reconocer su adecuación	No Aplica	0%	**100%**
	Capacidad para ser entendido	Alta	90%	
	Operatividad	No Aplica	0%	
	Protección contra errores del usuario	Baja	10%	
	Estética de la Interfaz del usuario	No Aplica	0%	
	Accesibilidad técnica	No Aplica	0%	
C5 - Seguridad	Confidencialidad	No Aplica	0%	**0%**
	Integridad	No Aplica	0%	
	No repudio	No Aplica	0%	
	Responsabilidad	No Aplica	0%	
	Autenticidad	No Aplica	0%	
C6 - Compatibilidad	Co – existencia	Media	100%	**100%**
	Interoperatividad	No Aplica	0%	
C7 - Mantenibilidad	Capacidad de ser analizado	Media	80%	**100%**
	Capacidad de ser modificado	No Aplica	0%	
	Capacidad de ser probado	Media	20%	
C8 - Portabilidad	Adaptabilidad	No Aplica	0%	**100%**
	Capacidad de ser Instalado	Media	100%	
	Capacidad de ser Reemplazado	No Aplica	0%	

Anexo C: Matriz de calidad React

EVALUACIÓN DE CALIDAD EXTERNA																		
Característi	Subcaracterísticas	Métrica	Propósito-métrica	Método de aplicación	Fase ciclo de vida de calidad de producto	Fórmula / Variable	Peor caso	Valor Deseado	Aplica	Variables			Valor Obtenido	Valor Métrica 10	Final Subcaracterística	Total Característica	Final Característica	Calidad Externa del Sistema
										A	B	T	X					
Adecuación Funcional	Completitud funcional	Completitud de la implementación funcional	¿Cuán completa es la implementación de acuerdo a la especificación de requerimientos?	Contar el número de las funciones indicadas en la especificación de requerimientos y el número de funciones que faltan o están incorrectas	Interna/Externa	X = A / B A = Número de funciones que están incorrectas o que no fueron implementadas B = Número de las funciones establecidas en la especificación de requisitos Dónde: B > 0	1	0	Si	0,00	4,00		0,00	10,00	10	**10,00**	**3,00**	
Eficiencia en el desempeño	Comportamiento del tiempo	Tiempo de respuesta	¿Cuál es el tiempo estimado para completar una tarea?	Tomar el tiempo desde que se envía la petición hasta obtener la respuesta	Interna/Externa	X = B - A A= Tiempo de envío de petición B = Tiempo en recibir la primera respuesta	>15seg	1seg	No					0,00	1,32	**3,71**	**0,93**	
		Tiempo de espera	¿Cuál es el tiempo desde que se envía una instrucción, para que inicie un trabajo, hasta que lo completa?	Tomar el tiempo cuando se inicia un trabajo y el tiempo en completar el trabajo	Interna/Externa	X = B - A A= Tiempo cuando se inicia un trabajo B = Tiempo en completar el trabajo	>15min	15min	No					0,00				
		Rendimiento	¿Medir el rendimiento al cargar de una página web?	Obtener las siguientes métricas indicadas por Lighthouse: FCP, SI,LCP,TTI,TBT,CLS	Interna/Externa	X =Calificación calculadora Lighthouse	0	100	Si				79,00	7,90				
	Utilización de recursos	Utilización de CPU	¿Cuánto tiempo de CPU es usado para realizar una tarea dada?	Tomar el tiempo de operación y la cantidad de tiempo de CPU que se usa para realizar una tarea	Interna/Externa	X = B-A A= La cantidad de tiempo de CPU que realmente es usado para realizar una tarea B = Tiempo de operación Dónde: B > 0	>=1seg	0	Si	0,42	1,35		0,93	9,07	2,39			
		Utilización de la memoria	¿Cuánto espacio de memoria es usado para realizar una tarea dada?	Medir la cantidad total de espacios de memoria y la cantidad de espacios de memoria que realmente es usado para realizar una tarea	Interna/Externa	X = A A = Cantidad de espacios de memoria que realmente es usado para realizar una tarea	10	1	Si	4,71			4,71	5,29				
		Utilización de los dispositivos de E/S	¿Cuánto tiempo los dispositivos de E/S utilizan para realizar una tarea?	Tomar el tiempo de operación y el tiempo que los dispositivos de E/S pasan ocupados para realizar la tarea	Interna/Externa	X =B-A A = Tiempo que los dispositivos de E/S pasan ocupados para realizar la tarea B = Tiempo de operación Dónde: B > 0	>=15seg	0	No					0,00				

Facilidad de Uso	Capacidad para ser entendido	Efectividad de la documentación del usuario o ayuda del sistema	¿Qué cantidad de funciones están descritas correctamente en la documentación del usuario o ayuda en línea?	Contar el número de funciones descritas correctamente y contar el número total de funciones implementadas	Interna/Externa	X = A / B A= Número de funciones descritas correctamente B = Número total de funciones implementadas Dónde: B > 0	0	1	Si	3,00	11,00		0,82	8,18	7,36	7,86	1,18	7,86
	Protección contra errores del usuario	Verificación de entradas válidas.	¿Qué cantidad de ítems de entrada son validados?	Contar el número de ítems de entrada que son validados y el número de ítems que necesitan ser validados	Interna/Externa	X = A/B A= Número de ítems de entrada que son validados B = Número de ítems que necesitan ser validados Dónde: B > 0	0	1	No						0,5			
		Prevención del uso incorrecto	¿Cuántas funciones tienen la capacidad de evitar operaciones incorrectas?	Contar el número de funciones implementadas para evitar fallos de funcionamiento provocados por un uso incorrecto y el número total de operaciones iniciales incorrectas	Interna/Externa	X = A/B A = Número operaciones iniciales incorrectas B = Número de funciones implementadas para evitar fallos de funcionamiento provocados por un uso incorrecto Dónde: B > 0	0	1	Si	1,00	2,00		0,50	5,00				
Compatibilidad	Co - existencia	Co - Existencia disponible	¿Qué tan adaptable es el sistema en compartir su entorno con otros sistemas sin causar efectos adversos?	Contar el número de entidades con las que el producto puede coexistir y el número de entidades en el entorno de operación que requieren de coexistencia	Interna/Externa	X = A/B A = Número de entidades con las que el producto puede coexistir B = Número de entidades en el entorno de operación que requieren de coexistencia Dónde: B > 0	0	1	Si	6,00	6,00		1,00	10,00	10,00	10,00	0,50	
idad	Capacidad de ser analizado	Capacidad de pistas de auditoría	¿Los usuarios pueden identificar fácilmente la operación específica que causó el fallo?	Contar el número de datos realmente grabadas durante la operación y el número de datos previstos a grabarse para controlar el estado del sistema durante la operación	Interna/Externa	X = A / B A = Número de datos realmente grabadas durante la operación B = Número de datos previstos a grabarse para controlar el estado del sistema durante la operación Dónde: B > 0	0	1	No						8			
Mantenibili		Diagnóstico de funciones suficientes	¿Hasta qué punto las funciones de diagnóstico están preparadas o hasta qué punto funcionan para el análisis causal?	Contar el número de funciones de diagnóstico implementadas y contar el número de funciones de diagnóstico requeridas en la especificación de requerimientos	Interna/Externa	X = A/B A = Número de funciones de diagnóstico implementadas B = Número de funciones de diagnóstico requeridas en la especificación de requerimientos Dónde: B > 0	0	1	Si	1,00	1,00		1,00	10,00		10,00	2,00	
	Capacidad de ser probado	Capacidad de probar funciones y componentes	¿Probee herramientas que permitan probar funciones y componentes	Contar el tipo de pruebas que permite realizar a funciones y compoentes	Externa	X = A/B A = Número de tipo de pruebas que permite realizar B = Número de tipo de	0	1	Si	3,00	3,00		1,00	10,00	2,00			
Portabilidad	Capacidad de ser Instalado	Eficiencia en el tiempo de instalación	¿Cuánto tiempo es requerido para realizar una instalación?	Contar el tiempo total transcurrido al instalar el sistema y contar el número de reintentos al instalar el sistema	Externa	X = A/T A = Número de reintentos al instalar el sistema T = Tiempo total transcurrido	>=10min	0min	No					0,00	5	5,00	0,25	
		Facilidad de instalación	¿Puede fácilmente el usuario o el desarrollador instalar el software en un entorno operacional?	Contar cuantas formas de instalación ofrece el producto	Externa	X = A/B A = Número de métodos de instalación B = Número total de métodos de instalación Dónde: B > 0	0	1	Si	2,00	2,00		1,00	10,00				
6	3	16							10									

Anexo D: Matriz de calidad Vue.js

EVALUACIÓN DE CALIDAD EXTERNA																			
Característica	Subcaracterística	Métrica	Propósito métrica	Método de aplicación	Fase ciclo de vida de calidad del producto	Fórmula / Variable	Peor caso	Valor Deseado	Aplica	Variables			Valor Obtenido	Valor Métrica 10	Final Subcaracterística	Total Características	Final Característica	Calidad Externa del Sistema
										A	B	T	X					
Adecuación Funcional	Completitud funcional	Completitud de la implementación funcional	¿Cuán completa es la implementación de acuerdo a la especificación de requerimientos?	Contar el número de las funciones indicadas en la especificación de requerimientos y el número de funciones que faltan o están incorrectas	Interna/Externa	X = A / B A = Número de funciones que están incorrectas o que no fueron implementadas B = Número de las funciones establecidas en la especificación de requisitos Dónde: B > 0	1	0	Si	0,00	4,00		0,00	10,00	10	10,00	3,00	
Eficiencia en el desempeño	Comportamiento del tiempo	Tiempo de respuesta	¿Cuál es el tiempo estimado para completar una tarea?	Tomar el tiempo desde que se envía la petición hasta obtener la respuesta	Interna/Externa	X = B - A A= Tiempo de envío de petición B = Tiempo en recibir la primera respuesta	>15seg	1seg	No					0,00	1,43	4,44	1,11	
		Tiempo de espera	¿Cuál es el tiempo desde que se envía una instrucción, para que inicie un trabajo, hasta que lo completa?	Tomar el tiempo cuando se inicia un trabajo y el tiempo en completar el trabajo	Interna/Externa	X = B - A A= Tiempo cuando se inicia un trabajo B = Tiempo en completar el trabajo	>15min	15min	No					0,00				
		Rendimiento	¿Medir el rendimiento al cargar de una página web?	Obtener las siguientes métricas indicadas por Lighthouse: FCP, SI,LCP,TTI,TBT,CLS	Interna/Externa	X =Calificación calculadora Lighthouse	0	100	Si				86,00	8,60				
	Utilización de recursos	Utilización de CPU	¿Cuánto tiempo de CPU es usado para realizar una tarea dada?	Tomar el tiempo de operación y la cantidad de tiempo de CPU que se usa para realizar una tarea	Interna/Externa	X = B-A A= La cantidad de tiempo de CPU que realmente es usado para realizar una tarea B = Tiempo de operación Dónde: B > 0	>=1seg	0	Si	0,34	1,14		0,80	8,20	3,01			
		Utilización de la memoria	¿Cuánto espacio de memoria es usado para realizar una tarea dada?	Medir la cantidad total de espacios de memoria y la cantidad de espacios de memoria que realmente es usado para realizar una tarea	Interna/Externa	X = A A = Cantidad de espacios de memoria que realmente es usado para realizar una	10	1	Si	1,14			1,14	8,86				
		Utilización de los dispositivos de E/S	¿Cuánto tiempo los dispositivos de E/S utilizan para realizar una tarea?	Tomar el tiempo de operación y el tiempo que los dispositivos de E/S pasan ocupados para realizar la tarea	Interna/Externa	X =B-A A = Tiempo que los dispositivos de E/S pasan ocupados para realizar la tarea B = Tiempo de operación Dónde: B > 0	>=15seg	0	No					0,00				

Facilidad de Uso	Capacidad para ser entendido	Efectividad de la documentación del usuario o ayuda del sistema	¿Qué cantidad de funciones están descritas correctamente en la documentación del usuario o ayuda en línea?	Contar el número de funciones descritas correctamente y contar el número total de funciones implementadas	Interna/Externa	X = A / B A= Número de funciones descritas correctamente B = Número total de funciones implementadas Dónde: B > 0	0	1	Si	10,00	11,00		0,91	9,09	8,18	9,18	1,38	8,24
	Protección contra errores del usuario	Verificación de entradas válidas.	¿Qué cantidad de ítems de entrada son validados?	Contar el número de ítems de entrada que son validados y el número de ítems que necesitan ser validados	Interna/Externa	X = A/B A= Número de ítems de entrada que son validados B = Número de ítems que necesitan ser validados	0	1	No						1			
		Prevención del uso incorrecto	¿Cuántas funciones tienen la capacidad de evitar operaciones incorrectas?	Contar el número de funciones implementadas para evitar fallos de funcionamiento provocados por un uso incorrecto y el número total de operaciones iniciales incorrectas	Interna/Externa	X = A/B A = Número operaciones iniciales incorrectas B = Número de funciones implementadas para evitar fallos de funcionamiento provocados por un uso incorrecto Dónde: B > 0	0	1	Si	2,00	2,00		1,00	10,00				
Compatibilidad	Co - existencia	Co - Existencia disponible	¿Qué tan adaptable es el sistema en compartir su entorno con otros sistemas sin causar efectos adversos?	Contar el número de entidades con las que el producto puede coexistir y el número de entidades en el entorno de operación que requieren de coexistencia	Interna/Externa	X = A/B A = Número de entidades con las que el producto puede coexistir B = Número de entidades en el entorno de operación que requieren de coexistencia Dónde: B > 0	0	1	Si	6,00	6,00		1,00	10,00	10,00	10,00	0,50	
	Capacidad de	Capacidad de pistas de auditoría	¿Los usuarios pueden identificar fácilmente la operación específica que causó el fallo?	Contar el número de datos realmente grabadas durante la operación y el número de datos previstos a grabarse para controlar el estado del sistema durante la operación	Interna/Externa	X = A / B A = Número de datos realmente grabadas durante la operación B = Número de datos previstos a grabarse para controlar el estado del sistema durante la operación Dónde: B > 0	0	1	No						8			

Mantenibilidad	ser analizado	Diagnóstico de funciones suficientes	¿Hasta qué punto las funciones de diagnóstico están preparadas o hasta qué punto funcionan para el análisis causal?	Contar el número de funciones de diagnóstico implementadas y contar el número de funciones de diagnóstico requeridas en la especificación de requerimientos	Interna/Externa	X = A/B A = Número de funciones de diagnóstico implementadas B = Número de funciones de diagnóstico requeridas en la especificación de requerimientos Dónde: B > 0	0	1	Si	1,00	1,00		1,00	10,00		10,00	2,00	
	Capacidad de ser probado	Capacidad de probar funciones y componentes	¿Probee herramientas que permitan probar funciones y componentes ?	Contar el tipo de pruebas que permite realizar a funciones y compoentes	Externa	X = A/B A = Número de tipo de pruebas que permite realizar B = Número de tipo de pruebas requerido Dónde: B > 0	0	1	Si	3,00	3,00		1,00	10,00	2,00			
Portabilidad	Capacidad de ser Instalado	Eficiencia en el tiempo de instalación	¿Cuánto tiempo es requerido para realizar una instalación?	Contar el tiempo total transcurrido al instalar el sistema y contar el número de reintentos al instalar el sistema	Externa	X = A/T A = Número de reintentos al instalar el sistema T = Tiempo total transcurrido al instalar el sistema Dónde: T > 0	>=10min	0min	**No**					0,00	5	5,00	0,25	
		Facilidad de instalación	¿Puede fácilmente el usuario o el desarrollador instalar el software en un entorno operacional?	Contar cuantas formas de instalación ofrece el producto	Externa	X = A/B A = Número de métodos de instalación B = Número total de métodos de instalación Dónde: B > 0	0	1	Si	2,00	2,00		1,00	10,00				
6	**3**	**16**							**10**									

Anexo E: Especificación de requerimientos de software

para

SVC (Sistema de control de ventas)

Versión 1.0

Elaborado por Javier Paucar

Tabla de contenidos

1. Introducción

1.1 Propósito

El sistema SCV permite a los usuarios registrar las ventas que realicen y controlar los ingresos de dinero por medio de una interfaz de usuario web. Además del registró de ventas el sistema también dará la posibilidad de llevar el control de existencias de los productos vendidos.

El propósito de este documento es describir la funcionalidad de sistema SCV que se va a desarrollar.

1.2 Alcance

El alcance del proyecto consiste en desarrollar el sistema SCV con los siguientes módulos:

- Registro de los locales que maneja la panificadora.
- Registro de usuarios del sistema.
- Manejo de inventario de cada local.
- Registro de ventas realizadas en cada local.
- Módulo de seguimiento técnico para el registro de ventas.
- Reportes de las ventas realizadas.

1.3 Referencias

– Laravel - The PHP Framework For Web Artisans. https://laravel.com/docs/5.7

– Empezando – React. https://es.reactjs.org/docs/getting-started.html

– Introduction. Vuejs.org. https://v2.vuejs.org/v2/guide/

– Inspinia - Responsive Admin Template by WebAppLayers. WrapBootstrap. https://wrapbootstrap.com/theme/inspinia-responsive-admin-template-WB0R5L90S

2. Descripción General

2.1 Perspectiva del producto

El producto es una aplicación web que mediante un navegador web va a permitir a los usuarios, según sus roles, registrar las ventas de los diferentes productos y llevar un control de las existencias de los productos vendidos.

Para el acceso al sistema se usa como mecanismo de autenticación un identificador y una contraseña para cada usuario que será asignados por el administrador del sistema. Para el control de acceso a los diferentes módulos del sistema se asignan permisos según los siguientes roles: administrador, supervisor y vendedor.

2.2 Características del Producto

En esta sección se listan las principales características del sistema:

- Seguridad
  - Autenticación: controlar el acceso de cada usuario al sistema mediante un identificador y contraseña.
  - Control de acceso: controlar el acceso a los diferentes recursos del sistema mediante permisos de acceso basados en roles.
  - Llevar un registro de los diferentes movimientos que el usuario realiza en el sistema.
- Gestión de usuarios: permitir registrar nuevos usuarios al sistema, modificar información de usuarios existentes y dar de baja usuarios que ya no deban tener acceso.
- Gestión de sucursales: permitir registrar nuevas sucursales del negocio en el sistema, modificar información de las sucursales existentes y dar de baja a sucursales, además se pueden asignar usuarios a cada sucursal.
- Gestión de categorías de productos: permitir registrar nuevas categorías de productos, modificar información de categorías de productos ya registradas y dar de baja categorías de productos.
- Gestión de productos: permitir registrar nuevos productos, modificar información de productos ya registrados y dar de baja productos.
- Gestión de clientes: permitir registrar nuevos clientes, modificar información de clientes ya registrados y dar de baja clientes.
- Gestión de caja:
  - Abrir caja, con un monto fijo.
  - Registrar ventas en la caja abierta.
  - Permitir registrar ingresos de dinero en caja, sin necesidad de venta.
  - Permitir registrar retiros de dinero en caja.
  - Cerrar caja, el usuario debe informar la cantidad de dinero que entrega al hacer el cierre
  - Llevar un registro de todas las transacciones realizadas en la caja.
- Gestión de inventario:
  - Permitir establecer y modificar las cantidades existentes de los productos.
  - Ajustar las cantidades de los productos automáticamente al registrar las ventas.
  - Llevar un registro de todas las transacciones realizadas en el inventario.
- Reportes: generar reportes de ventas, inventario y auditoria de movimientos del usuario.

2.3 Clases de Usuarios y Sus Características

Hay tres tipos de usuarios que interactúan con el sistema estos son: usuario administrador, usuarios supervisor y usuarios vendedor.

- Administrador: el usuario administrador es el encargado de gestionar el registro de usuarios y sucursales, además se encarga de otorgar los permisos a los usuarios y asignar usuarios a las sucursales, también tiene acceso a los reportes de transacciones de caja, transacciones de inventario y registro de movimientos del sistema.
- Supervisor: el usuarios supervisor es el encargado de gestionar los registros de clientes, productos y categorías de productos, este usuario puede modificar el inventario de los productos además tiene acceso a reportes de caja y reportes de inventario.
- Vendedor: el usuario vendedor es el encargado de registrar las ventas también puede gestionar el registro de clientes, tiene acceso a reportes de ventas.

2.4 Entorno Operativo

Para que el sistema opere de forma correcta debe ejecutarse en un equipo donde se encuentren instaladas las siguientes herramientas:

- Sistema Operativo Windows o Linux
- PHP v7.1.3 o superior
- Google Chrome v51 o superior
- MariaDB server v10.4
- Servidor Apache v2.4

Se recomienda usar el paquete de herramientas XAMPP Versión: 7.4.33 para Windows o Linux que ya incluye todas las herramientas descritas anteriormente.

Además de las herramientas anteriores para el entorno de desarrollo se debe instalar:

- Composer v1.7.3 o superior
- Node v6.17.1

2.5 Restricciones de Diseño e Implementación

- El back-end será desarrollado usando el framework de PHP Laravel
- Para la persistencia de datos se usará el motor de base de datos relacional MariaDB.
- El front-end de la aplicación será desarrollado usando una combinación entre el motor de plantillas, de Laravel, Blade y React o Vue.js, no será un SPA.
- Para el diseño de la interfaz gráfica se usará la plantilla INSPINIA v2.8.
- Para el cifrado de las contraseñas de los usuarios se usa la función de cifrado bcrypt proporcionada por Laravel.

2.6 Supuestos y Dependencias

- Para que poder usar el sistema el usuario debe tener un equipo donde estén instaladas las herramientas listadas en la sección Entorno Operativo o contratar un servicio de hosting que proporcione este entorno.
- Si se quiere manejar varias sucursales en un mismo sistema, el usuario debe contratar un hosting o disponer de un servidor y una IP pública que le permita mantener conectadas las sucursales.

3. Requerimientos Funcionales

3.1 Autenticar usuarios

Caso de uso 1: Autenticar usuarios
Contexto de uso: Los usuarios que ingresan al sistema deben tener credenciales
Ámbito: Sistema SVC
Actor: Usuario del sistema
Escenario principal:
1. El usuario accede a la pantalla de ingreso al sistema
2. El usuario ingresa su identificador y contraseña en el formulario de acceso
3. Las credenciales son correctas
4. El usuario ingresa al sistema
Extensiones:
3a. Las credenciales no son correctas

4a. El usuario permanece en la pantalla de ingreso al sistema.

3.2 Gestionar usuarios y sucursales

Caso de uso 2: Gestionar usuarios y sucursales
Contexto de uso: Permitir el registro y modificación de información de usuarios y sucursales
Ámbito: Sistema SCV
Actor: Administrador
Escenario principal:
1. Desde el menú principal ingresar al módulo de usuarios/sucursales
2. Mostrar la pantalla principal del módulo
3. Dar clic en el botón "Nuevo"
4. Mostrar formulario de ingreso de datos
5. Llenar los datos del usuario/sucursal en el formulario
6. Guardar los datos haciendo clic en el botón "Guardar"
7. Regresar a la pantalla principal del módulo
Extensiones:
3a.1. Seleccionar un registro de la lista de usuarios/sucursales
3a.2. Desplegar menú
3a.3. Seleccionar la opción editar

3.3 Gestionar categorías de producto, productos y clientes

Caso de uso 3: Gestionar categorías de producto, productos y clientes
Contexto de uso: Permitir el registro y modificación de información de categorías de producto, productos y clientes
Ámbito: Sistema SCV
Actor: Supervisor
Escenario principal:
1. Desde el menú principal ingresar al módulo de categorías de producto/productos/clientes
2. Mostrar la pantalla principal del módulo
3. Dar clic en el botón "Nuevo"
4. Mostrar formulario de ingreso de datos
5. Llenar los datos de la categoría de producto/producto/cliente
6. Guardar los datos haciendo clic en el botón "Guardar"
7. Regresar a la pantalla principal del módulo
Extensiones:
3a.1. Seleccionar un registro de la lista de producto/producto/cliente
3a.2. Desplegar menú
3a.3. Seleccionar la opción editar

3.4 Gestiona clientes

Caso de uso 4: Gestiona clientes
Contexto de uso: Permitir el registro y modificación de información de clientes
Ámbito: Sistema SCV
Actor: Vendedor
Escenario principal:
1. Desde el menú principal ingresar al módulo de clientes
2. Mostrar la pantalla principal del módulo de clientes
3. Dar clic en el botón “Nuevo”
4. Mostrar formulario de ingreso de datos
5. Llenar los datos de productos
6. Guardar los datos del producto dando clic en el botón “Guardar”
7. Regresar a la pantalla principal del módulo de clientes
Extensiones:
3a.1. Seleccionar un cliente de la lista
3a.2. Desplegar menú
3a.3. Seleccionar la opción editar

3.5 Dar de baja a usuarios y sucursales

Caso de uso 5: Dar de baja a usuarios y sucursales
Contexto de uso: Permitir dar de baja a usuarios y sucursales
Ámbito: Sistema SCV
Actor: Administrador
Escenario principal:
1. Desde el menú principal ingresar al módulo de usuarios/sucursales
2. Mostrar la pantalla principal del módulo
3. Seleccionar un usuario/cliente de la lista
4. Desplegar menú
5. Seleccionar “Desactivar”
6. Mostrar diálogo de confirmación
7. Confirmar desactivación
Extensiones:
7a. Cancelar desactivación

3.6 Dar de baja categorías de producto, productos y clientes

Caso de uso 6: Dar de baja categorías de producto, productos y clientes
Contexto de uso: Permitir dar de baja categorías de producto, productos y clientes
Ámbito: Sistema SCV
Actor: Supervisor
Escenario principal:
1. Desde el menú principal ingresar al módulo de categorías de producto/productos/clientes
2. Mostrar la pantalla principal del módulo

3. Seleccionar un registro de la lista de categorías de producto/productos/clientes
4. Desplegar menú
5. Seleccionar “Desactivar”
6. Mostrar diálogo de confirmación
7. Confirmar desactivación

Extensiones:

7a. Cancelar desactivación

3.7 Dar de baja clientes

Caso de uso 7: Dar de baja clientes

Contexto de uso: Permitir dar de baja clientes

Ámbito: Sistema SCV

Actor: Vendedor

Escenario principal:

1. Desde el menú principal ingresar al módulo de clientes
2. Mostrar la pantalla principal del módulo de clientes
3. Seleccionar un cliente de la lista de clientes
4. Desplegar menú
5. Seleccionar “Desactivar”
6. Mostrar diálogo de confirmación
7. Confirmar desactivación

Extensiones:

7a. Cancelar desactivación

3.8 Añadir usuario a sucursal

Caso de uso 8: Añadir usuario a sucursal

Contexto de uso: Permitir añadir usuarios a las sucursales

Ámbito: Sistema SCV

Actor: Administrador

Escenario principal:

1. Desde el menú principal ingresar al módulo de sucursales
2. Mostrar la pantalla principal del módulo de sucursales
3. Seleccionar una sucursal de la lista de sucursales
4. Desplegar menú
5. Seleccionar “Gestionar Usuarios”
6. Mostrar pantalla de gestión de usurarios de sucursales
7. Dar clic en el botón “Añadir Usuario”
8. Mostar dialogó para buscar usuario
9. Buscar usuarios y seleccionarlo
10. Dar clic en el botón Añadir

3.9 Quitar usuario de sucursal

Caso de uso 9: Quitar usuario de sucursal
Contexto de uso: Permitir quitar usuarios de las sucursales
Ámbito: Sistema SCV
Actor: Administrador
Escenario principal:
1. Desde el menú principal ingresar al módulo de sucursales
2. Mostrar la pantalla principal del módulo de sucursales
3. Seleccionar una sucursal de la lista de sucursales
4. Desplegar menú
5. Seleccionar “Gestionar Usuarios”
6. Mostrar pantalla de gestión de usurarios de sucursales
7. Seleccionar un usuario de la lista y dar clic en el botón “Quitar”
8. Mostar dialogó de confirmación
9. Confirmar
Extensiones:
11a. Cancelar

3.10 Ajustar cantidades de producto por sucursal

Caso de uso 10: Ajustar cantidades de producto por sucursal
Contexto de uso: Permitir ajustar las cantidades de productos inventariables por sucursal
Ámbito: Sistema SCV
Actor: Supervisor
Condición previa: Los productos deben ser inventariables
Escenario principal:
1. Desde el menú principal ingresar al módulo de inventario
2. Mostrar pantalla principal del módulo de inventario
3. Seleccionar la sucursal
4. Seleccionar operación: Agregar, Quitar o Ajustar
5. Buscar producto
6. Indicar cantidad para la operación y el costo del producto
7. Añadir operación a la lista
8. Dar clic en el botón “Guardar Inventario”

3.11 Abrir caja

Caso de uso 11: Abrir caja
Contexto de uso: Realizar en el estado inicial de la caja cuando el vendedor comienza su jornada
Ámbito: Sistema SCV
Actor: Vendedor
Escenario principal:
1. Desde el menú principal ingresar al módulo de ventas
2. Mostrar pantalla del módulo de ventas

3. Seleccionar sucursal
4. Indicar monto inicial de apertura
5. Dar clic en el botón “Abrir caja”

3.12 Registrar venta

Caso de uso 12: Registrar venta
Contexto de uso: Registrar ventas realizadas
Ámbito: Sistema SCV
Actor: Vendedor
Condición previa: Abrir caja
Escenario principal:
1. Buscar producto
2. Indicar cantidad
3. Agregar producto a la lista
4. Dar clic en el botón “Cobrar”
5. Mostar ventana modal de pago
6. Buscar cliente (opcional)
7. Ingresar monto recibido
8. Ingresar observación (opcional)
9. Dar clic en “Aceptar” para guardar venta
Extensiones:
9a. Dar clic en “Cancelar” para seguir añadiendo productos

3.13 Hacer depósito de dinero en caja

Caso de uso 13: Hacer depósito de dinero en caja
Contexto de uso: Registrar depósito de dinero en caja
Ámbito: Sistema SCV
Actor: Vendedor
Condición previa: Abrir caja
Escenario principal:
1. Dar clic en el botón “Depositar”
2. Mostrar ventana modal de deposito
3. Ingresar el monto del deposito
4. Ingresar el motivo del deposito
5. Guardar deposito dando clic en “Aceptar”
Extensiones:
5a. Cancelar deposito dando clic en “Cancelar”

3.14 Hacer retiro de dinero de caja

Caso de uso 14: Hacer retiro de dinero de caja
Contexto de uso: Registrar retiro de dinero de caja
Ámbito: Sistema SCV

Actor: Vendedor
Condición previa: Abrir caja
Escenario principal:
1. Dar clic en el botón “Retirar”
2. Mostrar ventana modal de Retiro
3. Ingresar el monto del retiro
4. Ingresar el motivo del retiro
5. Guardar deposito dando clic en “Aceptar”
Extensiones:
5a. Cancelar deposito dando clic en “Cancelar”

3.15 Cerrar caja

Caso de uso 13: Cerrar caja
Contexto de uso: Registrar el cierre de caja al finalizar el turno del vendedor, indicando los montos finales de las ventas
Ámbito: Sistema SCV
Actor: Vendedor
Condición previa: Abrir caja
Escenario principal:
1. Dar clic en el botón “Cerrar caja”
2. Mostrar ventana modal del cierre de caja
3. Ingresar el monto de dinero que el vendedor tiene en caja al final de la jornada
5. Ingresar observación (opcional)
6. Cerrar caja dando clic en “Aceptar”
Extensiones:
5a. Cancelar cierre dando clic en “Cancelar”

4. Requisitos de Interfaz Externa

4.1 Interfaz de Usuario

En la figura 4.1 se muestra la pantalla de acceso al sistema, esta es la pantalla inicial del sistema donde el usuario debe ingresar sus credenciales para acceder al sistema.

Figura 4.1. Pantalla de acceso al sistema.

La interfaz de usuario del sistema constará de 3 partes; un menú en la parte superior, un menú lateral y el área principal de trabajo, en la figura 4.2 se puede ver con estará distribuida la interfaz.

Figura 4.2. Distribución general de la interfaz del sistema.

Para la gestión de los diferentes CRUDs del sistema la distribución de la pantalla va a constar de un botón que permita registra un nuevo registró ya sea de usuarios, productos, clientes, etc., una tabla donde se listaran los registros ingresados, cada fila de la tabla tendrá un menú con las opciones para editar y desactivar el registro, también habrá una sección para poder filtrar los registros por diferentes criterios y paginadores para poder navegar por la lista de registros, en la figura 4.3 se muestra un ejemplo de la pantalla.

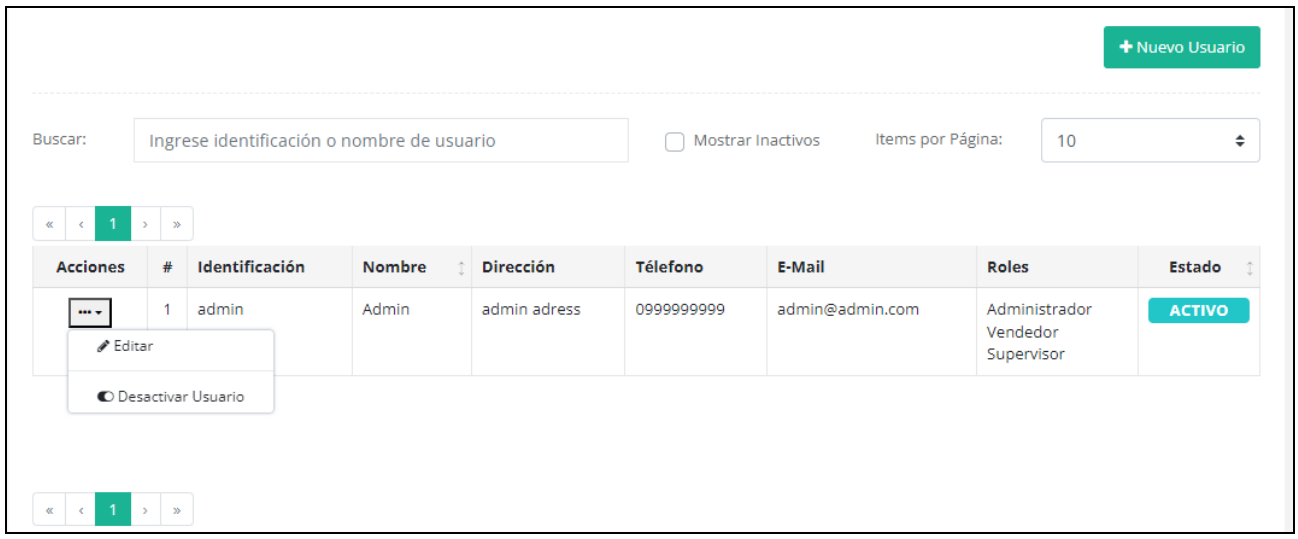


Figura 4.3. Ejemplo pantalla para gestión de CRUDs.

Para el registro de información se presentarán formularios con diferentes campos para el ingreso de información, al pie del formulario se dispondrán dos botones uno para guardar la información y otro para cancelar el proceso, en la figura 4.4 se muestra un ejemplo de formulario.

Figura 4.4. Ejemplo formulario de registro de información.

Para la apertura de caja la pantalla va a constar de un capo donde se seleccionará la sucursal, un campo para indicar el monto inicial y un botón para confirmar la apertura de caja, como se muestra en la figura 4.5.

Figura 4.5. Pantalla para apertura de caja.

La pantalla para registro de ventas contará con una sección para selección de producto, una tabla donde se irán listando los productos seleccionados, y una sección donde se encuentra los controles para realizar las acciones, como el cobro y cancelación de una venta, el depósito y retiro de dinero y el cierre de caja, en esta sección también se mostrará el total de la venta. En la figura 4.6 se muestra el diseño propuesto.

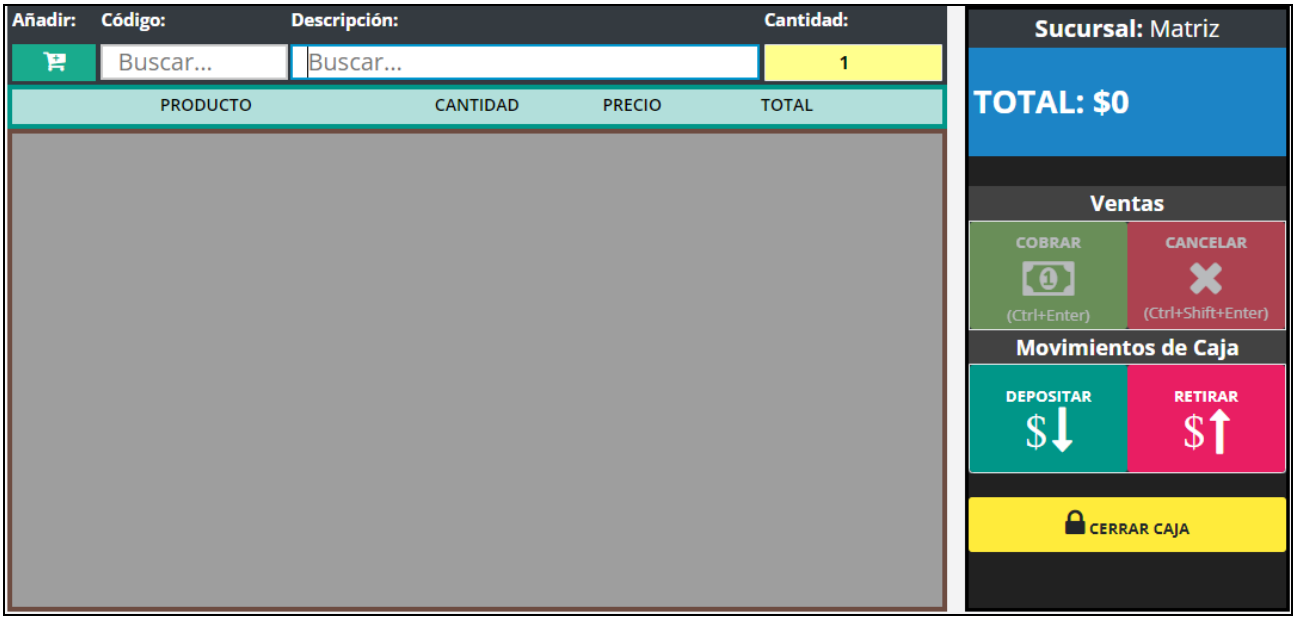


Figura 4.6. Pantalla de registro de venta.

La pantalla para registrar el pago de la venta será una ventana modal que constará de un campo para el cliente, un campo donde se ingresará la cantidad de dinero que entregó el cliente y un campo para indicar una observación, además se muestra otra información como el monto que se debe cobrar y si hay cambio que entregar. En la figura 4.7 se muestra el diseño de esta pantalla.

Figura 4.7. Diseño pantalla de registro de pago de una venta.

Para el registro de depósitos y retiros de dinero se presentarán ventanas modales donde el usuario dispondrá de un campo para indicar el monto de dinero y el motivo por el cual se realiza la acción, al pie de la ventana modal estarán colocados un botón para aceptar la acción y otro para cancelar la acción. En la figura 4.7 se muestra el diseño de las ventanas modales.

Figura 4.7. Diseño pantalla para el registro de retiros y depósitos de dinero.

Para registra y ajustar las cantidades de productos se presentará una pantalla donde se permita escoger la sucursal, el tipo de operación y el producto, se dispondrá de campos para indicar la cantidad y costo del producto seleccionado, un botón para ir agregado diferentes productos, una tabla donde se listarán los productos y operaciones seleccionados, se dispondrá de un paginador para navegar por la lista de productos seleccionados, al pie de la pantalla se tendrá un botón para guardar las operaciones seleccionadas. En la figura 4.8 se muestra el diseño de la pantalla.

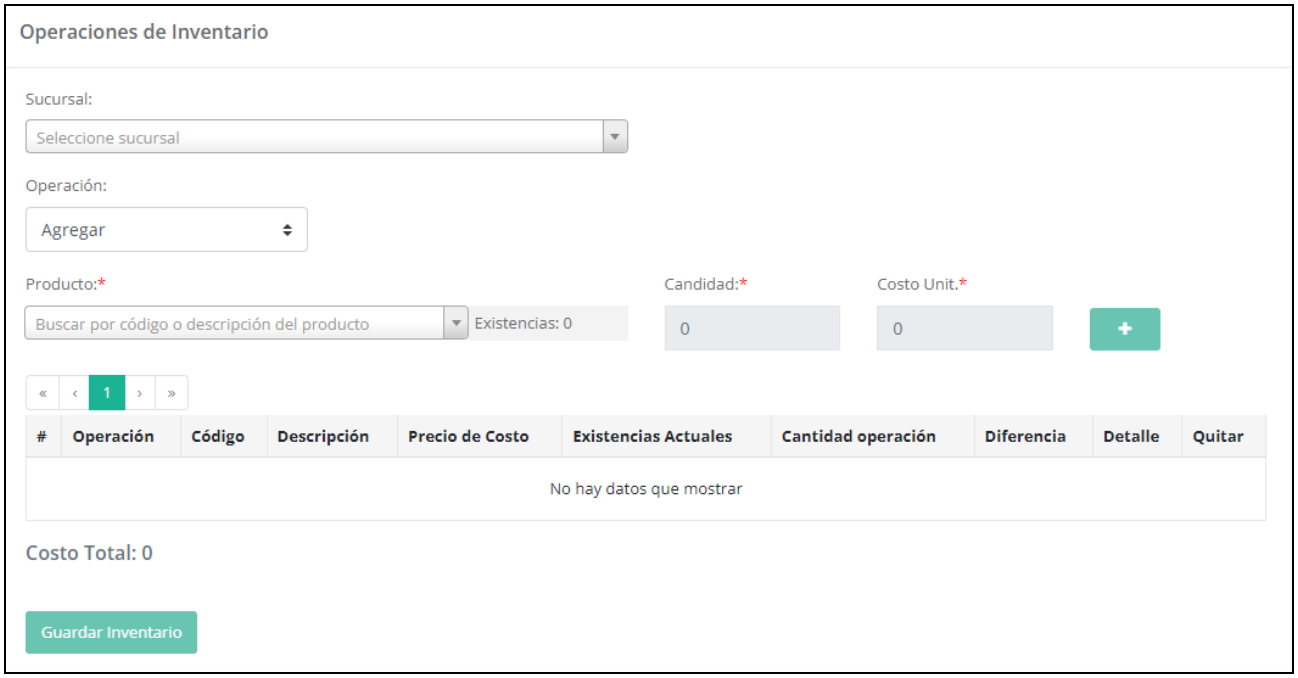


Figura 4.8. Diseño de pantalla para registrar y ajustar cantidades de producto por sucursal.

4.2 Interfaces de Software

El sistema SCV es una aplicación web desarrollada usando las tecnologías PHP, JavaScript, HTML y CSS, la aplicación será servida usando el servidor HTTP Apache. Para el almacenamiento de información se usará el motor de base de datos MariaDB. La aplicación usará como cliente el navegador web Google Chrome que se comunica con el servidor web mediante peticiones HTTP. Toda esta infraestructura se ejecuta sobre el sistema operativo Windows o Linux.

5. Otros Requerimientos No Funcionales

5.1 Requerimientos de seguridad

La autenticación de los usuarios se controlará mediante el mecanismo de usuario y contraseña, el administrador será el encargado de asignar las credenciales a cada usuario registrado.

La autorización se manejará usando el mecanismo de control de acceso basado en roles, el sistema tendrá tres roles, Administrador, Supervisor y Vendedor, cada rol tendrá permisos específicos para las diferentes áreas del sistema.

La trazabilidad de las acciones de los usuarios se la manejará guardando registros de las acciones más relevantes que el usuario realice en el sistema.

6. Apéndice A: Glosario de términos

CRUD: Acrónimo de Create (Crear), Read (Leer), Update (Actualizar) y Delete (Borrar) que se refieren a las operaciones que se hacen sobre la información almacenada.
HTTP: Hypertext Transfer Protocol es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML.
HTML: HyperText Markup Language hace referencia al lenguaje de marcado para la elaboración de páginas web.
CSS: Cascading Style Sheets s el lenguaje de estilos utilizado para describir la presentación de documentos HTML.
SPA: single-page application es un tipo de aplicación web que ejecuta todo su contenido en una sola página.

7. Apéndice B: Modelos de Análisis

Diagrama de proceso de registro de venta.

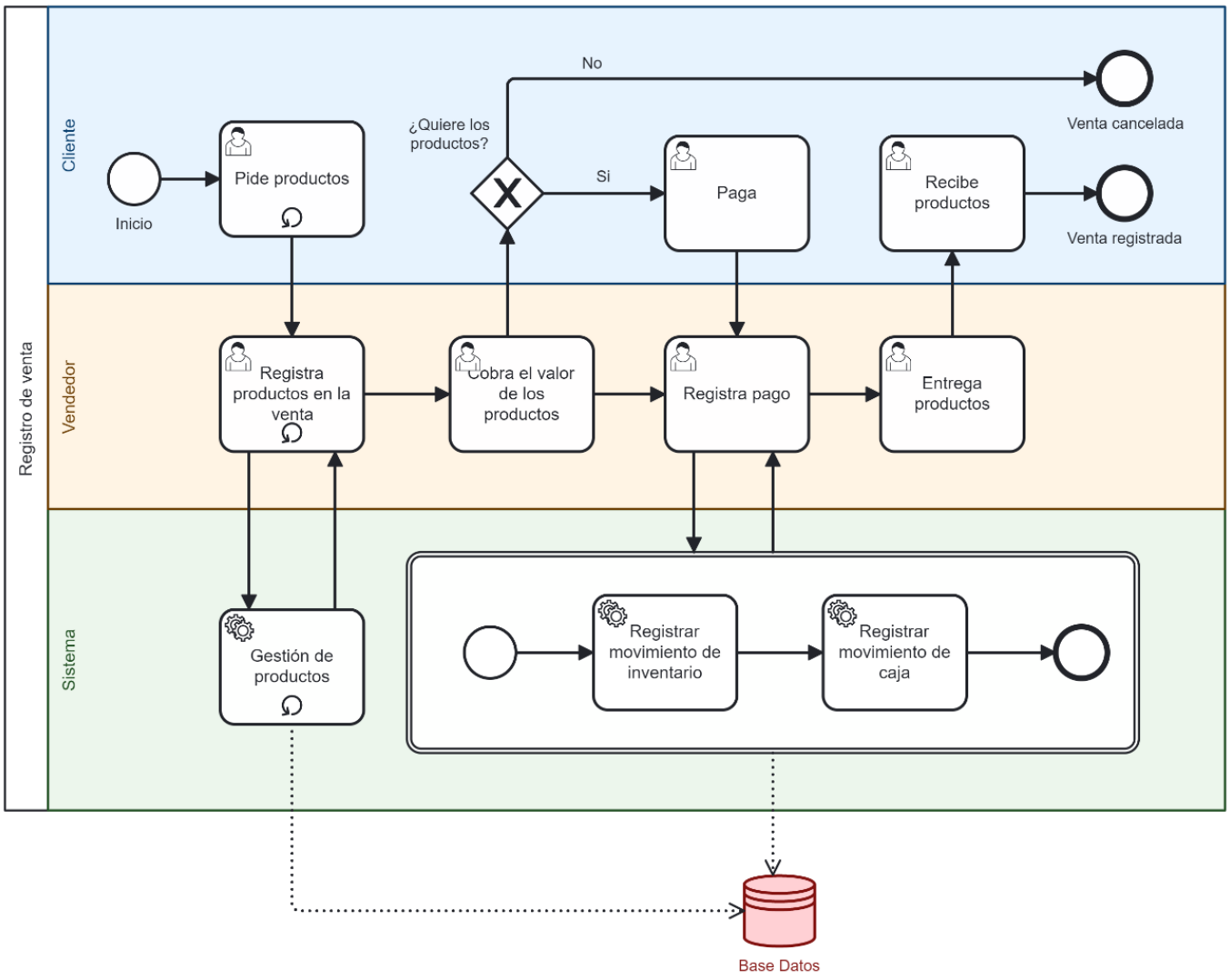


Diagrama de proceso de manejo de inventario de productos.

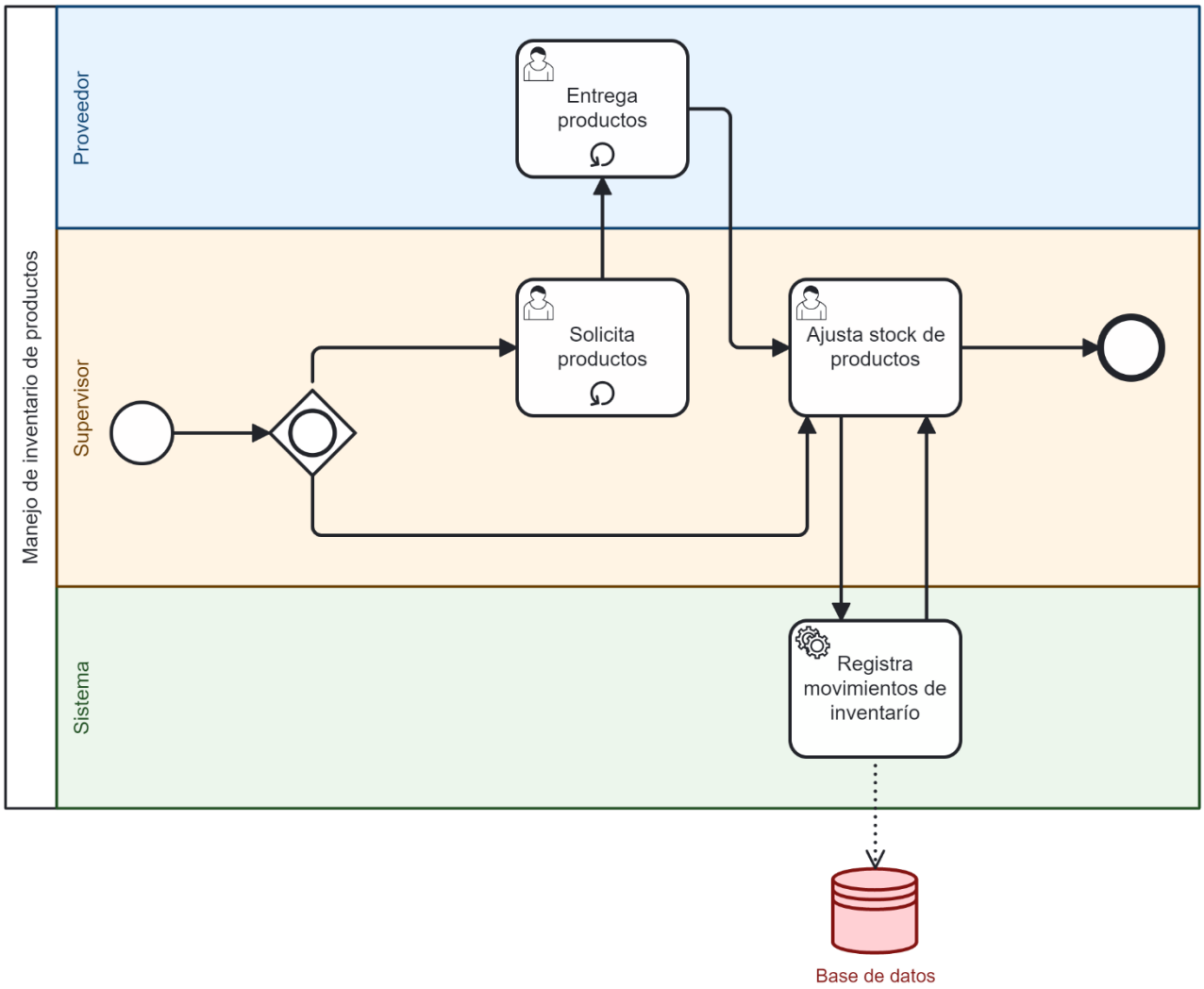

Manejo de inventario de productos
Proveedor
Supervisor
Sistema
Entrega productos
Solicita productos
Ajusta stock de productos
Registra movimientos de inventario
Base de datos

Diagrama de casos de uso del rol Supervisor.

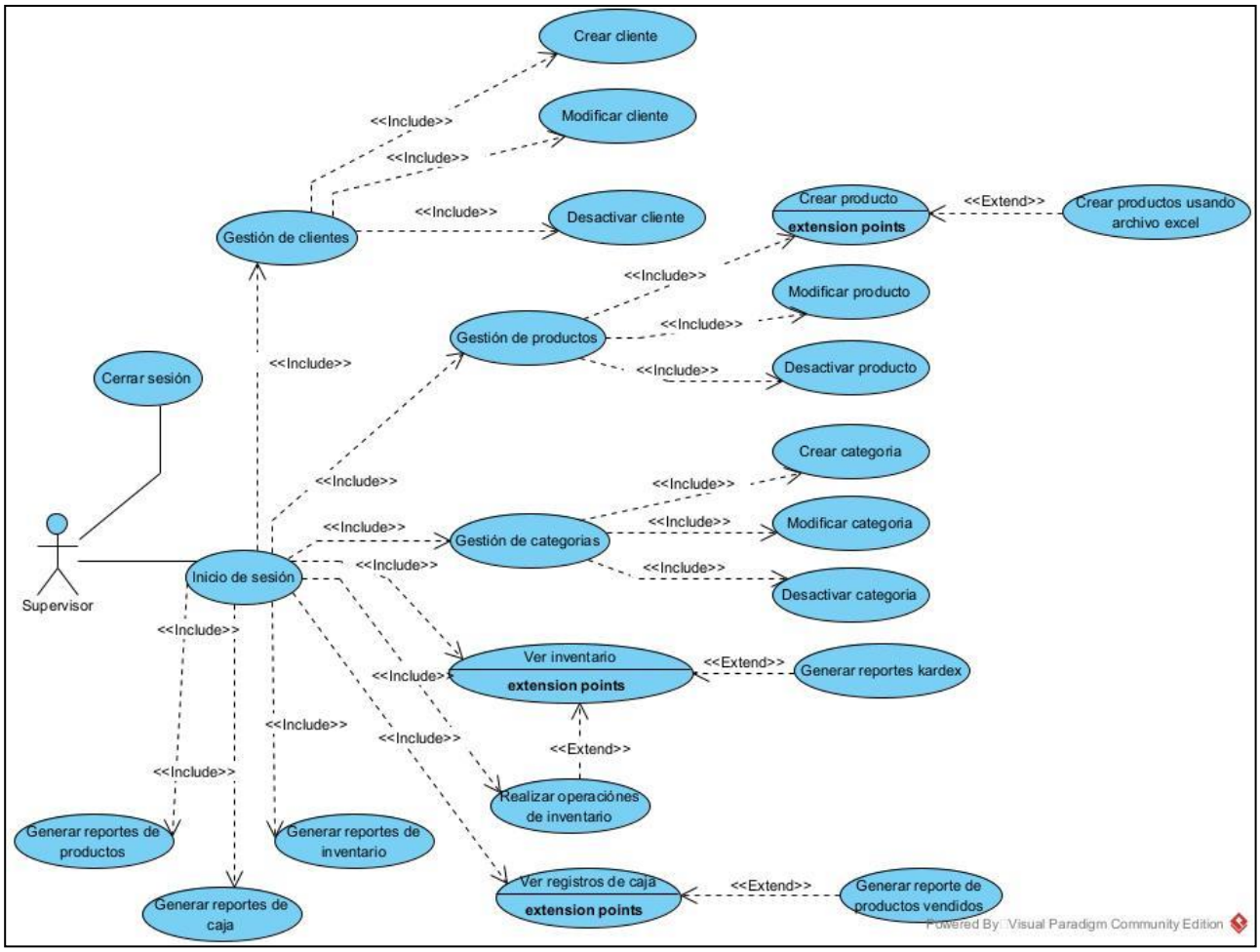


Diagrama de casos de uso del rol Administrador.

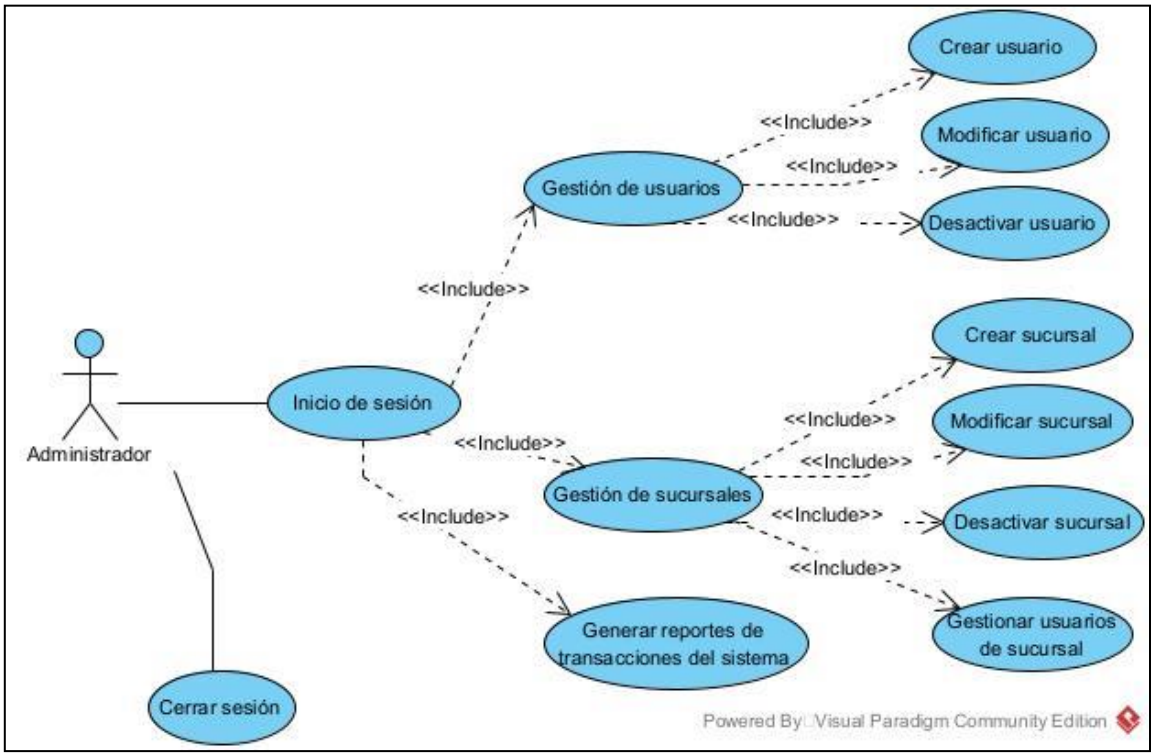
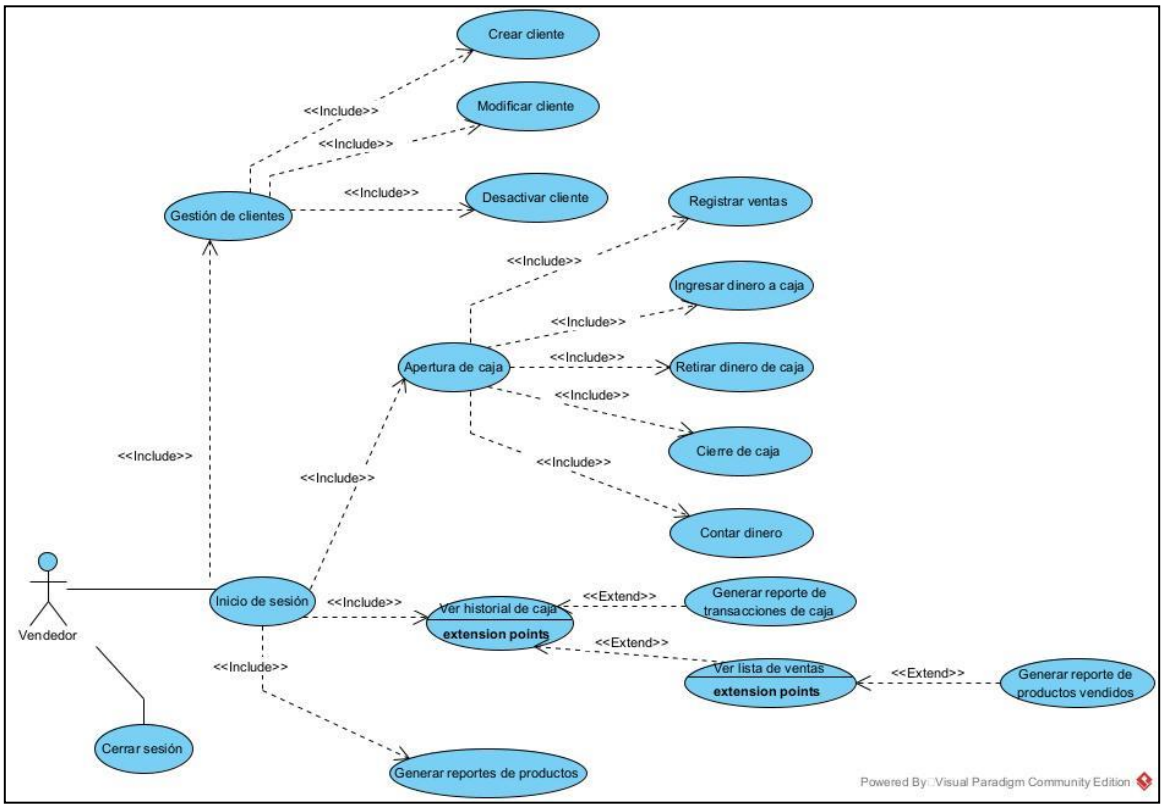
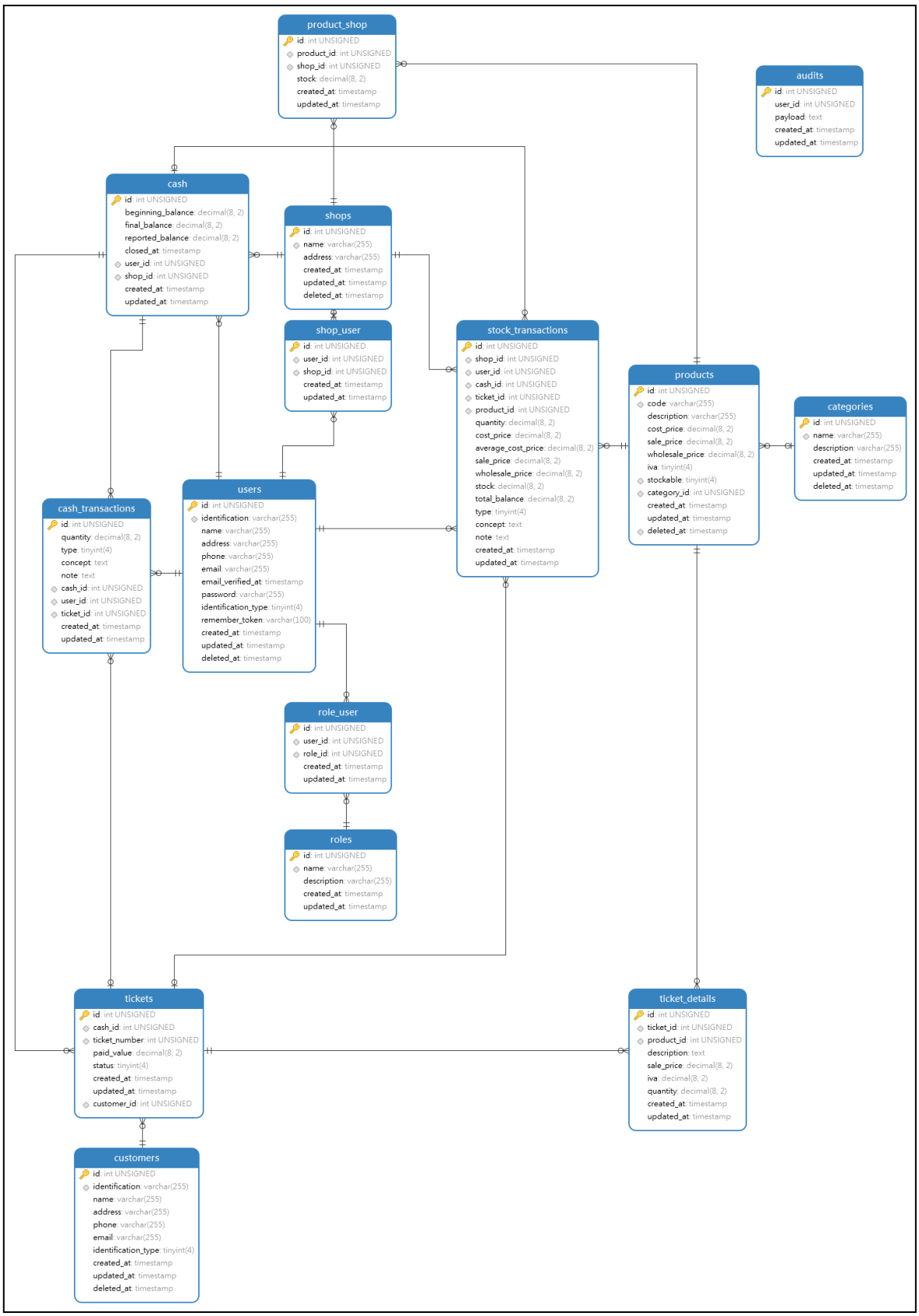


Diagrama de casos de uso del rol Vendedor.



Modelo Entidad Relación



Representación de Arquitectura en Capas

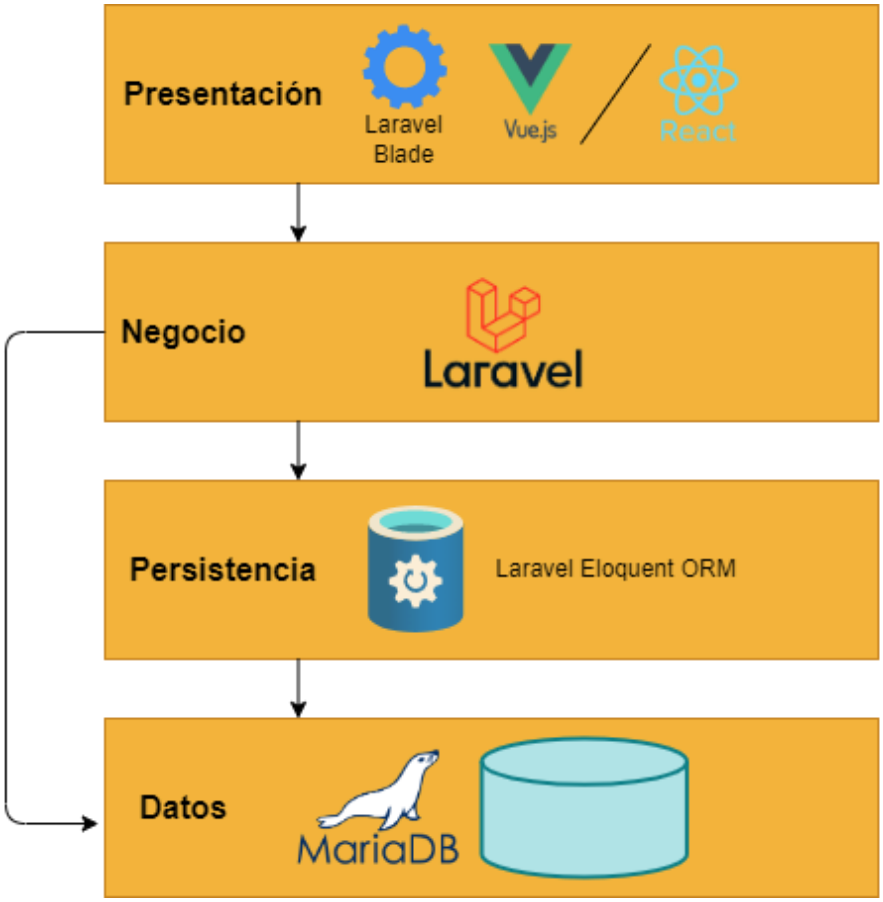


Diagrama de componentes.

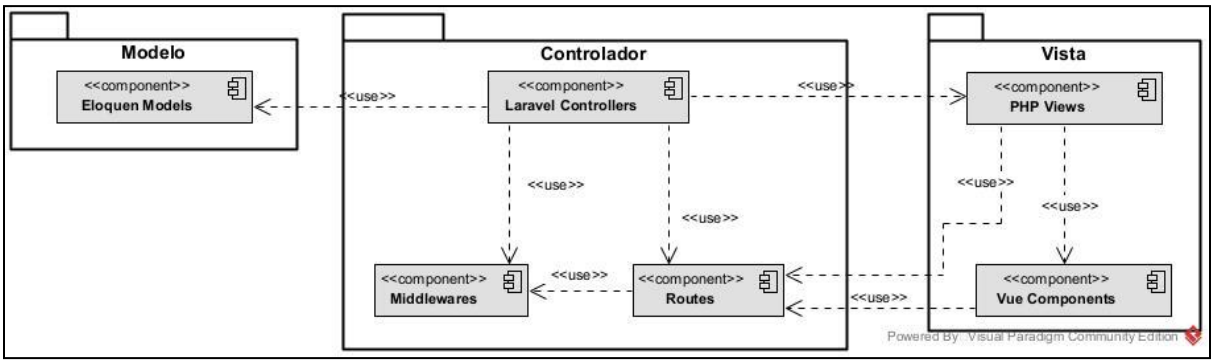


Diagrama de despliegue