PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

CARRERA DE: SISTEMAS DE INFORMACION

Trabajo de Titulación

Tema: Desarrollo de un prototipo funcional de una aplicación móvil para la gestión de inventario de una empresa textil.

AUTOR:

JIMMY DAMIAN ALVARADO RAMIREZ

QUITO DM, 2024

Tabla de Contenidos

1. Marco de Referencia 8

1.1 Justificación 8

1.3 Objetivos 11

1.3.1 Objetivo General 11

1.3.2 Objetivos Específicos 11

1.4 Antecedentes 12

1.5 Alcance 13

2. Marco Teórico 15

2.1. Aplicación Móvil 15

2.1.1 ¿Qué es una Aplicación Móvil? 15

2.2 Beneficios de la Automatización del Inventario 16

2.2.1 Ventajas de una Aplicación Móvil para la Gestión de Inventarios 18

2.3 Marco de trabajo para desarrollo móvil 18

2.3.1 React Native 19

2.3.2 Flutter 20

2.3.3 Selección del Marco de Trabajo para el Desarrollo Móvil 20

2.3 Base de Datos 23

2.3.1 ¿Qué es una Base de Datos Relacional? 23

2.3.2¿Qué es una Base de Datos no Relacional? 24

2.3.3 Diferencias entre Base de Datos Relacionales y no Relacionales 24

2.3.4 Selección del Modelo de Base de Datos 25

2.4 ¿Qué es Backend? 26

2.4.1 Node.js 26

2.4.2 Laravel 26

2.4.3 Diferencias entre Node.js y Laravel 27

2.5 Arquitectura de la Aplicación Móvil 29

2.5.1 ¿Qué es una Arquitectura Cliente Servidor? 29

2.5.2 Ventajas de la Arquitectura Cliente Servidor 29

2.6 Casos de Prueba de Aceptación 31

3.1 ¿Qué es una Metodología en el Desarrollo de Software? 32

3.2 Tipos de Metodologías 32

3.2.1 Metodología Tradicional 32

3.2.2 Metodología Ágil 33

3.2.2.1 Scrum 33

3.2.2.2 Extreme programming (XP) 34

3.3 Selección de Metodología 34

3.3.1 Aplicación de Scrum 35

3.3.1.1. Definición de Roles en Scrum 35

3.3.1.2 Gestión de Riesgos y Desafíos 36

4. Desarrollo de la Aplicación 36

4.1 Historias de Usuario 36

4.2 Iteración 1 37

4.2.1 Historias de Usuario 37

4.2.2 Tareas 38

4.2.3 Pruebas 39

4.3 Iteración 2 40

4.3.1 Historias de Usuario 40

4.3.2 Tareas 41

4.3.3 Pruebas 41

4.4 Iteración 3 43

4.4.1 Historias de Usuario 43

4.4.2 Tareas 43

4.4.3 Pruebas 44

4.5 Iteración 4 45

4.5.1 Historias de Usuario 45

4.5.2 Tareas 46

4.5.3 Pruebas 46

4.6 Iteración 5 47

4.6.1 Historias de Usuario 47

4.6.2 Tareas 48

4.6.3 Pruebas 49

5. Conclusiones y Recomendaciones 50

5.1 Conclusiones 50

5.2 Recomendaciones 51

Referencias 52

Anexo A: Pantalla de Inicio de Sesión. 56

Anexo B: Pantalla de Perfil 56

Anexo C: Pantalla de Usuarios. 57

Anexo D: Pantalla de Usuarios (Formulario para creación de Usuario) 57

Anexo E: Pantalla Agregar Productos 58

Anexo F: Pantalla Registro de Acciones de los usuarios 58

Anexo G: Pantalla Realizar Ventas 59

Anexo H: Pantalla Historial de Ventas 59

Tabla 1..... 23

Tabla 2..... 37

Tabla 3..... 38

Tabla 4..... 39

Tabla 5..... 40

Tabla 6..... 41

Tabla 7..... 41

Tabla 8..... 43

Tabla 9..... 43

Tabla 10..... 44

Tabla 11..... 45

Tabla 12..... 46

Tabla 13..... 46

Tabla 14..... 47

Tabla 15..... 48

Tabla 16..... 49

Ilustración 1 Arquitectura Cliente Servidor de la Aplicación 31

CAPÌTULO I: INTRODUCCIÓN

1. Marco de Referencia

1.1 Justificación

El desarrollo de un prototipo de aplicación móvil para la gestión de inventario como una solución clave para resolver los problemas actuales que enfrenta la empresa. En la actualidad, los procesos de control de inventarios se realizan de manera manual, lo que incrementa significativamente el riesgo de errores humanos, tales como la falta de productos o registros incorrectos de entradas y salidas. Estos errores no solo afectan la disponibilidad de los productos, sino que también impactan a la atención al cliente y la eficiencia de las operaciones diarias.

El foco principal de prototipo será el control de stock de productos ya que esta es la funcionalidad más crítica para la empresa en este momento. Tener un control preciso y actualizado en tiempo real del inventario permitirá a los empleados gestionar de manera más eficiente las existencias evitando tanto la sobrecarga de productos como la escasez de estos lo cual es esencial para mantener a una población fluida y satisfacer la demanda de los clientes.

Además, al contar con un sistema de inventario actualizado en tiempo real los empleados podrán tomar decisiones informadas sobre la reposición del producto lo que facilitará la gestión del stock y evitará tanto la escasez de productos críticos como el exceso de alimentario de lo que puede generar costos adicionales en almacenamiento. El hecho de que la aplicación se ejecute en un formato offline o dentro de una intranet es una ventaja adicional. Esto no solo para permite evitar los costos recurrentes asociados con servidores y hosting en la nube o incluso la contratación de proveedores, sino que también asegura que los datos de alimentario estén protegidos y no sean accesibles desde internet lo que refuerza la privacidad y seguridad de la información.

Los beneficiarios principales de la aplicación serán los empleados encargados de la gestión y venta de productos, así como la gerencia quienes podrán acceder a la información precisa y actualizada sobre el estado del inventario. Esto les permitirá tomar decisiones rápidas y fundamentadas optimizando las operaciones y mejorando la atención al cliente de esta forma se minimizarán problemas con la falta de productos y la inexactitud de las transacciones asegurando un control sobre el inventario.

1.2 Planteamiento del Problema

La empresa no cuenta con una aplicación, sistema, herramientas ni siquiera registros físicos adecuados para la gestión del inventario o el control de productos esto provoca varios inconvenientes como la falta de visibilidad en tiempo real del stock disponible. Por ejemplo, en ocasiones la empresa se da cuenta de que falta mercadería solo cuando se necesita realizar una venta o al momento de realizar la venta lo que retrasa las operaciones y afecta a la satisfacción del cliente.

La situación empeora cuando debido a la falta de visibilidad del inventario y de los productos los empleados no tienen acceso a información actualizada sobre el stock lo que los lleva a vender productos que no están disponibles o a ofrecerlos. Este tipo de errores genera insatisfacción en los clientes que esperan recibir productos específicos, pero no están disponibles en el momento que se necesita conllevando a la pérdida de ventas y ocasionando demoras innecesarias en atención.

Además, este desorden en la gestión del inventario, pero haciendo énfasis en el control de stock de productos lleva a situaciones como cuando un empleado vende un uniforme y se queda con el dinero, sin que este llegue a la empresa. Esto no solo representa una pérdida financiera,

sino que también genera desconfianza entre los empleados dentro de la empresa. La falta de trazabilidad de las transacciones y el control efectivo sobre las ventas contribuyen a que situaciones como estas pasen desapercibidas hasta que ya es demasiado tarde.

La falta de control sobre el stock de productos sumada a estos incidentes afecta directamente a las decisiones de compra de materia prima o la elaboración de nuevos productos. Sin información precisa sobre las existencias de productos en la empresa puede incurrir en compras excesivas o insuficientes lo que también resulta en un manejo ineficiente de los recursos. Este desorden no solo genera pérdidas económicas, sino que también deteriora la relación con los clientes y proveedores.

El impacto de la gestión manual del inventario sumado a otros errores de stock y transacciones afecta a la competitividad de la empresa. La falta de información precisa sobre el stock de los productos y las ventas provoca ineficiencias en el proceso de toma de decisiones y en el cálculo de las necesidades de producción. Esto en última instancia, disminuye la capacidad de la empresa para competir con otras del mismo rubro.

Impacto Cuantitativo de la Gestión Manual>

- Pérdidas Financieras: Un análisis preliminar muestra que las pérdidas debido a errores de alimentario y mal manejo de dinero asciende a aproximadamente el 8% de las ventas mensuales.
- Tiempo invertido en inventarios: Actualmente los empleados dedican entre 15 y 20 horas al mes realizando conteos manuales y verificaciones del inventario, lo que representa una significativa pérdida de productividad

- Frecuencia de errores: En un día laboral se suele perder mínimo 3 ventas diarias debido a la falta de productos lo que ha generado insatisfacción en los clientes sino también en la pérdida de ventas potenciales.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un prototipo funcional de una aplicación móvil para la gestión de inventario en la empresa que permita un control de stock, reducir errores y mejorar la eficiencia en las operaciones logísticas.

1.3.2 Objetivos Específicos

- Definir el alcance del proyecto, estableciendo los límites y las funcionalidades que se incluirán en la aplicación móvil de gestión de inventario.
- Definir la arquitectura de la aplicación móvil de gestión de inventario, incluyendo los componentes, las interfaces y la tecnología que se utilizará para el desarrollo.
- Desarrollar un prototipo funcional, que implemente las funcionalidades claves del sistema.

1.4 Antecedentes

Actualmente, la información de los inventarios no se maneja de forma física, tampoco de forma tecnológica la misma se lleva de manera manual. se lleva la información solo de manera temporal, igualmente las decisiones que se toman son en aquel preciso momento con la información que el personal tiene en “la mente”. No es una forma óptima de manejar la información en la empresa, ya que con el tiempo genera inconvenientes al realizar los inventarios. Ya que cada vez que se hace el inventario se tiene que hacer desde cero. E incluso existen momentos mantiene un inventario por largo tiempo.

Esto conlleva que al pasar bastante tiempo la información se pierda. Ya que esta no se encuentra de manera física o tecnológicamente. Además de que también al realizar procesos logísticos y de compra. Esto también conlleva en que se haga en este momento el inventario para poder saber que se tiene en la empresa.

La empresa busca adaptarse a las nuevas tecnologías para poder conllevar una aplicación móvil que les permita manejar el inventario a través del tiempo y que les muestre lo que tiene la empresa en tiempo real. Donde la implementación de una aplicación móvil para la gestión del inventario también traerá más beneficios a la empresa, además de la parte logística, ya que también afecta al área de compras y de ventas.

Antecedentes en el sector textil:

En el sector textil, la gestión automatizada de inventarios ha demostrado ser una solución eficiente para resolver problemas de visibilidad y precisión en el control de existencias. Diversos estudios han mostrado que la implementación de tecnologías como RFID (Radio Frequency Identificacion) y sistemas de gestión en tiempo real han permitió a las empresas textiles mejorar significativamente la exactitud de su inventario, mejorando la eficiencia operativa.

Por ejemplo, en un estudio realizado por la Universidad Javeriana, se evidencio que la implementación de sistemas automatizados para el control de inventarios en empresas textiles medianas contribuyo a la mejora de los procesos de compra y ventas, facilitando la toma de decisiones basadas en datos actualizados. La adopción de tecnologías en la gestión de inventarios también resulto en una mayor satisfacción de los clientes, al garantizar que los productos estuvieran disponibles cuando se requerían, mejorando la eficiencia operativa.

Adicionalmente, un análisis de la Universidad Cesar Vallejo sobre la automatización de inventarios en pequeñas empresas del sector textil subraya como estas tecnologías permiten el monitoreo continuo de las existencias, evitando desabastecimiento y optimizando el tiempo de reposición de productos. Este tipo de soluciones contribuye a la creación de una cadena de suministro más ágil y eficiente, con un impacto directo en los costos operacionales y en la competitividad de las empresas.

1.5 Alcance

El desarrollo de este prototipo tiene como objetivo crear una aplicación móvil de gestión de inventario para una empresa textil aplicando la metodología ágil a lo largo de sus fases. Al finalizar este proyecto se entregará lo que es un prototipo funcional que permitirá gestionar el control de stocks manejo de usuarios y roles, así como el registro de ventas. Este prototipo se enfocará en ofrecer funcionalidades clave que lo invitan a optimizar las operaciones logísticas de la empresa reducción de errores y mejorando la eficiencia.

El sistema incluirá las siguientes funcionalidades:

- Manejo de usuarios y roles: Sistema de autenticación y autorización que permita asignar roles específicos a los usuarios y restringir el acceso a funcionalidades según su rol.

- Control de stock: Gestión de productos mediante operaciones CRUD (crear, leer, actualizar, eliminar).
- Registro de ventas: Funcionalidad para registrar transacciones realizadas en efectivo o mediante transferencia con la posibilidad de adjuntar una fotografía como comprobante si es transferencia.

Limitaciones del prototipo:

La aplicación estará limitada a un entorno interno en la empresa, funcionando exclusivamente dentro de la red local de la empresa, sin necesidad de conexión a internet. Esto garantiza mayor seguridad y evita gastos recurrentes en infraestructura.

El prototipo se enfocará únicamente en las funcionalidades mencionadas anteriormente. Las demás funcionalidades no se contemplarán en esta etapa del prototipado de la aplicación, pero se podrán agregar en futuras iteraciones.

La escalabilidad del sistema no es de característica de alta prioridad en la etapa actual del prototipado de la aplicación. La aplicación está diseñada para satisfacer a las necesidades actuales del cliente en la empresa. Sin embargo, la arquitectura permitirá futuras expansiones si se requiere agregar más funcionalidades a medida que la empresa crezca.

CAPÌTULO II: FUNDAMENTACIÓN TEÓRICA

2. Marco Teórico

2.1. Aplicación Móvil

2.1.1 ¿Qué es una Aplicación Móvil?

Las aplicaciones móviles se definen como software para todos los dispositivitos móviles como, por ejemplo: celulares móviles, tabletas y otros dispositivos móviles. Un usuario puede usar varias aplicaciones móviles para ciertos propósitos ya sea personales o profesionales. (Silva, 2020).

Aplicaciones similares y sus impactos en otras empresas:

Desarrollo de una aplicación móvil en Android Studio para la gestión de alimentaria y maquinaria en una empresa: un ejemplo relevante es el desarrollo de una aplicación móvil para la gestión de alimentaria y maquinaria de una empresa realizada en Android Studio. Este tipo de aplicaciones permite optimizar la administración del inventario realizar un seguimiento de la ubicación y el estado de los productos y mejorar la eficiencia operativa. La implementación de la aplicación resultó en una mejora significativa en la precisión del inventario y la reducción de errores lo que permitió a la empresa manejar su stock de manera más efectiva y tomar decisiones más informadas sobre las compras y ventas. Estos beneficios generaron un impacto positivo en la empresa al mejorar la toma de decisiones y la eficiencia en los procesos operativos.

Aplicación móvil para el control de inventario en una empresa distribuidora: Otro ejemplo es el desarrollo de una aplicación móvil para el control de inventario en una empresa distribuidora. La aplicación permitió una gestión centralizada de alimentario integrando diferentes departamentos y asegurando que los productos se mantuvieran disponibles para los clientes sin exceder los niveles de stock. Esto resultó en una mejora de la eficiencia operativa y

un aumento en la satisfacción de los clientes al garantizar la disponibilidad de los productos en el momento adecuado. Además, al optimizar el control de inventarios la empresa fue capaz de reducir el riesgo de desabastecimiento y mejorar la planificación de compras y reposición.

Aplicación móvil para la gestión de inventarios en el sector textil: Una aplicación similar fue implementada en una empresa textil lo que permitió controlar el inventario de productos de forma más eficiente. Esta aplicación ayudó a la empresa a reducir los costos asociados con el exceso de stock mejorando el proceso de reposición y asegurando que los productos más demandados estuvieran siempre disponibles. Como resultado la empresa experimentó una reducción de costos y una mejora en la rotación de productos la que permitía a la empresa mejorar su competitividad y rentabilidad en el mercado.

Impacto de estas aplicaciones en las empresas: el impacto de estas aplicaciones en las empresas ha sido significativo mejorando la precisión en el manejo del inventario reduciendo costos operativos y optimizando los procesos de reposición. Al implementar soluciones móviles similares las empresas pueden mejorar la visibilidad de su inventario reducir el riesgo de desabastecimiento y mejorar la satisfacción del cliente lo que a su vez puede incrementar la rentabilidad y competitividad en el mercado.

2.2 Beneficios de la Automatización del Inventario

La gestión de inventarios en empresas textiles es crucial para optimizar la producción, reducción de costos y la mejora de satisfacción al cliente. La implementación de sistemas automatizados en este sector ha demostrado beneficios significativos como la reducción de errores una mayor eficacia una mayor eficiencia operativa y por último una mejora en la toma de decisiones estratégicas.

- Reducción de errores humanos: La automatización minimiza las discrepancias en el registro y seguimiento de inventarios lo que permite asegurar los datos precisos y confiables. Al eliminar el trabajo manual en el proceso de control de inventarios se reducen los errores humanos comunes como la entrada incorrecta de datos o la pérdida de productos, las empresas textiles que implementa la automatización de inventarios o experimentado una mejora considerable en la precisión de sus registros lo que ha permitido mantener un stock más fiable y una mayor confianza en la información.
- Eficiencia operativa: Los sistemas automatizados agilizan los procesos y optimizan la cadena de suministro en toda la empresa. La implementación de tecnologías como el software de gestión ERP y sistemas de seguimiento en tiempo real ha permitido a las empresas textiles reducir significativamente a los tiempos de reposición de productos lo que asegura que los niveles de inventario se mantengan adecuados sin sobrecargar el almacén con productos donde accesorios. Esto contribuye a una operación más ágil y eficiente de lo que se traduce en menores costos operativos y una respuesta más rápida a las demandas del mercado.
- Mejora en la toma de decisiones: La disponibilidad de información en tiempo real facilita la toma de decisiones estratégicas basadas en datos precisos y actualizados. Debido a la automatización las empresas textiles pueden acceder a información detallada sobre el estado de su inventario en cualquier momento lo que permite tomar decisiones informadas sobre la producción la compra de materias primas y la planificación de ventas. La visibilidad del inventario y también facilita la identificación de tendencias de demanda lo que ayuda a las empresas a ajustar su producción y stock en función de los cambios en el mercado.

Casos de éxito de automatización en la industria textil:

- Empresas de moda en España: varias marcas de moda en España como Zara, han implementado sistemas automatizados de gestión de inventarios que permiten controlar el stock en tiempo real. Estos sistemas han mejorado la eficiencia en la reposición de productos y reduciendo los tiempos de entrega lo que ha incrementado satisfacción al cliente.
- Textiles en América latina: Empresas textiles en países de América latina han experimentado mejoras en la eficiencia operativa tras implementar sistemas de gestión automatizados. Por ejemplo, algunas empresas han logrado reducir el tiempo de reposición de productos y mejorar la rotación de inventarios lo que les ha permitido optimizar sus operaciones y mantener niveles de stock adecuados.

2.2.1 Ventajas de una Aplicación Móvil para la Gestión de Inventarios

2.3 Marco de trabajo para desarrollo móvil

Una aplicación móvil para la gestión de inventarios en una empresa textil ofrece varios beneficios, entre los que destacan:

- Automatización de datos: Ayuda a reducir los errores humanos y garantiza la precisión de la información almacenada en la empresa, mejorando la fiabilidad de los registros de inventarios.
- Información en tiempo real: Proporciona datos actualizados sobre los niveles de inventario, lo que facilita la toma de decisiones oportunas en cuanto al reabastecimiento y la planificación logística. Esto contribuye a la mejora de la eficiencia operativa al evitar desabastecimientos o excesos de stock.

- Visibilidad del stock: Permite a los responsables del almacén tener una visión clara y completa de los productos, lo que facilita una gestión eficiente y rápida del inventario, optimizando los procesos de almacenamiento y distribución.

Contar con herramientas que gestionen el inventario en tiempo real no solo mejora la precisión de los datos, sino que también optimiza la eficiencia en los procesos críticos, como la compra y el reabastecimiento de materia prima o stock, lo que puede ayudar a asegurar una ventaja competitiva para la empresa.

2.3.1 React Native

React Native es un marco de trabajo de código abierto o desarrollado por Facebook, que permite crear aplicaciones móviles nativas para plataformas Android e IOS. Este framework se basa en la biblioteca React, aunque cabe aclarar que no es exactamente la misma biblioteca que se utiliza en el desarrollo web, ya que React Native está orientada específicamente al desarrollo de aplicaciones móviles. Sin embargo, conserva la mayoría de los conceptos fundamentales de React, como componentes, estados y el uso del ciclo de vida de estos. Una de las grandes ventajas de react native que permite utilizar JavaScript dentro del mismo código de manera nativa, facilitando la integración y el desarrollo rápido

Con lo mencionado anteriormente, el hecho de usar React Native nos permite optimizar la integración en las plataformas de Android e IOS, ya que se reduce los tiempos de desarrollo. Debido a su desarrollo multiplataforma, nos permite crear una única base de código, haciendo que el proceso de desarrollo sea más rápido y económico. Por último, como React Native se dijo anteriormente está basada en la librería de React, queda asimismo la posibilidad igualmente de hacerlo web, así mismo reduciendo tiempos de desarrollo y una curva de aprendizaje.

2.3.2 Flutter

Flutter es un framework de desarrollo de aplicaciones móviles de código abierto creado por Google. Permite desarrollar aplicaciones nativas para plataformas como Android, iOS, Windows, macOS y la web utilizando una sola base de código. Flutter se destaca por su alto rendimiento gracias a su motor de renderizado propio, que optimiza la velocidad y la fluidez de las interfaces de usuario. Este framework se basa en el lenguaje de programación Dart, también desarrollado por Google, lo que facilita la creación de aplicaciones rápidas y de alto rendimiento.

Una de las características clave de Flutter es su enfoque en el desarrollo de interfaces de usuario (UI) mediante widgets, lo que permite a los desarrolladores construir interfaces personalizadas y complejas con facilidad. Flutter proporciona una gran cantidad de widgets prediseñados, que se pueden usar o modificar según las necesidades del proyecto. Además, permite la integración directa con funciones nativas, como la cámara o el GPS, garantizando una experiencia fluida y personalizada para los usuarios.

Gracias a su arquitectura de desarrollo multiplataforma, Flutter simplifica el proceso de desarrollo al permitir que una sola base de código sea utilizada para múltiples plataformas, lo que reduce significativamente los costos y tiempos de desarrollo. Además, su integración con otras herramientas y plataformas facilita la creación de aplicaciones que se ajusten a los estándares de las principales tiendas de aplicaciones.

2.3.3 Selección del Marco de Trabajo para el Desarrollo Móvil

En este caso, se ha optado por utilizar React Native como el framework principal para el desarrollo del prototipo funcional. Esta decisión se basa en varias razones clave:

- Uso de JavaScript: React Native permite el uso de JavaScript, un lenguaje ampliamente conocido por los desarrolladores web. Esta característica facilita la curva de aprendizaje, permitiendo que los desarrolladores web puedan aprovechar sus conocimientos previos en el desarrollo móvil.
- Desarrollo multiplataforma: React Native ofrece la ventaja de poder crear aplicaciones para Android y iOS utilizando una única base de código. Esto no solo reduce significativamente el tiempo y los recursos necesarios para desarrollar en ambas plataformas, sino que también permite un mantenimiento más sencillo de las aplicaciones.
- Familiaridad y experiencia previa: React Native se basa en React, una librería de JavaScript ampliamente utilizada en el desarrollo web. Esta familiaridad con la tecnología permite una integración más fluida en el flujo de trabajo de los desarrolladores, reduciendo el tiempo de desarrollo y facilitando el uso del framework.
- Rendimiento optimizado: El framework utiliza componentes nativos, lo que asegura un rendimiento eficiente y una experiencia de usuario fluida. Al usar componentes que se ejecutan directamente en el dispositivo, React Native puede ofrecer un rendimiento similar al de las aplicaciones nativas.

No obstante, React Native también presenta algunas limitaciones:

- Dependencia de módulos nativos: Aunque el framework permite utilizar componentes nativos, en algunos casos es necesario integrar dependencias de

terceros para acceder a funcionalidades específicas del dispositivo, lo que puede aumentar la complejidad del desarrollo.

- Personalización limitada: La personalización de las interfaces puede ser más difícil en comparación con Flutter, que utiliza un sistema de widgets altamente personalizables desde su núcleo. Esto puede hacer que en React Native, la creación de interfaces complejas sea más laboriosa.
- Actualizaciones y soporte: React Native puede enfrentar problemas cuando las plataformas móviles actualizan sus sistemas operativos. La dependencia de bibliotecas externas puede complicar la integración con nuevas versiones de las plataformas móviles, lo que puede afectar el mantenimiento y soporte de las aplicaciones.

Por otro lado, Flutter ofrece una mayor personalización de la interfaz y un rendimiento consistente gracias a su motor de renderizado propio. Sin embargo, Flutter usa Dart, un lenguaje que tiene una curva de aprendizaje más pronunciada, especialmente para los desarrolladores que no tienen experiencia previa en él.

Debido a la familiaridad con JavaScript y React, así como la necesidad de una solución de desarrollo multiplataforma eficiente, React Native se ajusta mejor a las necesidades del proyecto, garantizando un desarrollo rápido y eficaz.

Análisis de Compatibilidad con MongoDB y Node.js

React Native es compatible con MongoDB mediante la integración con Node.js, lo que permite crear aplicaciones móviles que interactúen directamente con bases de datos no relacionales. Node.js, a su vez, es conocido por su capacidad para manejar múltiples peticiones

simultaneas de manera eficiente, siendo esto beneficioso para el desarrollo de aplicaciones que requieren interacciones en tiempo real con la base de datos.

Tabla 1

Comparativa entre React Native y Flutter

Aspecto	React Native	Flutter
Lenguaje de Programación	JavaScript (React)	Dart
Desarrollo Multiplataforma	Sí (Android y iOS con una sola base de código)	Sí (Android y iOS con una sola base de código)
Rendimiento	Bueno, con uso de componentes nativos	Excelente, debido a su motor de renderizado
Facilidad de Aprendizaje	Relativamente fácil (si ya se usa React)	Curva de aprendizaje más alta (Dart)
Flexibilidad en Diseño	Moderada, con componentes listos para usar	Alta, con widgets altamente personalizables

Nota: En esta tabla se puede evidenciar que Flutter es altamente personalizable pero debido a que se tiene un corto periodo de desarrollo, el previo conocimiento de la tecnología es crucial.

2.3 Base de Datos

2.3.1 ¿Qué es una Base de Datos Relacional?

Consiste en la colección o almacenamiento datos en relaciones que ya están predefinidas, donde estos mismos datos se almacenan en una o más tablas. Donde una base de datos relacional entabla relaciones entre tablas o la información mediante la unión de estas. El modelo de una base de datos relacional fue desarrollado por EF Codd desde IBM en los años de 1970, este modelo como se habló previamente nos permite crear relaciones entre datos, pero debe existir un atributo en común en las dos tablas para poder establecer una conexión, cada tabla en una base de datos relacional contiene una clave primaria, actuando como un identificador único esta es clave para poder generar relaciones en otras tablas donde está en otra table se la considera como una clave externa. Algunas ventajas de una base de datos relacional son: Flexibilidad, ACID

(Atomicidad, coherencia, aislamiento y durabilidad), Seguridad y Normalización de la base de datos relacional. (Google, s.f.)

2.3.2¿Qué es una Base de Datos no Relacional?

También conocida como base de datos NoSQL como su nombre lo dice implícitamente no contiene relaciones este tipo de base de datos sirven para almacenar datos con un esquema flexible. Pero el usar una base de datos no relacional tiene varias ventajas como: Flexibilidad, Escalabilidad, Alto rendimiento (consultas)y Altamente funcional. Por último, un registro en una base datos no relacional se guarda como un documento con sus respectivos atributos como un único documento no ocupa dependencias. (Amazon, s.f.)

2.3.3 Diferencias entre Base de Datos Relacionales y no Relacionales

Si bien los dos modelos se centran en almacenar información, existen diferencias entre sí, pero cada una guarda la información de manera diferente. Uno de los aspectos que más diferencia entre las dos es el rendimiento ya que en las bases de datos relacionales dependen del hardware para un mayor rendimiento y menor latencia, sin embargo, una base de datos no relacional depende de la red esta misma tenga una baja latencia esto permitiendo un mayor rendimiento. La escalabilidad en una base de datos relacional se puede manejar de manera vertical todo esto dependiendo del hardware, en cambio las bases de datos no relacionales son altamente escalables debido a que esta puede distribuir la carga de manera más eficiente, ya que esta distribuye la carga en cargas más pequeñas. (Amazon Web Services, s.f.)

2.3.4 Selección del Modelo de Base de Datos

En este caso se ha optado por utilizar un modelo de base de datos no relacional, específicamente el motor MongoDB, debido a las siguientes razones:

- Almacenamiento interno: los datos se almacenarán en servidores internos de la empresa, garantizando alta disponibilidad y control sobre la información
- Flexibilidad y autonomía de registros: MongoDB permite almacenar productos y sus atributos como documentos independientes, facilitando el manejo de datos dinámicos sin necesidad de relaciones estrictas entre registros
- Escalabilidad horizontal: MongoDB permite distribuir la carga eficientemente, lo que resulta crucial para un sistema que puede expandirse en el futuro. (IBM, s.f.)

Aparte de las características mencionadas anteriormente mencionadas sobre porque se eligió MongoDB, también se elige este motor de base de datos por el tipo de estructura de datos que manejara la aplicación. En un sistema como la gestión de productos y ventas, los datos tienden a ser no estructurados o mayormente dinámicos, lo que hace que una base de datos no relacional sea la mejor opción debido a la necesidad de definir un esquema rígido de antemano. En cambio, MongoDB permite agregar o modificar atributos sin afectar al resto de la base de datos.

En términos de rendimiento esperado, MongoDB es adecuado para aplicaciones que requieren un procesamiento rápido de datos en tiempo real, como el control de stock y registro de ventas. Además, MongoDB es capaz de manejar grandes volúmenes de datos sin comprometer la velocidad, especialmente con las operaciones CRUD necesarias en la aplicación.

2.4 ¿Qué es Backend?

El backend es la parte de una aplicación o página web que se encarga de gestionar y procesar la información detrás de escena. A diferencia del frontend, que es visible y accesible para los usuarios, el backend funciona en segundo plano. Su principal función es la de almacenar, procesar y organizar los datos, además de gestionar las operaciones que permiten que la aplicación o el sitio web funcione correctamente. El backend incluye componentes como bases de datos, servidores, y aplicaciones que se encargan de realizar operaciones, como la autenticación de usuarios o el procesamiento de solicitudes. Además, el backend interactúa con el frontend para devolver la información necesaria y proporcionar una experiencia de usuario fluida y eficiente.

2.4.1 Node.js

Se enfoca en la ejecución de código de JavaScript a eventos asincrónicos, este mismo puede ejecutarse en varios sistemas operativos donde este mismo se ejecuta como un servicio, se ejecutará el servicio en el backend o en segundo plano donde a el usuario no es visible ni accesible. Node.js tiene varias características como: Eventos asincrónicos, control de eventos, Alto rendimiento, Sin almacenamiento de buffer, multiplataforma, multiparadigma, código abierto, escalable. Existe una gran documentación y apoyo de la comunidad de desarrolladores permitiendo una mejora continua. (Lacruz, 2023)

2.4.2 Laravel

Es de código abierto desarrollado en 2011 enfocándose en simplificar el código a comparación de PHP tradicional donde los puntos fuertes de Laravel son: Documentación, Eloquent ORM, Routing, Middlewares, artisan (Consola), comunidad entre otras. El desarrollar un backend con Laravel puede resultar en uno muy robusto una de las grandes complicaciones es

la curva de aprendizaje, el cual puede afectar de manera significativamente en los tiempos de desarrollo del backend. Por último, Laravel nos da una manera de gestionar el código además de funciones de empaquetado propios para sustituir de cómo se hacía en PHP con su sintaxis, esto puede generar confusión si se ha trabajado antes con PHP, pero es cuestión de tiempo para trabajar con los nuevos conceptos que maneja Laravel. (Marc, 2020)

2.4.3 Diferencias entre Node.js y Laravel

La principal diferencia y más importante es en el lenguaje de programación de cada una ya que Laravel maneja usa PHP como base, en cambio Node.js maneja JavaScript como base. Cada una trabaja de manera diferente Laravel trabajando de manera MVC y en cambio Node.js trabaja de manera asincrónica y bucle de eventos este mismo puede manejar varias peticiones a la vez. La arquitectura monohilo que maneja Node.js le otorga una velocidad de ejecución menor a Laravel. (LaraAdmin, 2022)

La arquitectura monohilo de Node.js le otorga una velocidad de ejecución menor en comparación con Laravel en situaciones de alta concurrencia, ya que en Node.js todas las operaciones se manejan en un solo hilo de ejecución, mientras que Laravel, con su estructura MVC, maneja las operaciones de forma secuencial. A pesar de esto, Node.js se beneficia de su enfoque asincrónico y es más eficiente en aplicaciones que requieren alto rendimiento, como las que gestionan datos en tiempo real.

En este caso se selecciona Node.js para el backend debido a:

- Curva de aprendizaje: La facilidad de aprender JavaScript es un factor clave y crucial, debido a que se está familiarizado con este lenguaje. Se reduce de manera muy significativa la curva de aprendizaje, así mismo reduciendo la complejidad del proyecto y mejora la eficiencia en el desarrollo.

- Multiplataforma: Node.js es multiplataforma, esto quiere decir que el mismo backend funciona tanto para web y móvil. Eliminando la necesidad de rehacer el backend desde cero o crear incluso uno nuevo si se decide expandir la aplicación a otras plataformas en el futuro.
- Escalabilidad: Node.js se destaca por su escalabilidad. Su modelo de bucle de eventos permite que las aplicaciones manejen grandes volúmenes de solicitudes simultáneamente, lo que lo hace ideal para aplicaciones que requieren un manejo eficiente de datos en tiempo real.

En este proyecto, Node.js interactúa directamente con MongoDB. Donde ambas tecnologías están diseñadas para trabajar con el formato JSON, facilitando la integración. MongoDB almacena los datos en un formato similar a JSON (BSON), lo que hace que las operaciones de lectura, escritura y actualización de datos sean más eficientes. Además, MongoDB permite un esquema flexible que se adapta fácilmente a los cambios en los datos, algo fundamental cuando se gestionan productos con atributos que pueden variar.

Limitaciones de Laravel

Aunque este marco de trabajo es robusto y ampliamente utilizado, tiene algunas limitaciones en el contexto de este proyecto. La estructura MVC no está diseñada para manejar grandes volúmenes de datos y peticiones simultaneas. Esto puede generar cuellos de botella en el rendimiento cuando se requiere un procesamiento en tiempo real. Recordando que se base en PHP, lo que lo hace que no sea una buena opción para sistemas que necesitan una gran escalabilidad y un rendimiento más rápido en ambientes concurrentes.

2.5 Arquitectura de la Aplicación Móvil

2.5.1 ¿Qué es una Arquitectura Cliente Servidor?

La arquitectura cliente servidor es un modelo diseñado en el cual las funciones de una aplicación están separadas entre 2 entidades principales el cliente y el servidor donde el cliente realiza las solicitudes de servicios o recursos mientras que el servidor proporciona dichos servicios o recursos ya sea como base de datos aplicaciones servicios u archivos este modelo es ampliamente utilizado en redes informáticas y aplicaciones web o incluso en aplicaciones móviles. El cliente generalmente usa un dispositivo electrónico como una computadora o un dispositivo móvil qué interactúe directamente con el usuario mientras que el servidor es una máquina más robusta o un servidor que almacena los datos y gestionan los servicios requeridos normalmente estas comunicaciones entre ambos se realiza mediante protocolos estándar y desea como HTTP, FTP, SMTP o HTTPS.

2.5.2 Ventajas de la Arquitectura Cliente Servidor

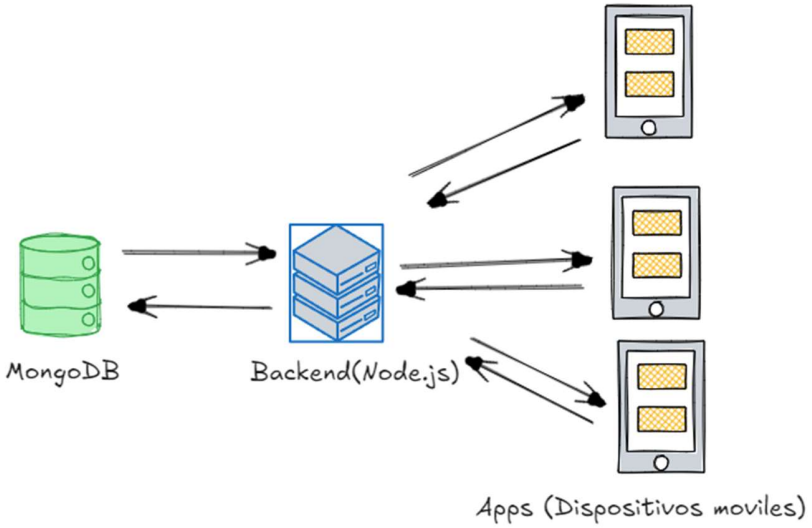
- Centralización de recursos: Los datos y servicios se encuentran centralizados en el servidor lo que facilita su gestión actualización mantenimiento.
- Escalabilidad: La arquitectura permite agregar nuevos clientes o expandir los recursos de un servidor según las necesidades.
- Facilidad de mantenimiento: las actualizaciones o cambios se realizan principalmente en el servidor reduciendo así la necesidad de modificar cada cliente a la vez.
- Seguridad: es relativamente más sencillo implementar políticas de seguridad centralizadas en un servidor mayormente cuando éste se encuentra dentro de una red LAN o dentro de la empresa como tal.

- Eficiencia en el uso de recursos: los clientes pueden ser menos potentes ya que esto sólo mandan solicitudes al servidor y este mismo se encarga de realizar las tareas más complejas.
- Flexibilidad: Es compatible con múltiples plataformas y tecnologías así permitiendo la integración de diferentes sistemas o incluso a diferentes servicios en el servidor como tal.

Donde estas ventajas hacen que una arquitectura de cliente servidor sea obviamente adoptada en el desarrollo de aplicaciones y servicios en red proporcionando así una estructura robusta y adaptable a las diversas necesidades tecnológicas además de las necesidades de la empresa o usuario. (IONOS, 2024)

Además, se trabajara con una arquitectura de cliente-servidor, donde en este caso el cliente consiste en la app móvil como tal y el servidor estaría alojada en el mismo dispositivo, algunas de las ventajas de este arquitectura para este contexto es el desempeño local, donde al estar alojado todo localmente los tiempos de respuesta se reducen considerablemente, otro punto a destacar es que la app es offline donde no es necesario tener una conexión a internet para que la aplicación móvil funcione, por último, tenemos que la arquitectura es considerablemente simple y fácil de implementar. (Daemon, 2024)

Ilustración 1
Arquitectura Cliente Servidor de la Aplicación



Nota: Elaboración propia

2.6 Casos de Prueba de Aceptación

Se realizarán casos de prueba de aceptación para garantizar que la aplicación móvil cumpla con los requisitos funcionales y satisfaga las expectativas del cliente y los usuarios finales. Estas pruebas aseguraran que la aplicación sea aceptable y adecuado para un entorno de producción. En el desarrollo ágil, los casos de prueba de aceptación se basan en los criterios de aceptación definidos en las historias de usuario, que especifican de cómo debe comportarse el sistema en este caso la aplicación desde el punto de vista del usuario final. La ejecución de estos casos de prueba de aceptación permite validar que las funcionalidades o características implementadas cumplan con lo solicitado por el cliente. Además, al integrar estos casos en el ciclo de desarrollo ágil, nos permite asegurar que el prototipo se mantenga en un estado funcional y alineado con las expectativas del cliente. (Venema, 2024)

CAPÌTULO III: METODOLOGÍA

3.1 ¿Qué es una Metodología en el Desarrollo de Software?

Se las puede definir como un marco de trabajo creados con el propósito de organizar, planificar y gestionar todo lo que abarca el desarrollo de un software. La misma nos ofrece lo que es un enfoque sistemático, reducción de riesgos y errores, optimización de recursos y una mejor comunicación entre todos los involucrados. (Vega, 2024)

3.2 Tipos de Metodologías

3.2.1 Metodología Tradicional

La metodología tradicional en el desarrollo de software surgió como una respuesta a lo que es la necesidad de estructurar un proceso que inicialmente era completar de una manera artesanal para mejorar los proyectos para alcanzar los objetivos deseados. Se adaptan principios y fundamentos de metodología de otras áreas al ámbito del software este enfoque introduce un desarrollo dividido en etapas secuenciales las cuales no son iterativas lo cual representa una mejora frente a las necesidades del momento donde tenemos entre las metodologías tradicionales más conocidas se encuentra RUP y MSF, donde se enfocan en lo que es la documentación detallada y en la ejecución de un plan de proyecto definido desde la fase inicial. Sin embargo, estas metodologías presentan ciertas limitaciones como altos costos para implementar cambios por el hecho mismo de que no son iterativas y una falta de adaptabilidad en entornos volátiles. En general las metodologías tradicionales priorizan la documentación planificación y procesos donde se usa plantillas técnicas de administración y revisiones para garantizar el cumplimiento de los objetivos.

3.2.2 Metodología Ágil

Las metodologías ágiles surgieron como una respuesta a las limitaciones de los enfoques tradicionales en el desarrollo de software ofreciendo flexibilidad y a la objetividad para proyectos en entornos cambiantes estas metodologías se centran en los aspectos clave como la toma de decisiones diferida y la planificación adaptativa lo que mejora la capacidad de reacción frente a cambios y potencia el desarrollo eficiente en proyectos de gran escala.

El manifiesto ágil define principios fundamentales cómo priorizar las interacciones humanas sobre los procesos, la creación de software funcional por encima de documentación extensa fomentar la colaboración con el cliente en lugar de depender únicamente de contratos y responder a cambios de manera eficiente en vez de seguir estrictamente un plan o un flujo de trabajo entre las metodologías ágiles más destacadas se encuentra XP siendo ésta extreme programming y scrum así como otras como icónicas y crystal methods.

3.2.2.1 Scrum

Siendo de una de las metodologías ágiles más utilizadas y se basa en un marco iterativo e incremental para la gestión de proyectos complejos su estructura está diseñada para ofrecer entregas frecuentes de valor al cliente fomentar la colaboración del equipo y adaptarse rápidamente a los cambios las ventajas de scrum son promueve la colaboración constante entre el cliente y el equipo facilita la adaptación rápida a los cambios en los requisitos mejora la transparencia a través de reuniones frecuentes y artefactos claros asegura entregas frecuentes de valor permitiendo así al cliente obtener resultados tangibles en poco tiempo y por último fomenta la mejora continua del equipo. Siendo las desventajas de Scrum son requiere un equipo comprometido y bien capacitado puede ser menos efectivo en proyectos con equipos muy

grandes o distribuidos y por último depende de gran utilidad en la claridad y gestión del Product backlog por parte del producto owner.

3.2.2.2 Extreme programming (XP)

Desarrollado por Kent Beck se centra en lo que es la adaptabilidad más que en la previa visibilidad reconociendo que los cambios en los requisitos son inevitables y fomenta la capacidad de respuesta a lo largo de todo el proyecto sus prácticas clave incluyen un desarrollo iterativo pruebas unitarias continuas programación en parejas interacción constante con el cliente fue factorización de código y simplicidad en el diseño.

Una pequeña comparación entre XP y Scrum es mientras que XP enfatiza en técnicas específicas de desarrollo como las pruebas unitarias y la programación por parejas SCRUM se centra más en la organización y gestión del equipo además del proyecto, ambos comparten la filosofía ágil y pueden complementarse dependiendo de las necesidades del proyecto. (Roberth Figueroa-Díaz, 2007)

3.3 Selección de Metodología

Se aplicará la metodología ágil Scrum, ya que nos permite enterrar un prototipo funcional en un lapso reducido. Scrum ofrece flexibilidad y adaptación a los requerimientos del cliente, permitiendo generar entregables prototipado que facilitan la retroalimentación. Esto asegura que los ajustes necesarios se realicen durante el desarrollo y no al final del proyecto. Además, esta metodología se adapta a tiempos ajustados, ya que busca iteraciones continúas con entregas funcionales en el camino (Saavedra, 2016).

La elección de Scrum en este caso tiene como objetivo garantizar la entrega de un prototipo funcional dentro de los plazos estrictos, alineándose con las necesidades específicas del desarrollo del prototipo funcional.

3.3.1 Aplicación de Scrum

Para el desarrollo del prototipo funcional de la aplicación móvil de gestión de inventario se definieron 3 sprint cada uno con objetivos específicos alineados a las necesidades del proyecto.

3.3.1.1. Definición de Roles en Scrum

Dado que este proyecto se realizará de manera individual, las responsabilidades de los roles de scrum fueron asumidas por una sola persona y organizadas de la siguiente manera:

- Product Owner: Actuando como representante del cliente, priorizando los requisitos del sistema según las necesidades detectadas y definiendo los entregables de para cada sprint.
- Scrum Máster: Desempeñando el rol de facilitador, asegurándose de que se siguieran las prácticas de Scrum y resolviendo cualquier impedimento durante el desarrollo.
- Equipo de desarrollo: Responsable de la implementación del frontend con react native, el backend con Node.js y Express, la configuración de la base de datos MongoDB, el diseño de la arquitectura y la realización de pruebas necesarias

Para gestionar múltiples roles asumidos por el mismo desarrollador de manera eficiente, una solución puede ser la división de tareas o responsabilidades únicas. Así mismo, se dividirán las actividades para cada rol, pero no se mezclarán tareas similares e incluso responsabilidades entre los diferentes roles. Asegurando que cada rol tenga la atención que se requiere, sin que se interrumpa otro rol.

3.3.1.2 Gestión de Riesgos y Desafíos

En este proyecto, el asumir múltiples roles puede ser un desafío. Para mitigar los riesgos y asegurar el cumplimiento de los objetivos:

- Gestión del tiempo: Los sprints se planificarán con objetivos claros y medibles para optimizar la productividad.
- Adaptación a cambios: La flexibilidad de Scrum permite reestructurar tareas y surgen imprevistos o cambios que sean necesarios durante el desarrollo.
- Resolución de problemas técnicos: Se dedicaron bloques específicos de tiempo para investigar y resolver cualquier dificultad técnica que surja durante los sprints.

CAPÌTULO IV: DESARROLLO

4. Desarrollo de la Aplicación

4.1 Historias de Usuario

Las Historias de Usuario son descripciones breves y sencillas de funcionalidades o características que el sistema debe tener, centradas en las necesidades del usuario final o cliente. En el contexto de nuestro proyecto, las historias de usuario están alineadas con las iteraciones planificadas para poder satisfacer las funcionalidades que debe cumplir la aplicación, pero también permitiendo en cada sprint añadir mejoras, ya sea en la interfaz y experiencia del usuario.

Cada iteración busca cumplir con una funcionalidad o característica de la aplicación, desde la creación de usuarios con su respectivo rol hasta el rediseño de la interfaz. A continuación, se desglosa cada sprint con sus historias de usuario, tareas y pruebas.

4.2 Iteración 1

4.2.1 Historias de Usuario

Tabla 2

Historia de Usuario Iteración I

Historia de Usuario	Descripción
Historia 1: Inicio de sesión como Administrador	Como administrador, quiero iniciar sesión con mis credenciales para poder gestionar usuarios y roles dentro de la aplicación.
Historia 2: Creación de usuarios por Administrador	Como administrador, quiero poder crear nuevos usuarios con sus respectivos roles para poder gestionar el acceso y control en el sistema.
Historia 3: Asignación de Roles	Como administrador, quiero asignar un rol específico a cada usuario para controlar su acceso a distintas funcionalidades, como agregar o editar productos.
Historia 4: Acceso a Funcionalidades según Rol	Como usuario con el rol adecuado, quiero acceder a funcionalidades específicas como visualizar productos, registrar nuevos productos o realizar cambios en el inventario.

Nota: Elaboración propia

4.2.2 Tareas

Tabla 3

Tareas Iteración I

Tarea	Descripción
Tarea 1: Implementación de autenticación para Administrador	Desarrollar el sistema de login y logout para permitir la autenticación del administrador que tendrá acceso al sistema.
Tarea 2: Funcionalidad de creación de usuarios por Administrador	Desarrollar la funcionalidad que permita al administrador crear nuevos usuarios con roles específicos, sin opción de registro por parte del usuario.
Tarea 3: Implementación de roles para usuarios	Desarrollar un sistema para asignar roles a los usuarios creados por el administrador, como “encargado” o “empleado”.
Tarea 4: Restricción de acceso según roles	Configurar el acceso a funcionalidades específicas como agregar productos, editar inventarios o consultar inventarios según el rol asignado al usuario.

Nota: Elaboración propia

4.2.3 Pruebas

Tabla 4

Pruebas de Aceptación iteración 1

Elemento	Descripción
ID del Caso de Prueba	CP-01
Funcionalidad	Sistema de autenticación y asignación de roles.
Descripción	Verificar que el sistema permita autenticar un administrador, asignar roles a los usuarios y controlar el acceso a funcionalidades.
Precondiciones	El sistema debe estar configurado con un administrador inicial y roles definidos. El administrador debe estar autenticado para crear usuarios.
Entrada Esperada	- Nombre de usuario y contraseña del administrador. - Roles asignados a los usuarios creados.
Procedimiento	1. Iniciar sesión con el administrador. 2. Crear un usuario con un rol específico. 3. Intentar iniciar sesión con las credenciales del usuario creado. 4. Verificar que el sistema permita o deniegue el acceso según las credenciales proporcionadas. 5. Verificar que el rol asignado se refleje en las funcionalidades accesibles por el usuario.
Resultado Esperado	- El sistema debe permitir al administrador gestionar la creación de usuarios y asignación de roles.

	- El sistema debe permitir que los usuarios accedan a las funcionalidades según su rol.
Criterios de Aceptación	- El sistema debe autenticar correctamente al administrador. - Los roles deben asignarse y gestionarse sin errores. - El acceso a las funcionalidades debe estar correctamente restringido o permitido según el rol del usuario.
Estado	Completado
Responsable	Jimmy Alvarado

Nota: Elaboración propia

4.3 Iteración 2

4.3.1 Historias de Usuario

Tabla 5

Historias de Usuario iteración 2

Historia de Usuario	Descripción
Historia 1: Crear Producto	Como administrador, quiero poder crear nuevos productos en el sistema para gestionar el inventario de manera eficiente.
Historia 2: Leer Productos	Como administrador, quiero poder consultar los productos existentes en el sistema para conocer el estado actual del inventario.
Historia 3: Actualizar Producto	Como administrador, quiero poder modificar los detalles de un producto para mantener la información actualizada.
Historia 4: Eliminar Producto	Como administrador, quiero poder eliminar productos que ya no están disponibles o no son relevantes para el inventario.

Nota: Elaboración propia

4.3.2 Tareas

Tabla 6

Tareas iteración 2

Tarea	Descripción
Tarea 1: Implementación de CRUD de Productos	Desarrollar las funcionalidades para crear, leer, actualizar y eliminar productos dentro del sistema de gestión de inventario.
Tarea 2: Interfaz para gestión de productos	Crear una interfaz para que el administrador pueda realizar las operaciones CRUD.
Tarea 3: Validación de campos	Implementar validaciones en los campos de productos (nombre, cantidad, precio, etc.) para asegurar datos correctos.

Nota: Elaboración propia

4.3.3 Pruebas

Tabla 7

Pruebas de Aceptación iteración 2

Elemento	Descripción
ID del Caso de Prueba	CP-02
Funcionalidad	Operaciones CRUD para productos.
Descripción	Verificar que las operaciones CRUD (crear, leer, actualizar, eliminar) funcionen correctamente para la gestión de productos en el inventario.

Precondiciones	El sistema debe permitir la creación, consulta, actualización y eliminación de productos.
Entrada Esperada	- Datos del producto (nombre, descripción, precio, cantidad).
Procedimiento	1. Crear un producto con los datos requeridos. 2. Consultar el producto recién creado. 3. Actualizar los detalles del producto. 4. Eliminar el producto.
Resultado Esperado	Las operaciones CRUD deben ejecutarse correctamente sin errores.
Criterios de Aceptación	- El producto debe crearse correctamente. - Los productos deben poder ser actualizados y eliminados sin errores.
Estado	Completado
Responsable	Jimmy Alvarado

Nota: Elaboración propia

4.4 Iteración 3

4.4.1 Historias de Usuario

Tabla 8

Historias de Usuario iteración 3

Historia de Usuario	Descripción
Historia 1: Registrar Venta	Como administrador, quiero registrar las ventas realizadas en efectivo o mediante transferencia para llevar un control de las transacciones.
Historia 2: Adjuntar Comprobante de Transferencia	Como administrador, quiero poder adjuntar una fotografía como comprobante si la venta fue realizada mediante transferencia.

Nota: Elaboración propia

4.4.2 Tareas

Tabla 9

Tareas iteración 3

Tarea	Descripción
Tarea 1: Implementación de Registro de Venta	Desarrollar la funcionalidad para registrar ventas realizadas en efectivo o mediante transferencia.
Tarea 2: Implementación de Adjunto de Comprobante	Crear la funcionalidad para adjuntar una fotografía como comprobante si la venta se realizó por transferencia.
Tarea 3: Interfaz de Registro de Ventas	Desarrollar una interfaz que permita al administrador registrar ventas fácilmente, con campos para monto y comprobante si aplica.

Tarea 4: Validación de Venta	Implementar validaciones para asegurar que las ventas se registren correctamente con los datos requeridos.

Nota: Elaboración propia

4.4.3 Pruebas

Tabla 10

Pruebas de Aceptación iteración 3

Elemento	Descripción
ID del Caso de Prueba	CP-03
Funcionalidad	Registro de ventas y adjunto de comprobantes.
Descripción	Verificar que el sistema permita registrar ventas realizadas y adjuntar un comprobante de pago si la venta es por transferencia.
Precondiciones	El sistema debe permitir el registro de ventas y la carga de archivos de imágenes.
Entrada Esperada	- Datos de la venta (monto, tipo de pago). - Foto del comprobante (si aplica).
Procedimiento	1. Registrar una venta en efectivo. 2. Registrar una venta por transferencia y adjuntar el comprobante.
Resultado Esperado	El sistema debe permitir registrar las ventas correctamente y adjuntar el comprobante en el caso de transferencias.

Criterios de Aceptación	- El sistema debe permitir registrar ventas correctamente. - El comprobante debe adjuntarse correctamente en caso de transferencia.
Estado	Completado
Responsable	Jimmy Alvarado

Nota: Elaboración propia

4.5 Iteración 4

4.5.1 Historias de Usuario

Tabla 11

Historia de Usuario iteración 4

Historia de Usuario	Descripción
Historia 1: Unificación de Pantallas	Como usuario, quiero que todas las pantallas de la aplicación tengan una estructura coherente para una navegación más fluida.
Historia 2: Reestructuración de Botones y Estilos	Como usuario, quiero que los botones y los estilos de la aplicación estén mejor organizados y sean más intuitivos para una experiencia de uso más agradable.
Historia 3: Ajustes de Diseño según Preferencias del Usuario	Como usuario, quiero que algunos elementos del diseño se ajusten según mis preferencias para que la aplicación se vea y funcione de la mejor manera para mí.

Nota: Elaboración propia

4.5.2 Tareas

Tabla 12

Tareas iteración 4

Tarea	Descripción
Tarea 1: Revisión de pantallas y componentes	Realizar una revisión de las pantallas actuales y los componentes utilizados en la aplicación para asegurarse de que todo tenga un diseño uniforme.
Tarea 2: Ajustes de botones y estilos	Modificar los botones y los estilos generales (fuentes, tamaños, márgenes, etc.) para que estén alineados con el diseño unificado.
Tarea 3: Ajustes de diseño adicionales	Realizar ajustes en el diseño que el usuario final solicite, como la reorganización de elementos o cambios específicos en la interfaz.

Nota: Elaboración propia

4.5.3 Pruebas

Tabla 13

Pruebas de Aceptación iteración 4

Elemento	Descripción
ID del Caso de Prueba	CP-04
Funcionalidad	Unificación de pantallas y ajustes de diseño.
Descripción	Verificar que todas las pantallas tengan un diseño uniforme y que los botones y estilos estén reestructurados de manera coherente.
Precondiciones	El sistema debe estar en funcionamiento con las pantallas ya desarrolladas.

Entrada Esperada	- Pantallas unificadas. - Botones y estilos mejorados.
Procedimiento	1. Revisar todas las pantallas y verificar que las modificaciones sean coherentes. 2. Realizar ajustes en los botones y estilos de acuerdo con lo solicitado. 3. Probar la experiencia del usuario con las modificaciones.
Resultado Esperado	Las pantallas deben tener un diseño uniforme y coherente. Los botones y otros elementos deben ser más fáciles de usar.
Criterios de Aceptación	- El diseño debe ser coherente en toda la aplicación. - Los botones y estilos deben mejorar la experiencia del usuario.
Estado	Completado
Responsable	Jimmy Alvarado

Nota: Elaboración propia

4.6 Iteración 5

4.6.1 Historias de Usuario

Tabla 14

Historias de Usuario iteración 5

Historia de Usuario	Descripción
Historia 1: Implementación del color celeste en la aplicación	Como cliente, quiero que la aplicación utilice el color celeste (o sus variantes) para que coincida con la identidad visual de mi marca.

Historia 2: Implementación del color blanco en la aplicación	Como cliente, quiero que el color blanco se utilice junto con el celeste para crear un diseño atractivo y equilibrado.
Historia 3: Revisión del diseño por parte del cliente	Como cliente, quiero revisar el diseño final para asegurarme de que los colores seleccionados cumplen con mis expectativas visuales.

Nota: Elaboración propia

4.6.2 Tareas

Tabla 15

Tareas iteración 5

Tarea	Descripción
Tarea 1: Aplicación de los colores solicitados	Implementar el color celeste y blanco en las pantallas y componentes de la aplicación, según las indicaciones del cliente.
Tarea 2: Ajustes de diseño según los colores aplicados	Ajustar la disposición de los elementos para asegurar que los colores se integren de manera armónica en todo el diseño.
Tarea 3: Revisión con el cliente	Realizar una revisión con el cliente para asegurarse de que los colores aplicados coinciden con sus expectativas y hacer ajustes si es necesario.

Nota: Elaboración propia

4.6.3 Pruebas

Tabla 16

Pruebas de Aceptación iteración 5

Elemento	Descripción
ID del Caso de Prueba	CP-09
Funcionalidad	Implementación del diseño con los colores solicitados.
Descripción	Verificar que el diseño de la aplicación esté actualizado con los colores celeste y blanco, solicitados por el cliente.
Precondiciones	El cliente debe haber proporcionado las tonalidades específicas de celeste y blanco.
Entrada Esperada	- Uso de los colores celeste y blanco en las pantallas y componentes.
Procedimiento	1. Aplicar los colores solicitados en las pantallas. 2. Ajustar la disposición de los elementos para una correcta integración de los colores. 3. Revisar la aplicación con el cliente.
Resultado Esperado	El diseño debe estar completamente actualizado con los colores solicitados y debe ser aprobado por el cliente.
Criterios de Aceptación	- El cliente debe aprobar los colores aplicados. - Los colores deben integrarse armoniosamente en todo el diseño de la aplicación.
Estado	Completado

Responsable	Jimmy Alvarado

Nota: Elaboración propia

5. Conclusiones y Recomendaciones

5.1 Conclusiones

Reducción de errores: El prototipo de la aplicación móvil ha demostrado un notable avance en la optimización de los procesos de control de inventarios haciendo énfasis en el control de stock y registro de ventas. La automatización de estos procesos ha permitido disminuir significativamente los errores que suelen surgir con la gestión manual, lo que mejora la precisión de los datos y facilita una visión más clara del stock disponible. Aunque aún no se encuentra en uso por los usuarios finales.

La metodología Scrum ha permitido un desarrollo ágil y flexible del prototipo, facilitando la entrega de incrementos funcionales y ajustes rápidos según las necesidades cambiantes del proyecto. Este enfoque ha sido clave para asegurar que el prototipo se adapte a las expectativas del cliente, garantizando que los avances sean constantes y que se puedan implementar mejoras a medida que surjan nuevas necesidades o se identifiquen posibles ajustes.

MongoDB, como base de datos no relacional, ha demostrado ser una opción eficiente para manejar el almacenamiento dentro del prototipo. Gracias a su flexibilidad y escalabilidad, el sistema tiene la capacidad de adaptarse al crecimiento futuro de la empresa, pueda manejar un volumen de daos mayor sin comprometer el rendimiento.

5.2 Recomendaciones

Aunque el prototipo esta alineado con los requisitos iniciales, es importante seguir con una retroalimentación continua con el cliente. Esto ayudara a identificar posibles áreas de mejora y permitirá ajustar las funcionalidades según las necesidades de la empresa, de modo que el prototipo se mantenga actualizado y se ajuste a medida de los requerimientos del cliente, antes que la aplicación sea lanzada a producción.

A medida que el prototipo avance hacia su versión final, se recomienda planificar la integración con otros sistemas externos en case de ser necesario, que puedan optimizar aún más la gestión del inventario. Esta integración no solo mejorara la eficacia del sistema, sino que también garantizara su capacidad de adaptarse a las necesidades crecientes de la empresa a medida que esta se expande.

Aunque la aplicación aún no se encuentra en uso por los usuarios finales, es fundamental realizar pruebas periódicas y ajustar la interfaz de usuario para asegurar que sea intuitiva y fácil de usar. Mantener una experiencia de usuario agradable y funcional es esencial para cuando la aplicación finalmente se ponga en producción, asegurando que los usuarios finales se adapten rápidamente y aprovechen las capacidades del sistema de manera eficiente.

Referencias

3PL, L. (15 de Jun de 2023). Lógicos 3PL: https://www.logicos3pl.com/blog/importancia-visibilidad-tiempo-real-los-inventarios-para-una-gestión-eficiente#:~:text=En%20el%20mundo%20del%20comercio,mejor%20servicio%20a%20los%20clientes.

Amazon. (s.f.). *Amazon Web Services*. Amazon Web Services: https://aws.amazon.com/es/nosql/

Amazon Web Services. (s.f.). Amazon Web Services: https://aws.amazon.com/es/compare/the-difference-between-relational-and-non-relational-databases/

Flutter. (s.f.). Flutter: Framework de desarrollo para aplicaciones nativas. Recuperado de https://docs.flutter.dev/?_gl=1xpgih7_gaODE1NTU1MDAwLjE3Mzc2ODQ3MDk._ga_04YGWK0175*MTczNzY4NDcxMC4xLjEuMTczNzY4NTExMy4wLjAuMA..

Daemon. (23 de Julio de 2024). *Daemon 4*. https://www.daemon4.com/empresa/noticias/arquitectura-cliente-servidor/

Doonamis. (28 de Mayo de 2024). *Doonamis*. https://www.doonamis.com/unit-testing-y-su-impacto-a-la-hora-de-desarrollar-las-apps/

Rodríguez, R. (2020). Desarrollo e implementación de un sistema de gestión de inventarios para mejorar la eficiencia operativa en empresas textiles. Universidad Católica de Colombia. Recuperado de https://repository.ucatolica.edu.co/server/api/core/bitstreams/74361ce6-5c85-4d14-be72-bf4a49077549/content

Martínez, J. (2021). Optimización de la gestión de inventarios mediante aplicaciones móviles en la industria textil. Universidad Técnica de Cotopaxi. Recuperado de https://repositorio.utc.edu.ec/items/604b6f0b-89cb-426c-8b23-680742c512fd

Google. (s.f.). *Google Cloud*. Google Cloud: https://cloud.google.com/learn/what-is-a-relational-database?hl=es-419

IBM. (s.f.). https://www.ibm.com/topics/nosql-databases

IONOS, E. e. (31 de Enero de 2023). IONOS: https://www.ionos.mx/digitalguide/servidores/know-how/modelo-cliente-servidor/

Álvarez, F., & García, M. (2021). Desarrollo y arquitectura de sistemas backend en aplicaciones web: Conceptos y aplicaciones. Dialnet. Recuperado de https://dialnet.unirioja.es/servlet/articulo?codigo=7220420

Kiely, A. (2018). Development of mobile applications with React Native. Theseus. Recuperado de https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf

Lacruz, L. J. (3 de Junio de 2023). *Medium*. Medium: https://ljcl79.medium.com/nodejs-javascript-para-backend-d028e4463fe

LaraAdmin. (27 de Diciembre de 2022). https://www.laravel-entwickler.de/es/laravel-vs-node-js-cual-elegir/

Marc. (26 de Mayo de 2020). *Abalit Technologies*. https://www.abalit.org/barcelona/desarrollo-de-backend-para-app-movil-en-laravel#:~:text=Laravel%20es%20un%20framework%20open,programación%20del%20lenguaje%20nativo%20PHP.

Masulli, G. (12 de 06 de 2020). *SAP: arquitectura de software de 2 y 3 niveles (Two-tier Vs Three-tier) [Fotografia]*. Fornacion SAP: https://www.formacionsap.com/sap-arquitectura-de-software-de-2-y-3-niveles-two-tier-vs-three-tier/

MICROTECH. (28 de 02 de 2024). MICROTECH: https://www.microtech.es/blog/riesgos-y-desventajas-de-no-tener-los-stocks-actualizados

Muradas. (23 de Marzo de 2018). *OpenWebinars*. https://openwebinars.net/blog/sqlite-para-android-la-herramienta-definitiva/

Roberth Figueroa-Díaz, C. S. (February de 2007). *METODOLOGÍAS TRADICIONALES VS METODOLOGÍAS ÁGILES.* ResearchGate.: https://www.researchgate.net/publication/299506242_METODOLOGIAS_TRADICIONALES_VS_METODOLOGIAS_AGILES

Saavedra, M. (16 de Nov de 2016). *Designthinking*. https://designthinking.gal/scrum/

Silva. (7 de Octubre de 2020). *Softcorp*. https://servisoftcorp.com/definicion-y-como-funcionan-las-aplicaciones-moviles/#Que_es_una_aplicacion_movil

Vega. (8 de Octubre de 2024). *Gurusis*. https://gurusis.com/metodologias-de-desarrollo-de-software/

Venema, M. (26 de 04 de 2024). *NimbleWork*. NimbleWork: https://www.nimblework.com/es/agile/pruebas-de-aceptacion/

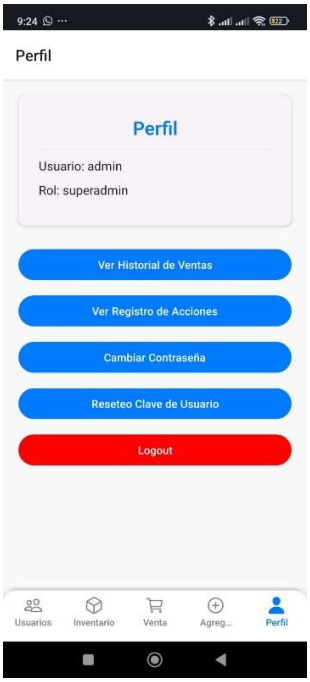
Revista de la Universidad Nacional Autónoma de México. (2020). React Native: Acortando las distancias entre desarrollo y diseño móvil multiplataforma. Recuperado de

https://www.revista.unam.mx/wp-content/uploads/v20_n5_a5_React-Native-acortando-las-distancias-entre-desarrollo-y-diseño-móvil-multiplataforma.pdf

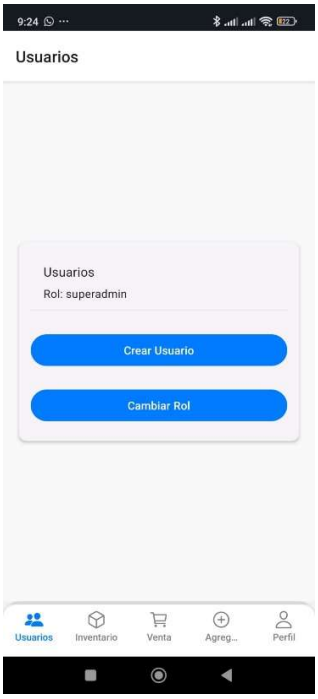
Anexos

Anexo A: Pantalla de Inicio de Sesión.

Anexo B: Pantalla de Perfil

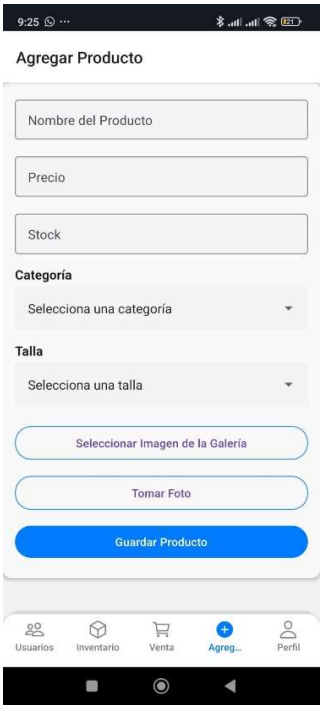


Anexo C: Pantalla de Usuarios.

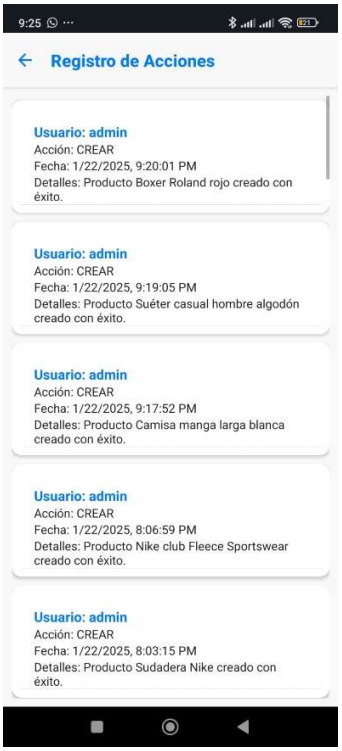


Anexo D: Pantalla de Usuarios (Formulario para creación de Usuario)

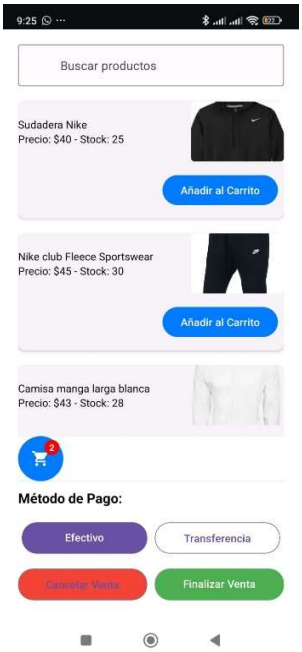
Anexo E: Pantalla Agregar Productos



Anexo F: Pantalla Registro de Acciones de los usuarios



Anexo G: Pantalla Realizar Ventas



Anexo H: Pantalla Historial de Ventas