ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

REINGENIERÍA DE MÓDULOS DE VENTAS, RECAUDOS E INVENTARIO DE APP A PWA CON HERRAMIENTA DEVOPS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

JOSE ISRAEL PEREZ ESPINOSA

jose.perez01@epn.edu.ec

ANDERSON FABIAN REVELO MORILLO

anderson.revelo@epn.edu.ec

DIRECTOR: VICENTE ADRIAN EGUEZ SARZOSA MSc.

adrian.eguez@epn.edu.ec

Quito, mayo 2022

AVAL

Certifico que el presente trabajo fue desarrollado por **Anderson Fabian Revelo Morillo** y **José Israel Pérez Espinosa** bajo mi supervisión.

Vicente Adrián Egüez Sarzosa MSc.

DIRECTOR DE PROYECTO

DECLARACIÓN DE AUTORÍA

Nosotros, **Anderson Fabian Revelo Morillo** y **José Israel Pérez Espinosa**, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

Anderson Fabian Revelo Morillo

José Israel Pérez Espinosa

AGRADECIMIENTO

En primer lugar, quiero agradecer a Dios por haberme dado la vida y permitirme llegar a este momento.

A mis padres Vidal y Angélica por ser los mejores padres que he podido pedir. Gracias por todo.

A Carmiña por su apoyo y por nunca dejar que me rinda.

A Anderson por su amistad y esfuerzo para terminar este proyecto.

A Adrián por su ayuda para culminar este sueño.

Israel Pérez

AGRADECIMIENTO

Quiero agradecer a toda mi familia. Todo el esfuerzo, tiempo, paciencia y apoyo tuvo como resultado la finalización de este proyecto de titulación.

A Judith por estar dándome ánimos en momento difíciles.

A todos mis amigos y en especial a mis compañeros de universidad Marcelo y mi compañero de tesis Israel por estar presente en los momentos de alegría y tristeza.

Por último, le agradezco al MSc. Adrián Egüez por guiarnos en la elaboración de este proyecto.

Anderson Revelo

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Las aplicaciones web progresivas en inglés progressive web apps (PWA) presentan ciertas ventajas sobre las aplicaciones híbridas, como el no necesitar pasar por los procesos de publicación de aplicativos en las tiendas, entre otras, que resultan en un ahorro de tiempo y dinero para los desarrolladores.

El objetivo principal del trabajo es aplicar la reingeniería de software en los módulos de ventas, recaudos e inventario de la aplicación móvil de la empresa Manticore Labs para convertirla a PWA mediante Scrum y Gitlab como herramienta DevOps.

La reingeniería se concentró en aspectos visuales, cambios en lógica de negocio, cambios en arquitectura, actualizaciones en los frameworks y documentación. El aplicativo fue desarrollado con el enfoque ágil Scrum con ayuda de las herramientas de DevOps de GitLab. Para la implementación se usó los frameworks Angular, Ionic para frontend y Nestjs con base de datos MySQL en el backend.

La implementación en producción se realizó utilizando los servicios de Firebase. Para las pruebas de usabilidad se utilizó SUS (System Usability Scale) encuestando a 12 personas obteniendo un resultado promedio de 73,12 de 100 en las preguntas planteadas, lo que permite concluir que la aplicación es amigable, pero que necesita mejorar la experiencia de usuario en nuevas implementaciones. Por lo tanto, se concluye que se cumplió el objetivo principal de este proyecto que consistió en aplicar una reingeniería al aplicativo móvil en los submódulos de ventas, recaudos e inventario a una aplicación PWA.

Palabras clave: Aplicación móvil, GitLab, PWA, reingeniería, Scrum, SUS.

ABSTRACT

The progressive web apps (PWA) have certain advantages over hybrid apps, such as not needing to go through the processes of publishing apps in stores, among others, which result in saving time and money for developers.

The main objective of the project is to apply software reengineering in the sales, collections and inventory modules of the mobile app of the company Manticore Labs to convert it to PWA through Scrum and Gitlab as a DevOps tool.

The reengineering was focused on visual aspects, changes in business logic, changes in architecture, updates in frameworks and documentation. The application was developed with the agile Scrum approach with the help of GitLab's DevOps tools. As for the frontend implementation Angular and Ionic frameworks were used, for backend Nestjs with MySQL database.

The production deployment was done using Firebase services. For the usability tests, 12 people were polled using SUS (System Usability Scale), getting an average result of 73.12 out of 100, which enable us to conclude that the application is friendly, except from it needs to improve the user experience in new implementations. Therefore, it is concluded that the main objective of this project was fulfilled, which consisted of applying a reengineering to the mobile application in the sales, collections and inventory submodules to a PWA application.

Keywords: GitLab, mobile app, PWA, reengineering, Scrum, SUS.

1. INTRODUCCIÓN

1.1. Antecedentes

El término aplicación web progresiva del inglés progressive web app (PWA) hace referencia a la creación de aplicaciones flexibles y adaptables mediante el uso exclusivo de tecnologías web [1]. Las PWA proporcionan la misma experiencia tanto en computadoras de escritorio como en dispositivos móviles. Evolucionan desde páginas en las pestañas del navegador hasta aplicaciones inmersivas de alto nivel [2], lo que permite que sean multiplataforma [3]. Una de las principales ventajas de las PWA, es evitar publicar el aplicativo en tiendas es decir que no tiene que pasar por todo el proceso de descarga, instalación y concesión de permisos. Una vez que un usuario muestra interés en una PWA, el buscador automáticamente sugerirá añadir un atajo de esta a la pantalla de inicio, el mismo que no se diferenciará al de una aplicación nativa [3]. Otro inconveniente de usar aplicaciones nativas con respecto a una PWA es que necesitan la decisión del usuario para tener instalada la versión actualizada, mientras una PWA siempre se tiene acceso a la versión más actualizada de la aplicación.

Manticore Labs es una empresa de desarrollo de software, que se encuentra ubicada en Quito, cuenta con una cartera de 14 clientes entre ellos Petroecuador, Uribe Schwarzkopf, Diners Club, etc. Manticore Labs ha desarrollado una aplicación híbrida, la cual entre sus funcionalidades, permite gestionar el inventario, ventas y recaudos de una empresa; esta aplicación se despliega a producción a través de las tiendas de aplicaciones tradicionales, Play Store de Google y App Store de Apple, lo que dificulta su actualización, debido al proceso que se lleva a cabo en estas tiendas de aplicaciones para aprobar su publicación, tomando entre 2 y 3 semanas, desde la solicitud de actualización hasta que los cambios en el aplicativo se ven reflejados en las tiendas. Además, Soporte de Google [4] indica que, para publicar la aplicación en la tienda de Play Store, se debe de pagar una cantidad de dinero.

El presente proyecto busca dar solución a esta problemática, mediante la reingeniería de esta aplicación a una PWA, la cual no necesita de una tienda de aplicaciones para ser instalada o actualizada en dispositivos.

1.2. Objetivos

1.2.1. Objetivo general

Aplicar la reingeniería de software en módulos de ventas, recaudos e inventario del aplicativo móvil de la empresa Manticore Labs a PWA mediante Scrum y Gitlab como herramienta DevOps.

1.2.2. Objetivos específicos

- Definir el alcance de la reingeniería de los módulos de ventas, recaudos e inventario.
- Establecer los requerimientos específicos a implementarse en la aplicación PWA.
- Rediseñar la base de datos y la arquitectura.
- Iniciar los flujos de integración y desarrollo continuo e implementarlos usando GitLab.
- Implementar los requerimientos del aplicativo en la aplicación PWA.
- Probar usabilidad de la aplicación PWA.

1.3. Alcance

En el presenta proyecto se realizará la reingeniería de los módulos de ventas, recaudos e inventario del aplicativo móvil de la empresa Manticore Labs a PWA, pasando por los procesos de ingeniería inversa, mejoramiento de la estructura, modularización del programa y reingeniería de datos del modelo general de reingeniería. Además, se realizará pruebas de usabilidad utilizando el sistema de escalas de usabilidad en inglés system usability scale (SUS).

1.4. Marco teórico

1.4.1. Aplicación móvil

Una aplicación móvil es un software que se ejecuta en dispositivos móviles y realizan tareas para el usuario. Las aplicaciones pueden estar preinstaladas en el dispositivo o puede descargarse de internet mediante los proveedores de aplicaciones móviles [5].

Existen varias formas de desarrollar una aplicación móvil. Cada una de ellas presentan diferentes características y limitaciones. Entre los tipos de aplicaciones móviles que se conocen son: nativas, web e híbridas [6].

Opciones en el desarrollo de una aplicación móvil:

Aplicaciones móviles nativas

Las aplicaciones móviles nativas están escritas en el lenguaje de programación y los marcos proporcionados por el propietario de la plataforma y se ejecutan directamente en el sistema operativo del dispositivo, como iOS y Android.

Aplicaciones móviles nativas multiplataforma

Las aplicaciones móviles nativas multiplataforma se pueden escribir en una variedad de marcos y lenguajes de programación diferentes, pero se compilan en una aplicación nativa que se ejecuta directamente en el sistema operativo del dispositivo.

Aplicaciones móviles híbridas

Las aplicaciones móviles híbridas se crean con tecnologías web estándar, como JavaScript, CSS y HTML5, y se incluyen como paquetes de instalación de aplicaciones. A diferencia de las aplicaciones nativas, las aplicaciones híbridas funcionan en un contenedor web que proporciona un tiempo de ejecución del navegador y un puente para las API de dispositivos nativos a través de Apache Cordova.

Aplicaciones web Progresivas

Las aplicaciones web progresivas en inglés progressive web app (PWA) son aplicaciones web que utilizan APIs y funciones emergentes del navegador web junto a una estrategia tradicional de mejora progresiva para ofrecer una aplicación nativa. Las aplicaciones web progresivas son un patrón de diseño útil, aunque no son un estándar formalizado. Se puede pensar que PWA es similar a AJAX u otros patrones similares que abarcan un conjunto de atributos de aplicación, incluido el uso de tecnologías y técnicas webs específicas.

Para poder llamar PWA a una aplicación web, técnicamente hablando debe tener las siguientes características: Contexto seguro, uno o más service workers y un archivo de manifiesto [1].

- **Contexto seguro:** la aplicación web se debe servir a través de una red segura. Ser un sitio seguro no solo es una buena práctica, sino que también establece la aplicación web como un sitio confiable, especialmente si los usuarios necesitan realizar transacciones seguras. La mayoría de las funciones relacionadas con una PWA, como la geolocalización e incluso los service workers, solamente están

disponibles cuando la aplicación se ha cargado mediante hypertext transfer protocol secure (HTTPS).

- **Service workers:** un service worker es un script que permite interceptar y controlar cómo un navegador web maneja las solicitudes de red y el almacenamiento en caché. Con los service workers se pueden crear páginas web rápidas y fiables sin la necesidad de una conexión activa a internet.
- **El archivo manifest:** un archivo JSON que controla cómo se muestra la aplicación al usuario y garantiza que las PWA sean detectables. Describe el nombre de la aplicación, la URL de inicio, los iconos y todos los demás detalles necesarios para transformar el sitio web en un formato similar al de una aplicación.

1.4.2. Reingeniería de software

La reingeniería es el proceso que implica el análisis y alteración de un sistema software existente para reconstituirlo en una nueva forma. El proceso de reingeniería engloba una combinación de procesos como se muestra en la Figura 1.

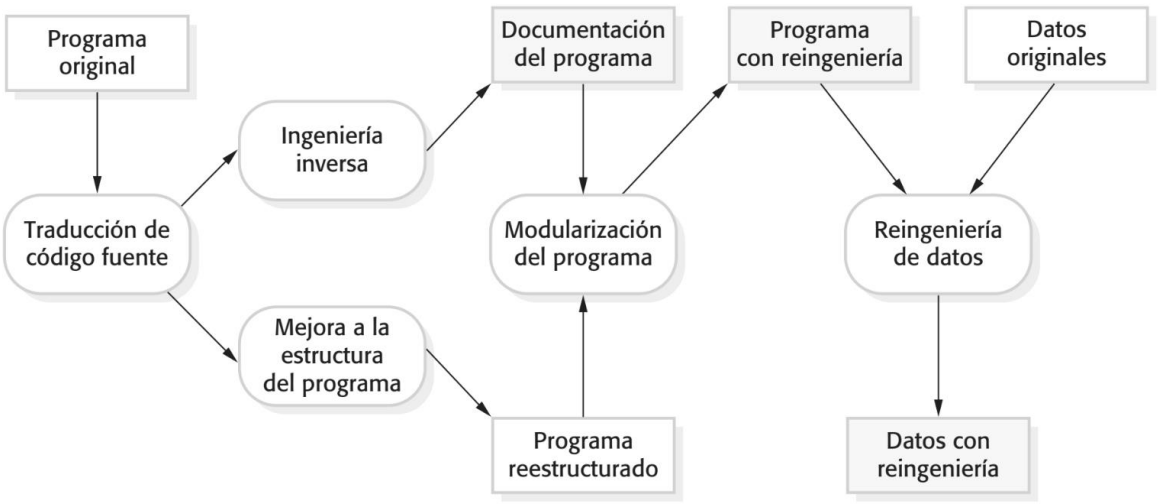


Figura 1. El proceso de reingeniería [7]

La Figura 1 representa el proceso de reingeniería. Toma como entrada un sistema heredado, mientras que la salida es una versión mejorada y reestructurada del mismo programa. Las actividades en este proceso de reingeniería son las siguientes:

- **Traducción del código fuente:** con una herramienta de traducción, el programa se convierte de un lenguaje de programación antiguo a una versión más moderna del mismo lenguaje, o a un lenguaje diferente.
- **Ingeniería inversa:** el programa se analiza y se extrae información de él. Esto ayuda a documentar su organización y funcionalidad. De nuevo, este proceso es, por lo general, completamente automatizado.
- **Mejoramiento de la estructura del programa:** la estructura de control del programase analiza y modifica para facilitar su lectura y comprensión, lo cual suele estar parcialmente automatizado, pero se requiere regularmente alguna intervención manual.
- **Modularización del programa:** las partes relacionadas del programa se agrupan y, donde es adecuado, se elimina la redundancia. En algunos casos, esta etapa implicará refactorización arquitectónica (por ejemplo, un sistema que use muchos almacenes de datos diferentes puede refactorizarse para usar un solo depósito). Éste es un proceso manual.
- **Reingeniería de datos:** los datos procesados por el programa cambian para reflejar cambios al programa. Esto puede significar la redefinición de los esquemas de bases de datos y convertir las bases de datos existentes a la nueva estructura. Por lo general, hay que limpiar los datos. Esto implica encontrar y corregir errores, eliminar registros duplicados, etcétera. Hay herramientas disponibles para auxiliar en la reingeniería de datos. Según Somerville [7] indica que para realizar una reingeniería no es necesario realizar todos los procesos especificados anteriormente.

1.4.3. Scrum

Scrum es un marco de trabajo para afrontar problemas complejos, que implementa una estructura iterativa, incremental, en la que, al inicio de una iteración el equipo revisa las tareas a realizarse, y selecciona lo que se convertirá al final del sprint en un incremento de la funcionalidad [8]. La Figura 2 representa el marco de trabajo Scrum.

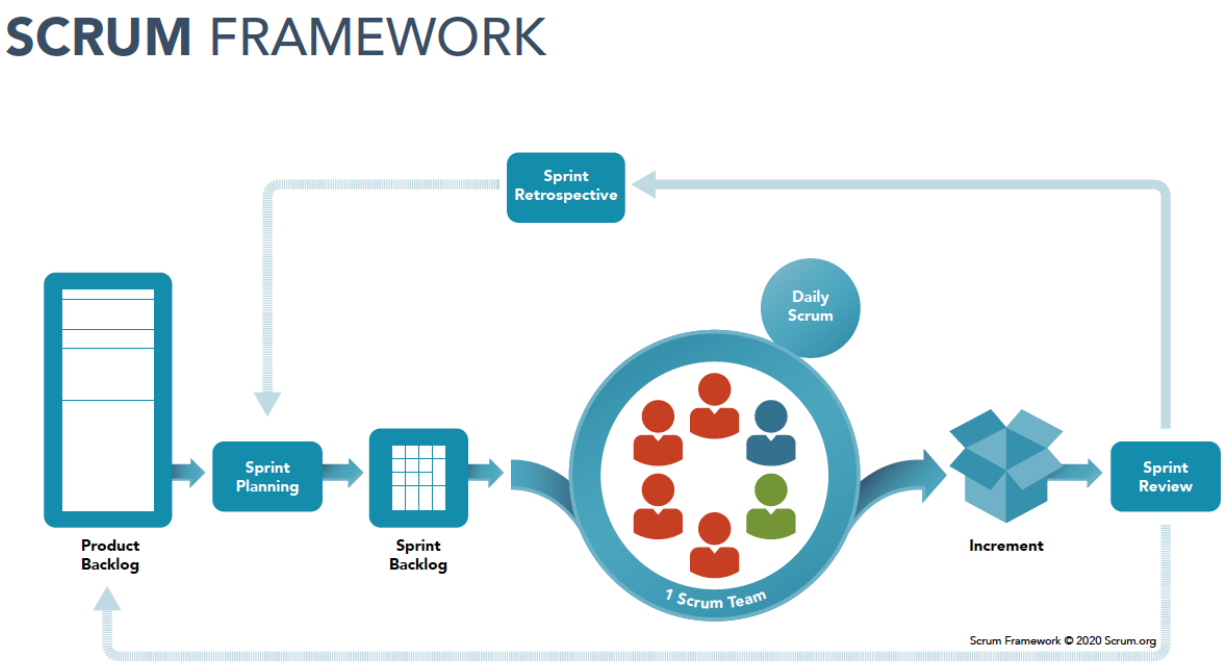


Figura 2. Esquema de Scrum **[9]**

Un equipo de trabajo de Scrum está conformado por tres roles:

- **Product Owner:** determina los requisitos iniciales generales y el plan de las entregas.
- **Scrum Master:** responsable de enseñar scrum y asegurarse de que todos sigan las reglas y prácticas.
- **Development Team:** responsable de desarrollar la funcionalidad y del éxito de cada iteración y del proyecto.

Scrum cuenta con los siguientes artefactos:

- **Product Backlog:** los requerimientos para el sistema son listados en el product backlog, que es, una lista de historias de usuario (HU). Siendo una historia de usuario una descripción corta y esquemática de las funcionalidades que debe incorporar el software. Para la presente solución se ha propuesto el formato mostrado en la Tabla 1. para la recopilación de historias de usuario.
  El producto owner es el responsable del contenido, priorización y disponibilidad del product backlog.
- **Sprint Backlog:** Define las tareas del product backlog que se realizarán en el sprint, además se ven reflejados los avances en cada tarea.
- **Increment:** es el producto completo que se va logrando sprint a sprint.

Eventos de Scrum:

- **Sprint:** es un periodo de tiempo generalmente comprendido entre una semana y un mes, durante el sprint el equipo trabaja para completar un conjunto de tareas (Sprint Backlog). El objetivo de cada sprint es obtener un producto final utilizable.
- **Sprint Planning:** es una reunión donde participa todo el equipo Scrum, con la finalidad de detallar, aclarar y delegar las características del producto a desarrollar.
- **Daily Scrum:** es una reunión que se lleva a cabo todos los días durante el sprint y que tiene una duración de 15 minutos, con el fin de comprobar el progreso o inconvenientes que se tiene al realizar cada tarea asignada a los miembros del equipo de desarrollo.
- **Sprint Review:** es una reunión que dirige el scrum master, en la cual asisten el equipo de desarrollo junto a las partes interesadas, con fin de revisar y analizar si se logró cumplir con el objetivo del sprint durante el trabajo realizado.
- **Sprint Retrospective:** es una reunión que se lleva a cabo con todo el equipo scrum, con el fin de conocer e inspeccionar aquello que produjo problemas durante el sprint, de modo que se permita proponer mejoras para iniciar con un nuevo sprint.

La Tabla 1 presenta el formato de las historias de usuario a utilizarse.

Tabla 1. Formato de historias de usuario

Historia de usuario	**Identificación**
Título:	
Descripción:	
Prioridad:	**Peso:**

El formato consta de los siguientes campos:

- **Identificación:** es el código que identifica a una historia de usuario.
- **Título:** es una descripción global del requerimiento del cliente.
- **Descripción:** es una pequeña descripción de la funcionalidad que se desea realizar.
- **Prioridad:** esta característica indica el nivel de importancia de la historia de usuario. Puede contener tres niveles: alta, media o baja.
- **Peso:** indica la complejidad del requerimiento.

1.4.4. GitLab

GitLab es una plataforma DevOps que crea un flujo de trabajo de software optimizado ofreciendo al equipo de desarrollo organizar, planificar, crear, proteger e implementar software de forma colaborativa. GitLab ayuda a los equipos a administrar y optimizar su ciclo de vida de entrega de software, permite la planificación a través de historias épicas, grupos e hitos para tener un seguimiento del progreso, también permite la administración de código a través de un sistema de control de versiones, y tiene procesos para la integración continua en inglés continuos integration (CI) y entregas continuas en inglés continuos delivery (CD), también conocido como CI/CD [10].

1.4.5. DevOps

El término DevOps se formó combinando las palabras "desarrollo" y "operaciones" y significa un cambio cultural que cierra la brecha sobre el modelo de desarrollo de software tradicional [11] . Cuando se adopta una cultura DevOps junto con un conjunto de prácticas y herramientas de DevOps se logra una automatización e integración de los procesos entre el desarrollo de software y los equipos de TI, para que puedan construir, probar y proveer software de manera más rápida y confiable [12].

Fases de DevOps

DevOps interviene en cada fase del ciclo de vida de la aplicación en planificación, desarrollo, implementación y operación [12].

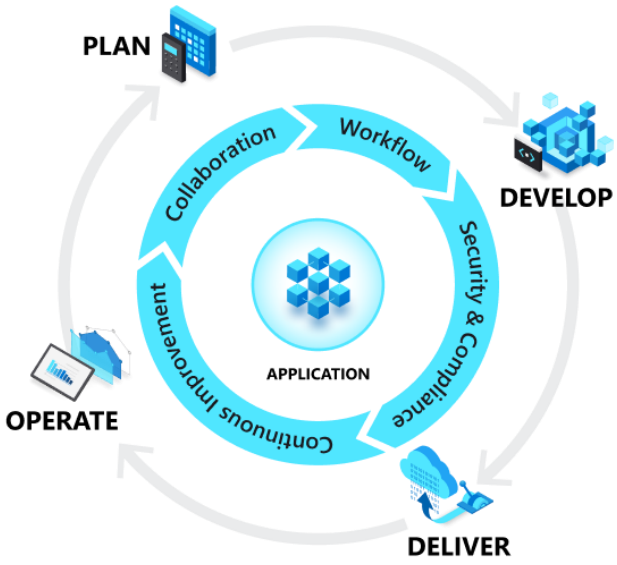


Figura 3. DevOps y el ciclo de vida de una aplicación [12]

- **Plan:** en esta fase los equipos idean, definen y describen características y capacidades de las aplicaciones y sistemas que se van a construir. Y realizan seguimiento de errores, la gestión del software con Scrum y el progreso del proyecto.
- **Develop:** en esta fase se realiza los aspectos de la codificación tanto en escritura, pruebas e integración del código. Se hace uso de herramientas que automatizan procesos como pruebas automatizadas e integración continua.
- **Deliver:** es el proceso de implementación de aplicaciones en entornos de producción consistentes y confiables. También incluye la implementación y configuración de la infraestructura y se definen procesos de gestión de versiones.
- **Operate:** en esta fase implica realizar mantenimiento, la supervisión y la resolución de problemas de aplicaciones en entornos de producción. La identificación mitigación de problemas y las realiza los equipos de DevOps antes de que afecten la experiencia del cliente.

Prácticas de DevOps

Los equipos implementan prácticas que aceleran, automatizan, mejoran fases y aumentan la productividad de los equipos en el ciclo de vida de desarrollo del software [12] .

- **Continuous integration and continuous delivery (CI/CD):** la integración continua es una práctica de desarrollo de software en la que los desarrolladores combinan los cambios de código con frecuencia en la rama de código principal. La integración continua emplea pruebas automatizadas, que se ejecutan cada vez que se confirma un nuevo código para que el código en la rama principal sea siempre estable.
- **Version control:** el control de versiones administra el código por versiones, realizando un seguimiento de las revisiones y el historial de cambios logrando una facilidad para revisar y recuperar códigos. Generalmente se utiliza sistemas de control de versiones como Git.
- **Agile software development:** es un enfoque donde se destaca la colaboración en equipo, comunicación con clientes y usuarios y una flexibilidad al cambio a través de ciclos de lanzamiento cortos en función a necesidades del cliente.
- **Infrastructure as code:** la infraestructura como código permite implementar entornos de desarrollo y pruebas reduciendo el riego del error humano al garantizar recursos del sistema de manera confiable, repetible y controlada.
- **Configuration management:** se refiere a la administración de los recursos en un sistema que incorpora servidores, máquinas virtuales y bases de datos. A través el uso de herramientas de gestión de la configuración, los equipos pueden realizar

cambios de manera controlada y ordenada, disminuyendo los riesgos de cambiar la configuración del sistema.

- **Continuous monitoring:** significa supervisar en tiempo real el rendimiento y estado de las aplicaciones, tanto de la infraestructura sobre la que se ejecutan como del software. La supervisión consiste en la recopilación de datos, y en la configuración de alertas para circunstancias predefinidas que requieren la acción de un operador.

1.4.6. Modelo vista controlador

Modelo Vista Controlador (MVC) es una arquitectura de software que separa en tres componentes los datos de la aplicación, la interfaz de usuario y la lógica de control [13].

El modelo que viene a representar los datos del sistema, la lógica de negocio y mecanismo de persistencia tiene la responsabilidad de almacenar los datos y definir reglas de negocio que son las funcionalidades del sistema.

La vista, que viene a representar la interfaz de usuario, es responsable de mostrar la información de los datos recibidos por el modelo.

El controlador actúa como intermediario entre el modelo y la vista permitiendo un flujo de información entre ellos mediante eventos recibidos de un usuario que actualiza o modifica al modelo [13].

1.4.7. Modelo vista vista modelo

La intención de la arquitectura modelo vista vista modelo (MVVM) es proporcionar una separación entre los controles de la interfaz de usuario y su lógica [14].

La vista es responsable de definir la estructura, el diseño y la apariencia de la interfaz de usuario, además no contiene lógica de negocio. Obtiene datos de su modelo vista a través de data-binding o invocando métodos en la vista modelo.

El modelo en MVVM es una implementación del modelo de dominio de la aplicación que incluye un modelo de datos junto con la lógica empresarial y de validación. Los ejemplos de objetos de modelo incluyen repositorios, objetos comerciales, objetos de transferencia de datos, objetos de entidad y proxy generados.

La vista modelo actúa como intermediaria entre la vista y el modelo, y es responsable de manejar la lógica de la vista. La vista modelo recupera datos del modelo y luego los pone a disposición de la vista, y puede reformatear los datos de alguna manera que simplifique el manejo de la vista. La vista modelo también proporciona implementaciones de comandos que un usuario de la aplicación inicia en la vista. Por ejemplo, cuando un usuario hace clic

en un botón en la interfaz de usuario, esa acción puede activar un comando en la vista modelo. La vista modelo también puede ser responsable de definir cambios de estado lógicos que afecten algún aspecto de la visualización en la vista, como una indicación de que hay alguna operación pendiente.

Cabe recalcar que cada uno de estos componentes tiene un papel distinto y separado, por lo que están desacoplados entre sí, lo que permite:

- Componentes que se pueden intercambiar.
- Implementación que se puede cambiar sin afectar a los demás.
- Componentes sobre los que trabajar de forma independiente.
- Pruebas unitarias aisladas.

1.4.8 Módulos

Ventas

El módulo de ventas ayuda a usuarios a realizar ventas. Revisar si deben entregar productos a clientes, si estos últimos han realizado previamente un pedido de productos. Está relacionado con el módulo de recaudos.

Recaudos

Este módulo se encarga del manejo de las cobranzas que realiza un vendedor, está relacionado con el de ventas. Permite que el vendedor tenga facilidad para repartir abonos a diferentes facturas. El vendedor puede elegir la cantidad de dinero que se abonará a una factura. Existen dos formas de pago: efectivo y depósito.

Inventario

El módulo de inventario ayuda a conocer el stock de productos de la empresa. Además, permite gestionar los inventarios del cliente para saber qué es lo que tiene o le hace falta.

1.4.9 Sistema de escalas de usabilidad

El sistema de escalas de usabilidad, en inglés system usability scale (SUS) evalúa la usabilidad percibida por los usuarios de un objeto, dispositivo o aplicación, utilizando 10 preguntas. Los usuarios completan el formulario una vez completada la prueba de usabilidad, las preguntas están alternadas entre positivas y negativas para que el usuario preste atención a sus respuestas. [15]

Las preguntas del formulario de SUS son las siguientes:

1. Creo que usaría este [sistema, objeto, dispositivo, aplicación] frecuentemente.

2. Encuentro este [sistema, objeto, dispositivo, aplicación] innecesariamente complejo.
3. Creo que el [sistema, objeto, dispositivo, aplicación] fue fácil de usar.
4. Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este [sistema, objeto, dispositivo, aplicación].
5. Las funciones de este [sistema, objeto, dispositivo, aplicación] están bien integradas.
6. Creo que el [sistema, objeto, dispositivo, aplicación] es muy inconsistente.
7. Imagino que la mayoría de la gente aprendería a usar este [sistema, objeto, dispositivo, aplicación] en forma muy rápida.
8. Encuentro que el [sistema, objeto, dispositivo, aplicación] es muy difícil de usar.
9. Me siento confiado al usar este [sistema, objeto, dispositivo, aplicación].
10. Necesité aprender muchas cosas antes de ser capaz de usar este [sistema, objeto, dispositivo, aplicación].

Para obtener los resultados, en las preguntas impares (1,3,5,7 y 9) se toma el valor asignado por el usuario, y se le resta 1. Por otra parte, para las preguntas pares (2,4,6,8 y 10), se toma el valor 5 y se resta el valor asignado por el usuario. A continuación, se suman los resultados de las preguntas pares e impares, y se multiplica por 2,5.

El mayor puntaje posible es 100, indicando una usabilidad muy alta, y a medida que la puntuación baje lo hará su usabilidad.

1.5. Metodología

Para el desarrollo del trabajo se definen las siguientes fases: inicio, planificación, implementación, revisión, y despliegue, en las cuales se puede ver la utilización de Gitlab como herramienta DevOps, el proceso de reingeniería y el marco de trabajo scrum.

Fase de inicio

En esta fase se definió el alcance de la reingeniería en los módulos de ventas, recaudos e inventario, para ello se revisó la documentación de los módulos. Luego se seleccionó los requerimientos a implementarse, los mismos que formaron parte del Backlog que fueron evaluados mediante su complejidad. Además, se realiza la reestructuración del diseño del aplicativo para finalmente poner en marcha los flujos de integración y desarrollo en GitLab. No se realizó la traducción del código fuente, pues la aplicación actual está implementada en JavaScript, mismo lenguaje a utilizar en la reprogramación. Las siguientes fases se repitieron en cada Sprint, y los cuales tuvieron una duración de 2 semanas.

Fase de planificación

Los requerimientos fueron recogidos y formaron parte del Sprint Backlog.

Fase de implementación

Se realizó la recodificación de funcionalidades del aplicativo móvil para que se comporte como una PWA. Para el control de las actividades se realizó reuniones a mitad del sprint. Las reuniones sirvieron para evaluar los avances y tomar medidas correctivas

Fase de revisión

Por cada sprint finalizado se realizó la presentación en una reunión (sprint review) del avance funcional a la parte interesada. El equipo scrum también realizó una reunión de retrospectiva.

Fase de despliegue

En base al nivel de aceptación se levantó en producción el aplicativo con las nuevas funcionalidades. Los últimos Sprints fueron tomados para efectuar las pruebas de usabilidad del aplicativo.

1.6. Ambiente de desarrollo

Las herramientas utilizadas para la reingeniería del aplicativo móvil son en su mayor parte software open-source. Las demás tienen una licencia educativa.

Lenguajes de programación utilizados se muestran en la Tabla 2.

Tabla 2. Descripción lenguajes de programación

Nombre	Descripción	Logo
HTML5	Es un lenguaje de marcado de hipertexto (HTML, por sus siglas en inglés) es el componente más básico de un sitio web, define el significado y la estructura del contenido web [16].	
CSS	Es un lenguaje de hojas de estilo usado para describir la presentación de un documento escrito en HTML o XML. Las Cascading Style sheets (CSS) describen como se deben renderizar los elementos en la pantalla [17].	
JavaScript	Es un lenguaje de programación y código abierto y popular en la web [18].	

TypeScript	TypeScript es un lenguaje de código abierto que se basa en JavaScript, una de las herramientas más utilizadas del mundo, al agregar definiciones de tipos estáticos [19].	

Frameworks utilizados se muestran en la Tabla 3.

Tabla 3. Descripción de Frameworks

Nombre	Descripción	Logo
Angular	Es un marco de diseño de aplicaciones y una plataforma de desarrollo para crear aplicaciones de una sola página construida sobre TypeScript, gratuito y de código abierto [20].	
NestJS	Es un marco para crear aplicaciones del lado del servidor Node.js escalables y eficientes [21].	
Node JS	Es un entorno ideado para la ejecución de JavaScript y la creación de aplicaciones escalables orientado a eventos asíncronos [22].	
Ionic	Ionic Framework es un conjunto de herramientas de interfaz de usuario de código abierto para crear aplicaciones móviles y de escritorio de alta calidad y rendimiento utilizando tecnologías web (HTML, CSS y JavaScript) con integraciones para frameworks populares como Angular, React y Vue [23] .	

Bases de datos utilizadas se muestran en la Tabla 4.

Tabla 4. Descripción de base de datos

Nombre	Descripción	Logo
MySQL	Es un gestor de bases de datos relacional de código abierto, permitiendo así MySQL es un gestor de bases de datos relacionales de código abierto [24].	

Entorno de desarrollo utilizado se muestra en la Tabla 5.

Tabla 5. Descripción del entorno de desarrollo

Nombre	Descripción	Logo
WebStorm	Es un entorno de desarrollo integrado para codificar en JavaScript y sus tecnologías relacionadas, que incluyen TypeScript, React, Vue, Angular, Node.js, HTML y hojas de estilo [25].	

Herramientas de control de versiones utilizadas se muestran en la Tabla 6.

Tabla 6. Descripción de herramientas de control de versiones.

Nombre	Descripción	Logo
Git	Git es un sistema de control de versiones distribuido de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia [26].	
GitLab	Es una herramienta de ciclo de vida de DevOps basada en la web que proporciona un administrador de repositorio Git que proporciona funciones de wiki, seguimiento de problemas e integración continua y canalización de implementación [10].	
GitKraken	Es una interfaz gráfica para la gestión de repositorios Git [27].	

Fuente de Recursos y herramienta de diagramación utilizado se muestran en la Tabla 7.

Tabla 7. Descripción de otras herramientas

Nombre	Descripción	Logo
Lucidchart	Es una herramienta web que permite diseñar y compartir diagramas de forma colaborativa en tiempo real [28].	Lucidchart
Figma	Es una plataforma para realizar diseños y prototipos [29].	

1.7. Estructura del proyecto

El presente proyecto integrador se encuentra estructurado por las siguientes secciones:

- **Metodología**

Revisión del aplicativo actual

En esta sección se describe el diseño y funcionamiento de las pantallas de pedidos e inventario del antiguo aplicativo móvil al que se le aplicó la reingeniería y una descripción de diseños de pantallas para el módulo de recaudos que iban a ser implementados.

Definición de los pasos del modelo general de reingeniería a realizarse

En esta sección se detallada las fases de la reingeniería a utilizarse, historias de usuario con el backlog, mejoramiento de la arquitectura y modularidad del aplicativo y un resumen de cambios en la base de datos.

Implementación de los flujos de integración y desarrollo continuo.

En esta sección se presenta la implementación del CI/CD.

Aplicación de la metodología de desarrollo

En esta sección se muestra la aplicación de la metodología de desarrollo junto con la planificación de los Sprints y la definición de roles.

- **Resultados y discusión.**

La sección de resultados y discusión muestran los resultados de usabilidad y la descripción de las pantallas del nuevo aplicativo móvil.

- **Conclusiones**

La sección de conclusiones evalúa los resultados obtenidos.

2. METODOLOGÍA

2.1. REVISIÓN APLICATIVO MÓVIL ACTUAL

Este apartado presenta una breve explicación del funcionamiento del aplicativo móvil actual, exponiendo los procesos que lleva a cabo.

2.1.1. Pantalla principal

La pantalla principal del aplicativo presenta al usuario vendedor 4 pestañas:

- Pedidos.
- Lubricantes.
- Control aceite.
- Ajustes.

Para el alcance del desarrollo del proyecto únicamente se tomó en cuenta realizar la reingeniería de la pestaña de pedidos. Esta pestaña muestra una pantalla con la lista de cronogramas del vendedor, lo que le permite al usuario caber los lugares que debe visitar en el día laboral. Los cronogramas son gestionados en el aplicativo web. La pantalla principal y el listado de cronogramas se pueden ver en la Figura 4.

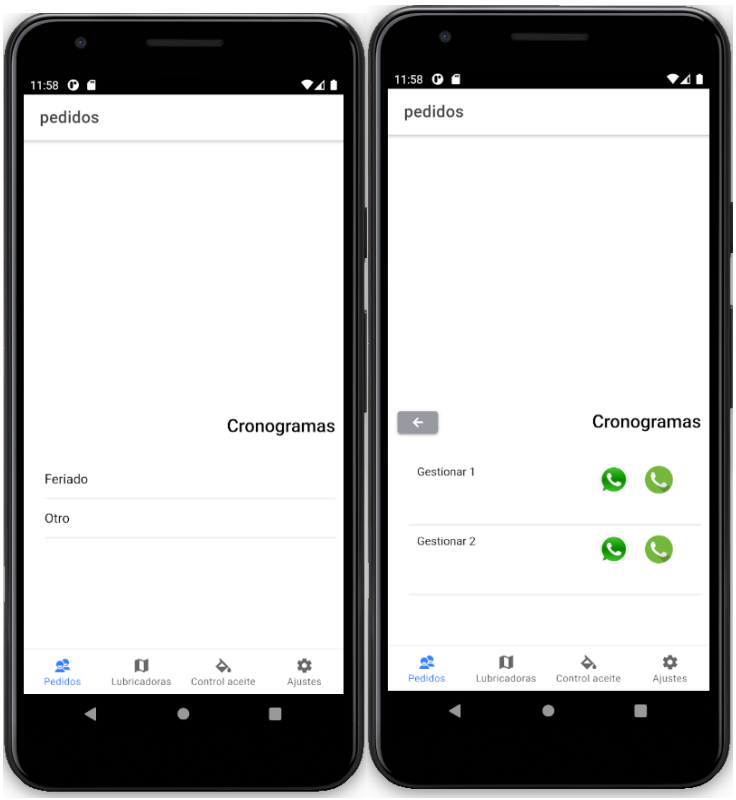


Figura 4. Pantalla principal y pestaña de pedidos**.**

Al seleccionar un cronograma, se muestra una pantalla con los nombres de los clientes que se deben visitar, al momento de seleccionar a un cliente se muestran las siguientes opciones:

- Registrar pedidos.
- Gestionar pedidos.
- Revisar stock.
- Cargas masivas.

La pantalla de gestión de cliente se puede ver en la Figura 5.

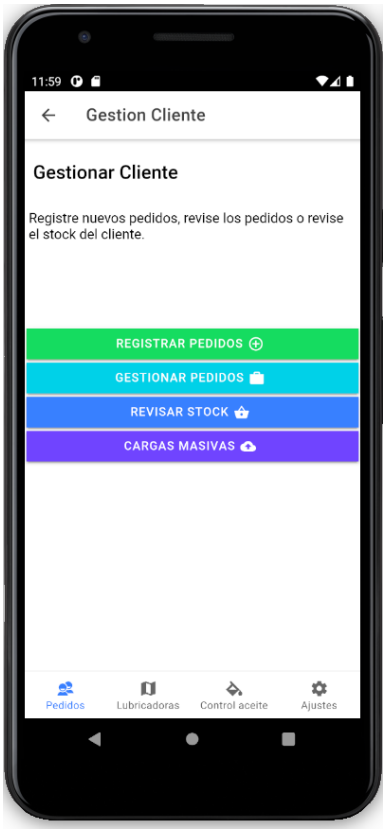


Figura 5. Pantalla de gestión de cliente

2.1.2 Revisar stock

El módulo de inventario permite revisar los artículos de un cliente. Para eso se debe seleccionar la opción “REVISAR STOCK”, en la pantalla de “Gestión cliente”.

El Inventario permite conocer los siguientes datos sobre cada artículo de un cliente:

- **Stock actual**: cantidad actual del artículo que posee el cliente.
- **Stock máximo**: cantidad máxima del artículo que puede tener el cliente.

- **Stock mínimo**: cantidad mínima del artículo que debe tener el cliente.
- **Stock alerta**: cantidad en la cual se debe realizar una visita al cliente.
- **Unidad de medida**: unidad de medida del artículo.

La pantalla de stock del cliente se puede ver en la Figura 6.

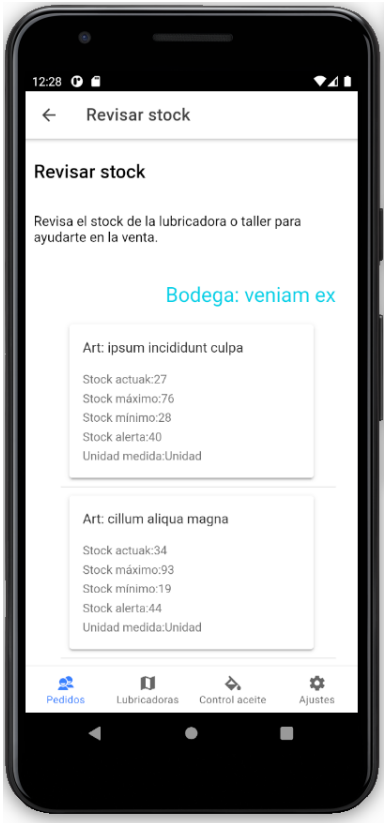


Figura 6. Pantalla de stock del cliente

2.1.3 Módulo pedidos

Para realizar un pedido se debe elegir un cronograma, en la pantalla de “Gestión Cliente” se presiona el botón “Registrar Pedidos” y se presiona el botón “Nuevo Artículo” para registrar un nuevo artículo como se muestra en la Figura 7.

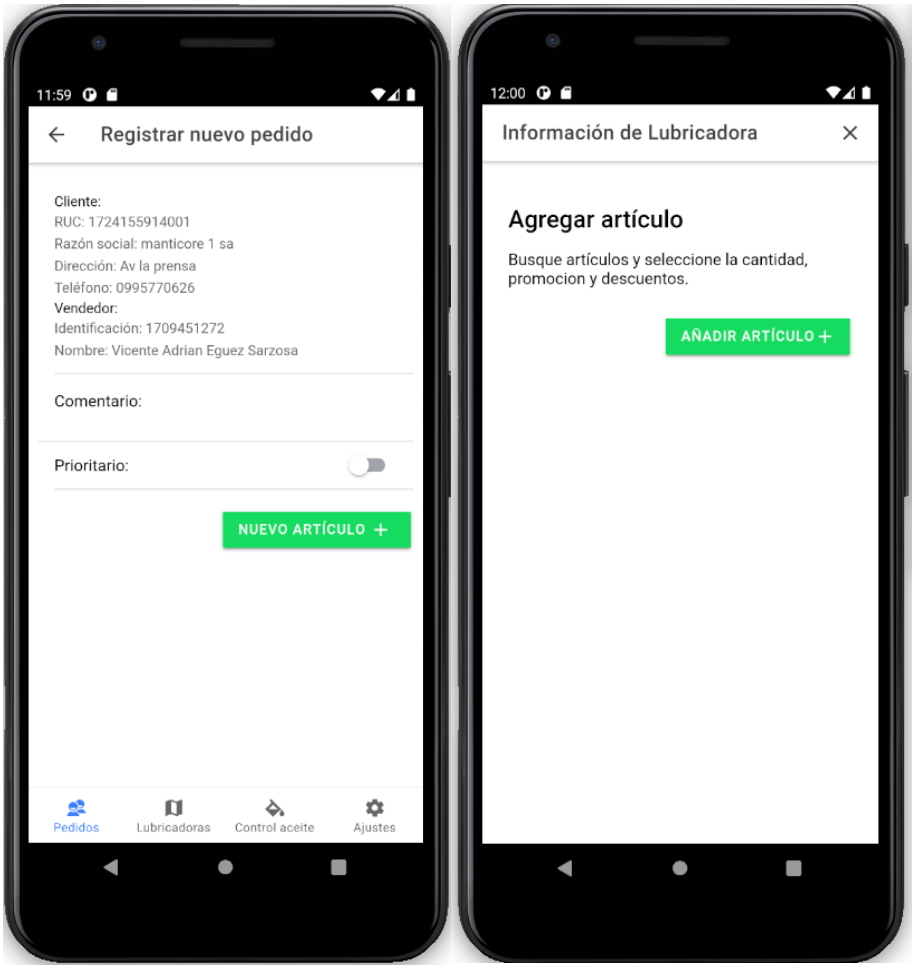


Figura 7. Pantallas para registrar artículo

Cuando se presiona el botón “Añadir Artículo”, se muestra una lista de artículos de la empresa en la cual al presionar alguno de ellos se despliega un formulario donde se realiza el pedido editando la cantidad, promoción y precio como se ve en la Figura 8. El valor total se calcula automáticamente teniendo en cuenta los descuentos que pueden ser agregados.

Los tipos de descuentos pueden ser: cliente, vendedor, y otro.

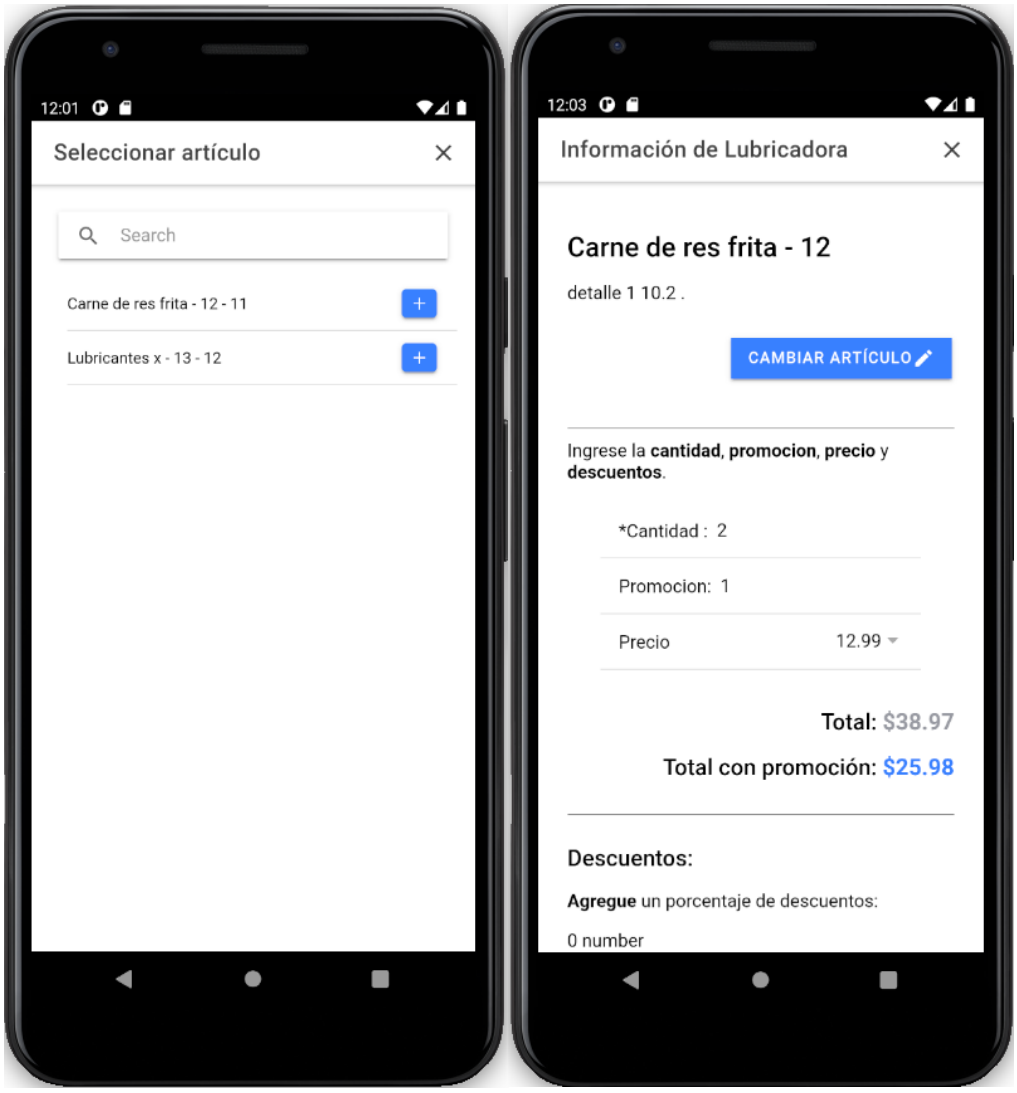


Figura 8. Pantallas añadir artículo

Cuando se completa el pedido de artículos se muestra una pantalla donde el usuario vendedor puede observar el total con descuento, ahorro, total con promociones y total sin promociones como se muestra en la Figura 9Figura 9.

Figura 9. Pantalla de pedido completado

Para la entrega de los artículos el vendedor debe seleccionar “Gestión de Pedidos” de la pantalla izquierda de la Figura 5, seleccionar un pedido realizado previamente. Los pedidos tienen estados de “Pendiente”, “Creado”, “Parcial”.

La pantalla de gestión de pedidos se puede ver en la Figura 10.

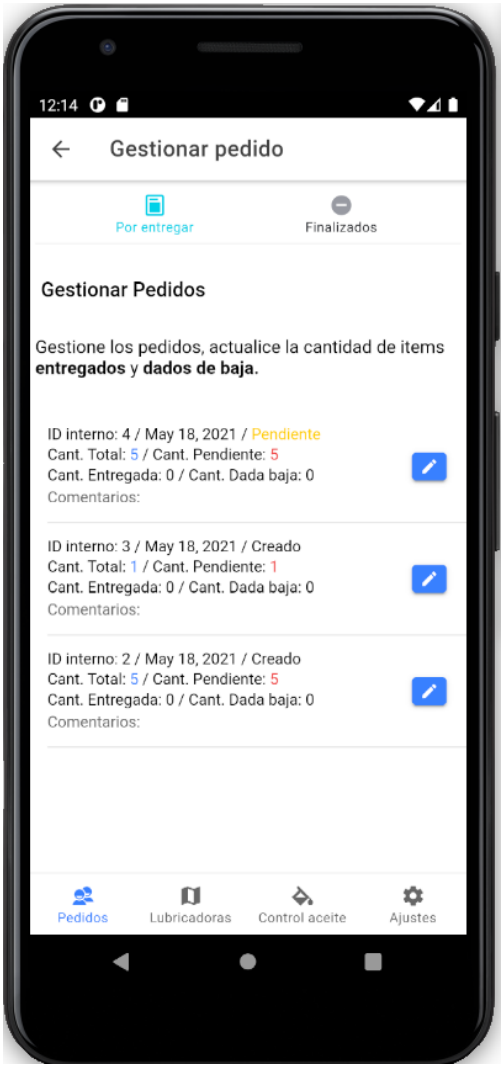


Figura 10. Pantalla de pedidos creados

Las acciones que se puede realizar en los pedidos son “Dar de baja” y “Entregar”. Para dar de baja a un pedido existen dos posibilidades; la primera es dar de baja sin entregar artículos, la segunda es entregar todos los productos sin dejar pendientes. Esta última se realiza especificando la cantidad total que se ha pedido como se indica en la pantalla derecha de la Figura 11.

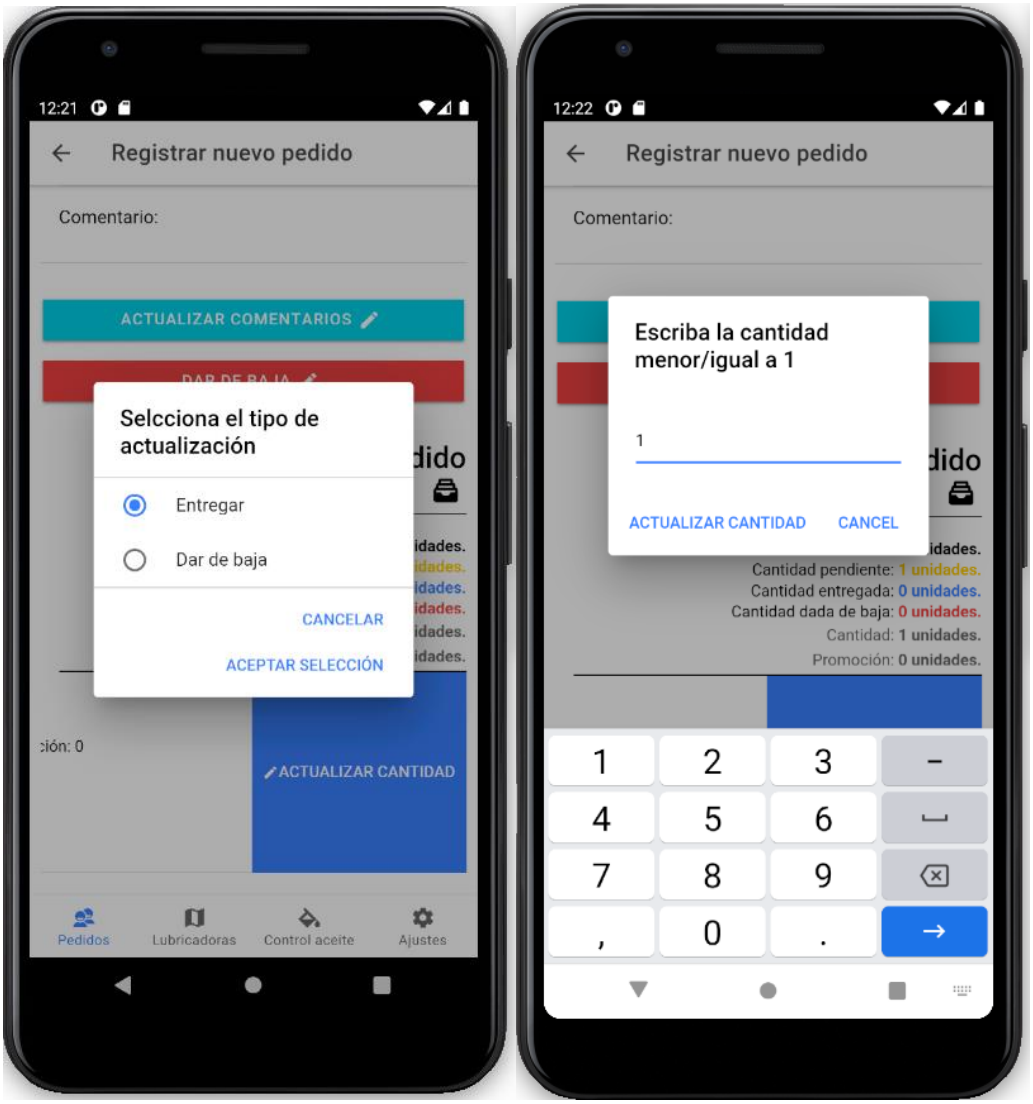


Figura 11. Pantalla dar de baja y entregar artículos

El pedido tomará el estado de “Pendiente” para el caso de un vendedor que no tiene los artículos que se le ha pedido. Por tanto, el usuario ingresa un valor menor a la cantidad del pedido. Posteriormente, para resolver este conflicto el vendedor toma la acción de

actualizar la cantidad faltante de artículos, y el servicio pueda darse de baja como indica la pantalla izquierda de la Figura 12. El pedido que se crea igualando la cantidad de pedido con la cantidad de artículos entregados muestra un mensaje de color verde el cual indica que el pedido puede darse de baja como se muestra en la pantalla derecha de la Figura 12.

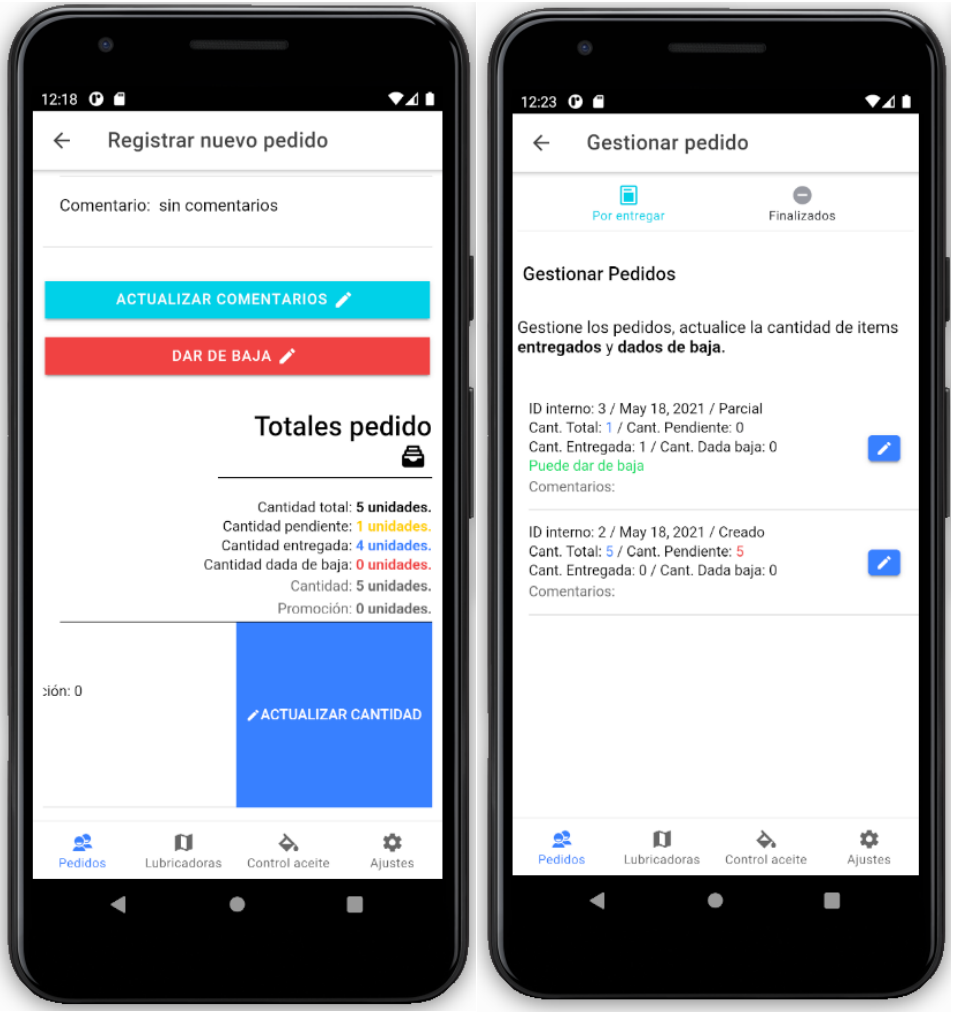


Figura 12. Pantalla actualizar entrega y lista de pedidos

2.1.4. Módulo recaudos

La empresa únicamente contaba con mockups para el flujo de recaudos. Dicho diseño fue modificado en base a la nueva librería con propiedad intelectual de la empresa con la que se desarrollará el aplicativo que será mostrado más adelante. A continuación, se revisa el diseño del módulo de recaudos y se realiza una breve descripción del flujo de recaudos.

El diseño empieza con una pantalla donde el usuario debería seleccionar el botón de seleccionar abonos o seleccionar factura como se muestra en la Figura 13 Figura 13 de lado izquierdo. Previamente el usuario debería tener registros de abonos. En caso de no tenerlos registrar uno nuevo en el formulario como se muestra en la parte derecha de la Figura 13.

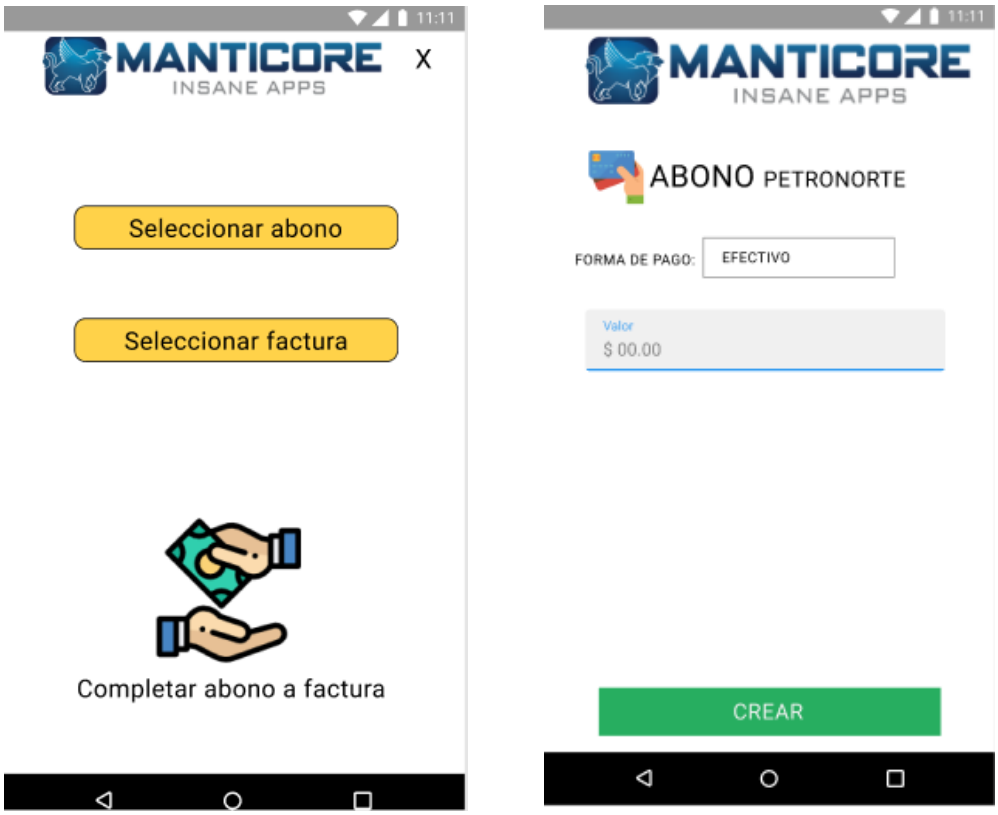


Figura 13. Mockups pantalla recaudos y registro de abono

Una vez creados los abonos, el usuario seleccionaría el botón de seleccionar factura y se mostraría una lista de facturas. En la Figura 14 se muestra cómo se iba a asignar a la factura los recaudos previamente creados. Los recaudos asignados a las facturas se mostrarían en la parte inferior tal como se muestra en la Figura 14Figura 14 en el lado derecho.

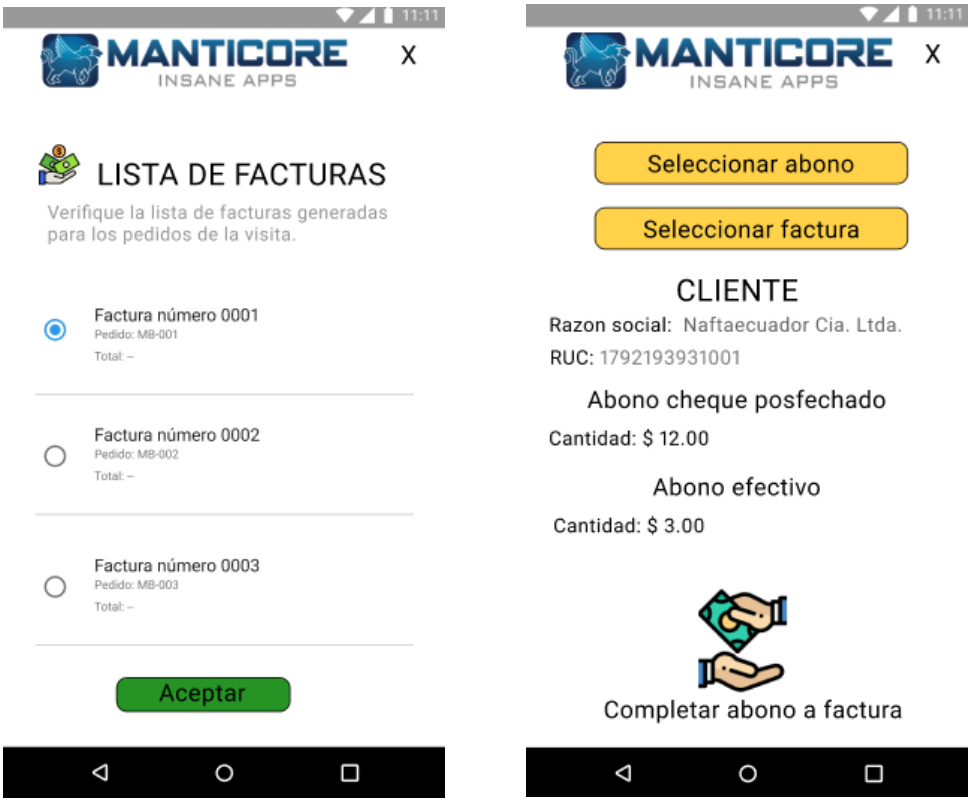


Figura 14. Mockups de selección de factura y lista de recaudos creados

2.2. DEFINICIÓN DE LOS PASOS DEL MODELO GENERAL DE REINGENIERÍA A REALIZARCE

Tomando como base el modelo general de reingeniería presentado en el marco teórico, se decidió que es necesario utilizar los siguientes pasos:

- **Ingeniería inversa:** en este paso se abstraen los requerimientos de la aplicación, y se añaden requerimientos nuevos del interesado.
- **Mejoramiento de la estructura:** en este paso se evalúa y mejora la estructura de la aplicación, además se añaden nuevos requerimientos.
- **Modularización del programa:** en este paso se divide la aplicación en módulos de acuerdo con la funcionalidad.
- **Reingeniería de datos:** en este paso se realiza la revisión y mejora de la base de datos.

Durante los procesos de la reingeniería se llevó a cabo el flujo presentado en la Figura 15 para la aceptación de propuestas, teniendo como resultado las mejoras que fueron aceptadas por el Producto Owner que se detallan más adelante en el punto 2.4.2. Cabe recalcar que no se propuso la traducción del código fuente, pues la aplicación actual está implementada en TypeScript, mismo lenguaje a utilizar en la reprogramación.

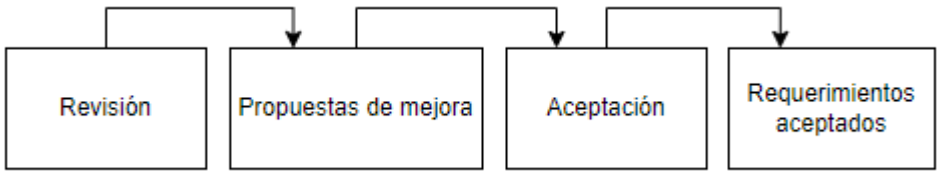


Figura 15. Proceso de aceptaciones de propuestas

2.2.1. Ingeniería inversa

Definición de Historias épicas

Al revisar la aplicación se infirieron Historia épicas que después serán detalladas para volver a documentar la aplicación que se catalogó con historias épicas. Adicionalmente, se agregó la historia épica para el submódulo de recaudos.

Para la codificación de las historias épicas se utilizó la siguiente nomenclatura:

- **HE - CÓDIGO:** historia épica.
- **HE - CÓDIGO -** #: historia normal.

En la Tabla 8 se puede ver a detalle la descripción de la nomenclatura, y los códigos utilizados para las historias de usuario.

Tabla 8. Tabla de historias épicas

Código	Título	Prioridad
HE-O	Otras	Media
HE-VE	Desarrollo del submódulo de Ventas	Alta
HE-IN	Desarrollo del submódulo de Inventario	Alta
HE-RE	Desarrollo del submódulo de Recaudos	Alta

2.2.2. Mejoramiento de la arquitectura

Revisión de la arquitectura

El aplicativo anterior utilizaba una arquitectura MVC la cual fue cambiada por MVVM para reducir el número de peticiones al servidor. El modelo anterior realiza peticiones por cada acción que el usuario vendedor realiza, es decir, que necesitaba conexión a internet para realizar pedidos, y ventas. La Figura 16 muestra a un usuario que representa a un vendedor, el cual realiza peticiones para obtener el inventario, pedidos y facturas por cada cliente, lo cual hace que la aplicación dependa demasiado de la conexión a internet y esta se vuelva lenta.

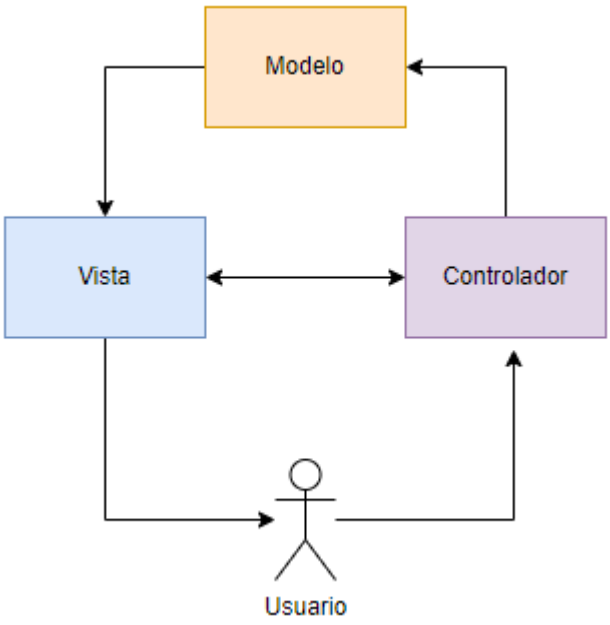


Figura 16. Arquitectura MVC

Como se muestra en la Figura 17 se utiliza una arquitectura basada en el modelo-vista-vista-modelo [14] , adicionalmente se especifican los frameworks a utilizarse en cada componente de la arquitectura y su interacción.

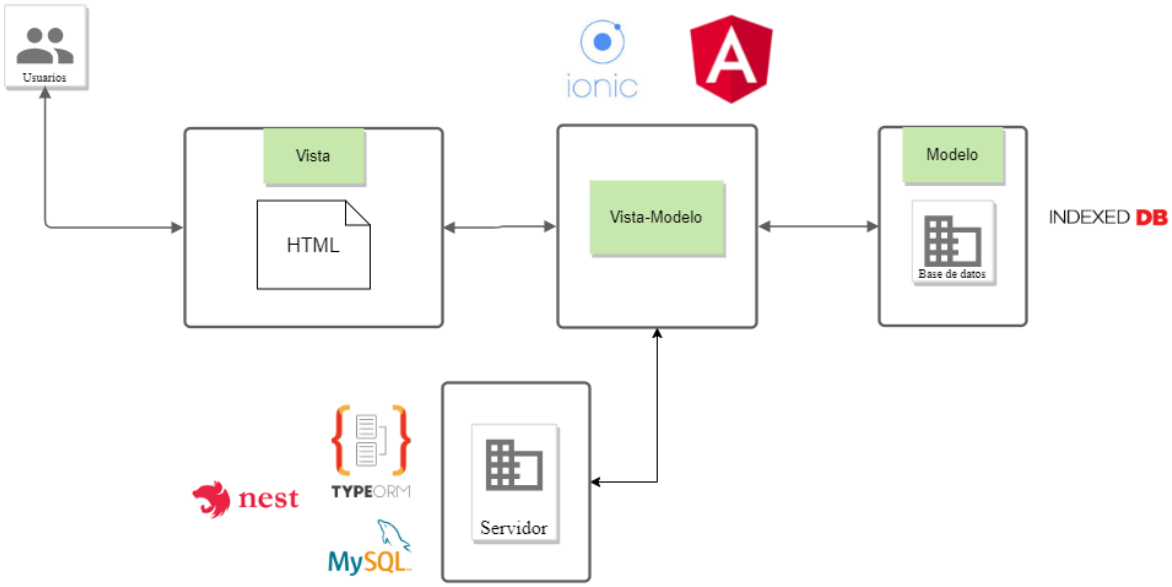


Figura 17. Estructura del software

La nueva arquitectura evita realizar peticiones al servidor. Los datos necesarios para el funcionamiento del aplicativo se consultan con una sola petición, la cual trae toda la información necesaria y la almacena en una base de datos local, para su posterior uso en los flujos de pedidos, recaudos e inventario. La información que contiene la base de datos local tiene información sobre todos los clientes que el usuario vendedor debe visitar. La documentación del software no incluye el diagrama de la arquitectura, por lo que se procedió a realizar el diagrama.

2.2.3. Modularización del programa

Una modularización adecuada ayuda a tener una visión a largo plazo en la escalabilidad y mantenimiento de las aplicaciones cuando en ellas se requiera implementar nuevas funcionalidades o cambios; por tal motivo, se usó la estructura LIFT donde la distribución de carpetas permite la ubicación del archivo que contiene el código rápidamente. Por lo tanto, cada módulo separa la complejidad de otros [30].

Estructura de carpetas de desarrollo

En la documentación del framework Angular [30] contiene una guía de codificación que ayudó a escoger la estructura LIFT. El estilo 04-02 indica que se debe ubicar archivos en una ubicación cuando éstos estén relacionados. La relación que se utilizó fue por funcionalidad y tipo. Otro punto para destacar es el nivel de carpetas, el estilo 04-04 y 04-07 indica que no debe de sobrepasar los 7 niveles y crear carpetas con nombres adecuados dependiendo de su contenido que contengan no más de 7 archivos.

En el caso de formularios esta carpeta puede contener más de 10 archivos. Para ello, se llamó precediendo el nombre del módulo y la palabra formulario para saber que se trata de formularios. Los módulos facilitan el aislamiento y reutilización de componentes.

Las carpetas de componentes reflejan el estilo 04-10 que indica que se debe crear una carpeta Shared en donde se declara componentes, directivas, pipes. Esta carpeta contiene componentes que pueden ser utilizados en cualquier parte del proyecto. Dentro de cada submódulo existe una de ellas ya que contienen componentes únicamente serán utilizados en dicho modulo. Y esto conlleva a utilizar el estilo 04-11 que indica que se use la carga perezosa la cual carga un módulo bajo demanda; es decir, cuando se requiera cargar dicho módulo será cargado.

Dicho esto, la modularización del aplicativo tanto backend, frontend y móvil presenta una distribución de carpetas similares donde se alojan módulos tanto para formularios, servicio, modales y componentes como se muestra en la Figura 18.

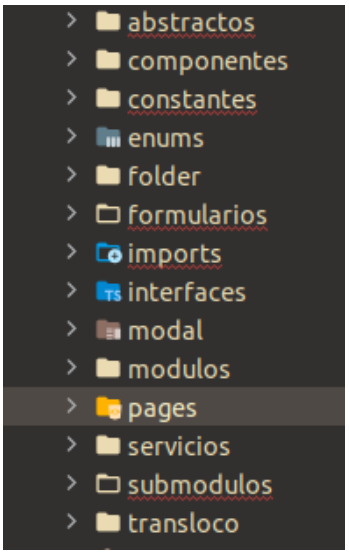


Figura 18. Estructura de las carpetas del proyecto

2.2.4. Reingeniería de datos

Revisión de la base de datos

La base de datos está dividida en submódulos, para este proyecto se revisó los submódulos de Inventario, Vendedores y Recaudos, que son los que nos caben en el alcance del proyecto.

La base de datos original se puede revisar en el Anexo 1.

Cambios a la base de datos

Los cambios realizados en su mayoría fueron en el submódulo de inventario, el mismo que fue rediseñado para facilitar los procesos de inventario, además se incluyó las entidades necesarias para manejar facturación y pedidos. Los cambios a los submódulos de vendedores, y recaudos, son principalmente eliminar relaciones que tenían con entidades del anterior diseño de submódulo de Inventario y añadir relaciones con las entidades del nuevo diseño. Los cambios resumidos se detallan en la Figura 19 y Figura 20, y el diseño original se muestra en Anexo1.

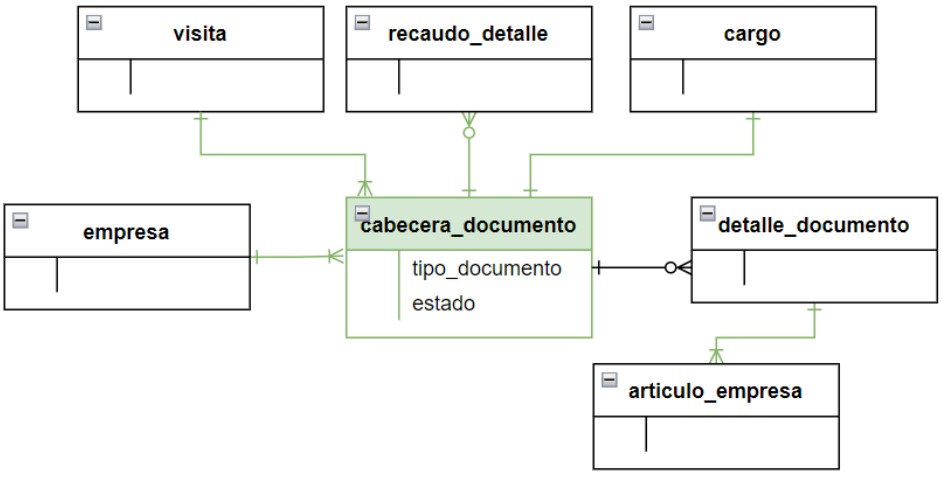


Figura 19. Cambios en base datos

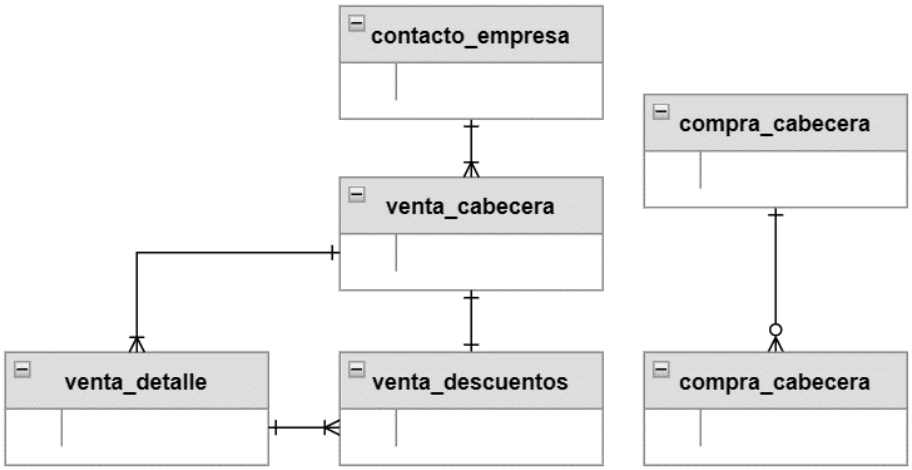


Figura 20. Base de datos antigua

Cambios de base de datos en submódulo vendedores

Se relacionó la tabla de visita con cabecera_documento de la Figura 19, para conocer fácilmente los datos de pedidos y abonos de una visita que son necesarios para el proceso de recaudación y pedidos en la aplicación móvil. Las tablas cabecera_documento y cabecera_detalle pertenecen al submódulo de inventarios.

Cambios de base de datos en submódulo inventario

El submódulo de inventario es un rediseño del submódulo de pedidos que contaba con un diseño de compras, ventas, pero no estaba integrado con el submódulo de recaudos. La integración del submódulo de recaudos al submódulo de inventario sirvió para facilitar los procesos de inventario, facturación y pedidos. La tabla cabecera_documento y detalle_documento de la Figura 19 reemplazan a venta_cabecera y venta_detalle de la Figura 20.

La integración con el submódulo de recaudos a través de las relaciones como se muestra en la Figura 19, entre las entidades cabecera_documento con cargo, y recaudo_detalle permite realizar abonos. A la tabla de cabecera_documentos se le añadió una lógica para determinar el tipo de documento que puede ser pedido o factura. Cuando se hace un pedido la cabecera_documento tendrá un estado de pedido, al momento de hacer entregas totales de los productos se cambiará a pendiente, y el tipo de documento se cambia a factura, cuando se termine de pagar una factura cambia a estado pagado.

La Figura 19 muestra la relación entre cabecera_documento con empresa, la cual permite saber de manera más eficiente a que cliente le pertenece un pedido o factura, pues anteriormente se podía conocer está información consultando varias relaciones con entidades externas a este submódulo. Finalmente se relacionó documento_detalle con articulo_empresa para saber que artículos pertenecen en ese pedido o factura.

Cambios en base de datos en submódulo recaudos

En la Figura 20 la entidad venta_cabecera funcionaban para registrar ventas, pero no estaba relacionada con el submódulo de recaudos. Ahora la entidad documento _cabecera se encarga de almacenar pedidos o ventas y se relaciona con las entidades recaudo_detalle y cargo como se muestra en la Figura 19. Las entidades recaudo_detalle y cargo manejan un historial de abonos que tiene el cliente y permite cambiar los tipos de documentos de la entidad cabecera_documento.

Todos los cambios realizados a la base de datos se pueden revisar en el Anexo 2.

2.3. IMPLEMENTACIÓN DE LOS FLUJOS DE INTEGRACIÓN Y DESARROLLO CONTINUO

Para la implementación de los flujos de integración y desarrollo continuo se utilizó Gitlab CI/CD, que es una herramienta de desarrollo de software que utiliza metodologías continuas [31].

Las etapas implementadas fueron:

- **Test:** en esta etapa se ejecuta el Linter para asegurar que el código cumple con los estándares establecidos por el Product Owner. Esta configuración se encuentra en el archivo tslint.json del proyecto.
- **Build:** en esta etapa se realiza el build del aplicativo, que, a continuación, será desplegado.
- **Deploy:** en esta etapa se realizan las acciones necesarias para el despliegue del aplicativo a pruebas y producción.

El Linter se ejecutó en la rama desarrollo, para evitar que cualquier error sea enviado a test o producción. El build de prueba se realizó usando la rama fear-test, y para el build de producción se usó la rama master.

Para la configuración, las figuras a continuación explican el proceso.

En el menú del repositorio se accedió a las configuraciones de CI CD, y se habilitó la opción Auto DevOps como se muestra en la Figura 21.

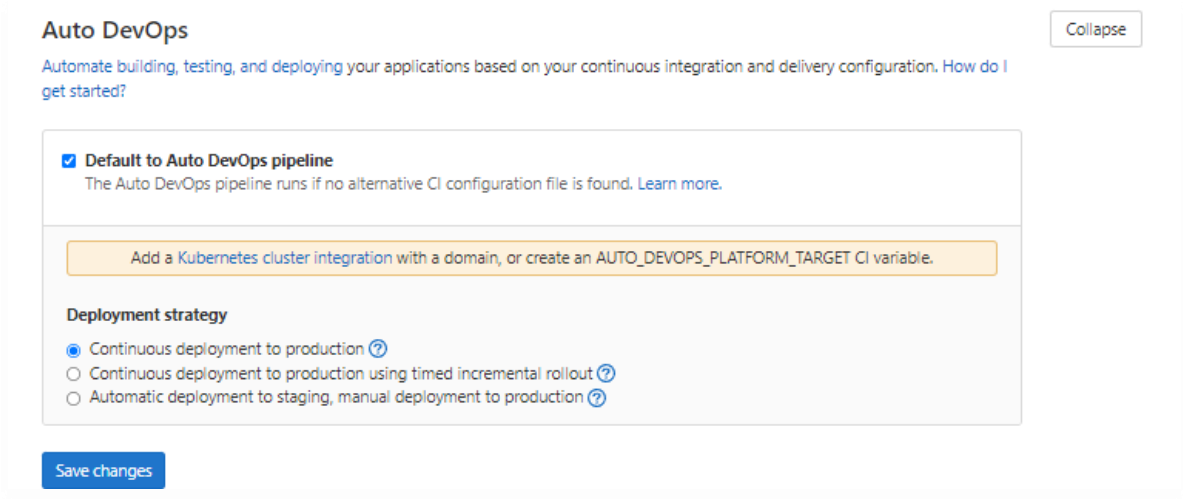


Figura 21. Configuración Auto devOps

Para añadir el archivo gitlab-ci.yml se utilizó la herramienta de edición en línea como se puede ver en la Figura 22. Este archivo contiene los pasos que se automatizarán al realizar cada commit en la rama master del proyecto. En el Anexo 3. Se puede revisar el archivo completo.

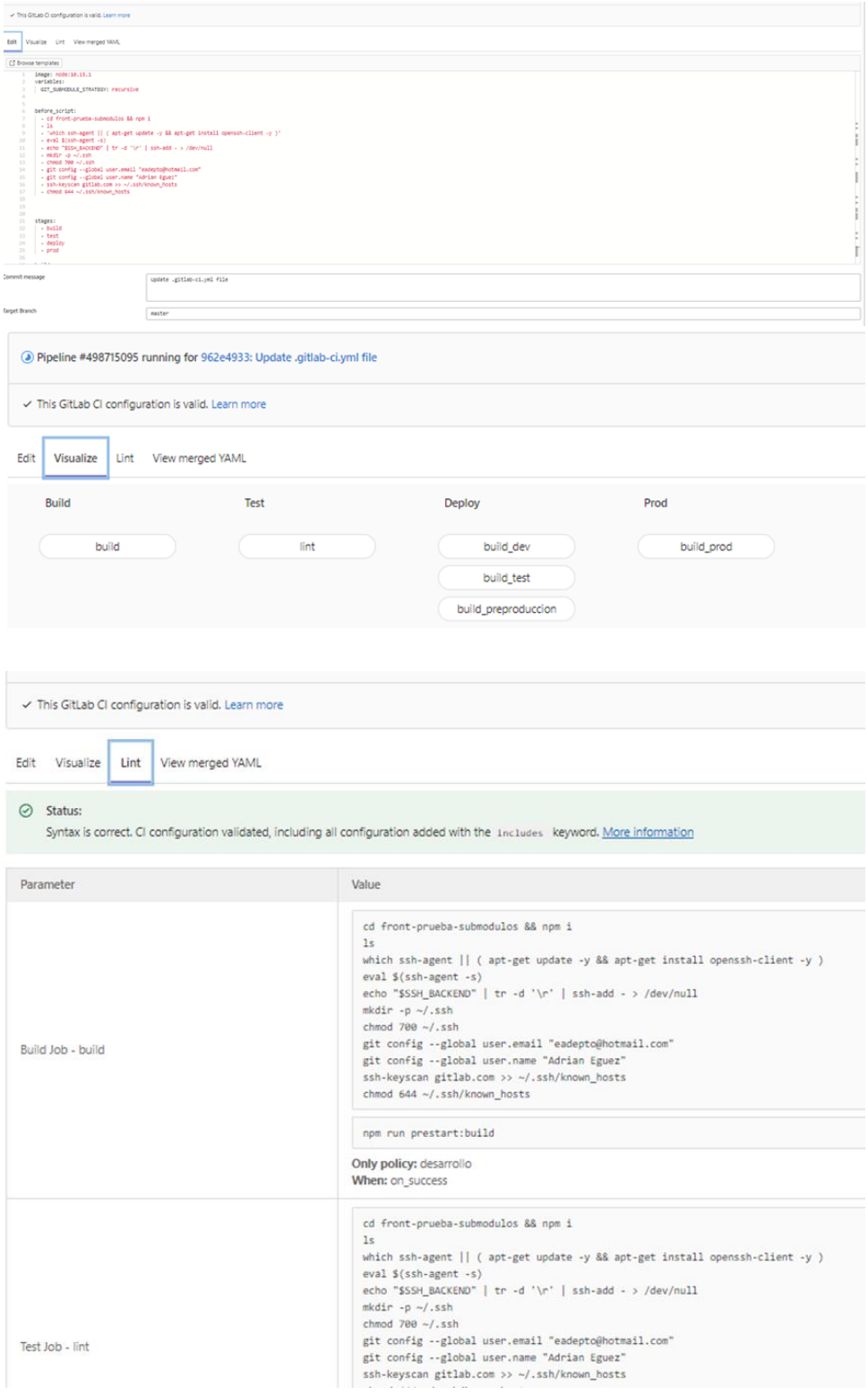


Figura 22. Editor el linea gitlab-ci.yml

2.4. APLICACIÓN DE LA METODOLOGÍA DE DESARROLLO

2.4.1. Definición de roles

Tomando en cuenta el marco de trabajo Scrum se establecieron los roles de la siguiente manera:

- Product Owner**:** Ing. Cristian Lara.
- Scrum Master: MSc. Adrián Egüez.
- **Equipo de desarrollo:**
    - Israel Pérez
    - Anderson Revelo

2.4.2. Pila de producto (Product Backlog)

Las historias épicas necesarias para el desarrollo de la aplicación fueron definidas la sección 2.2.1. En las siguientes tablas se presentan las historias necesarias para la implementación del proyecto.

Tabla 9. Historia de usuario HE-VE-01

HISTORIA DE USUARIO	**HE-VE-01**
Título: Mostrar cronograma del vendedor	
Descripción: Como vendedor, necesito ver el cronograma del día, para conocer la ruta que realizaré y saber los clientes que debo visitar.	
Prioridad: Media	**Peso:** 3

Tabla 10. Historia de usuario HE-VE-02

HISTORIA DE USUARIO	**HE-VE-02**
Título: Mostrar ruta del vendedor	
Descripción: Como vendedor, necesito ver mi ruta de visitas, para saber la zona a la que debo movilizarme.	
Prioridad: Media	**Peso:** 3

Tabla 11. Historia de usuario HE-VE-03

HISTORIA DE USUARIO	**HE-VE-03**
Título: Mostrar cliente en la ruta	
Descripción: Como vendedor, necesito ver los clientes en la ruta, para poder visitarlos en sus establecimientos.	
Prioridad: Media	**Peso:** 2

Tabla 12. Historia de usuario HE-IN-01

HISTORIA DE USUARIO	**HE-IN-01**
Título: Gestionar pedidos de un cliente	
Descripción: Como vendedor, necesito listar, crear, buscar pedidos de manera online y offline.	
Prioridad: Alta	**Peso:** 5

Tabla 13. Historia de usuario HE-IN-02

HISTORIA DE USUARIO	**HE-IN-02**
Título: Gestionar entregas a un cliente	
Descripción: Como vendedor, necesito entregar, dar de baja artículos del pedido, y finalizar el pedido al completar la entrega de manera online y offline.	
Prioridad: Alta	**Peso:** 8

Tabla 14. Historia de usuario HE-IN-03

HISTORIA DE USUARIO	**HE-IN-03**
Título: Añadir descuentos	
Descripción: Como vendedor, necesito añadir descuentos a los artículos en venta de manera online y offline.	
Prioridad: Media	**Peso:** 2

Tabla 15. Historia de usuario HE-IN-04

HISTORIA DE USUARIO	**HE-IN-04**
Título: Revisar stock de clientes	
Descripción: Como vendedor, necesito conocer el stock de mis clientes de manera online y offline, para poder saber cuándo un cliente va a necesitar artículos.	
Prioridad: Media	**Peso:** 2

Tabla 16. Historia de usuario HE-RE-01

HISTORIA DE USUARIO	**HE-RE-01**
Título: Gestionar los recaudos	
Descripción: Como vendedor, necesito recaudar dinero de las ventas realizadas previamente a clientes, facilitando la repartición de los pagos a diferentes facturas mediante pagos en efectivo o depósito de manera online y offline.	
Prioridad: Alta	**Peso:** 13

Tabla 17. Historia de usuario HE-O-01

HISTORIA DE USUARIO	**HE-O-01**
Título: Visualizar información del cliente	
Descripción: Como vendedor, necesito conocer la información del cliente.	
Prioridad: Baja	**Peso:** 2

2.4.3. Planificación de los Sprints

En la Tabla 18 se puede ver la planificación de sprints con su respectivo sprint backlog.

Tabla 18. Tabla de planificación de Sprints

Planificación de Sprints	
SPRINT 1	HE-VE-01 HE-VE-02
SPRINT 2	HE-VE-03 HE-IN-01
SPRINT 3	HE-IN-02 HE-IN-03 HE-RE-01
SPRINT 4	HE-O-01 HE-O-02

- **Sprint 1**

Objetivos del Sprint

Implementar las pantallas en el aplicativo móvil necesarias para mostrar los cronogramas del vendedor con sus respectivos detalles, con las rutas, y los clientes que debe visitar, estos últimos en un mapa para facilidad del vendedor.

Implementar las gestiones de cronograma, cronograma detalle, ruta, ruta cliente en el aplicativo web de la empresa para el funcionamiento de la aplicación móvil.

Revisión del sprint

La Tabla 19 muestra la revisión del sprint 1.

Tabla 19. Tabla de revisión del Sprint 1

Historia de usuario	Descripción	Criterios de aceptación	Cumple
HE-VE-01	Como vendedor, necesito ver el cronograma del día, para conocer la ruta que realizaré y saber los clientes que debo visitar.	Mostrar los cronogramas del vendedor, filtrados según la fecha y el día, estos se pueden seleccionar para avanzar al siguiente paso.	Si
HE-VE-02	Como vendedor, necesito ver mi ruta de visitas, para saber la zona a la que debo movilizarme.	Mostrar información de la ruta, según el cronograma.	Si

- **Sprint 2**

Objetivos del Sprint

Implementar la pantalla en el aplicativo móvil para mostrar los clientes de la ruta en un mapa para facilidad del vendedor. Implementar la gestión de pedido en la aplicación web de la empresa.

Revisión del sprint

La Tabla 20 muestra la revisión del sprint 2.

Tabla 20. Tabla de revisión del Sprint 2

Historia de usuario	Descripción	Criterios de aceptación	Cumple
HE-VE-03	Como vendedor, necesito ver los clientes en la ruta, para poder visitarlos en sus establecimientos.	Mostrar un mapa donde se visualice la ubicación de los establecimientos de los clientes, del cronograma seleccionado, además mostrar la ubicación actual del vendedor.	Si
HE-IN-01	Como vendedor, necesito listar, crear, buscar pedidos de manera online y offline.	Crear un pedido seleccionando el artículo y la cantidad. Mostrar la siguiente información al crear un pedido: Total sin impuestos, cantidad pedida, cantidad entregada, cantidad faltante, cantidad dada de baja.	Si

- **Sprint 3**

Objetivos del Sprint

Implementar descuentos en el formulario de pedidos de la aplicación móvil.

Implementar entrega, dar de baja artículos y completar pedido al terminar de entregar los artículos.

Implementar las gestiones de recaudo, recaudo detalle, tipo forma pago, y forma pago detalle, en la aplicación web de la empresa.

Revisión del sprint

La Tabla 21 muestra la revisión del sprint 3.

Tabla 21. Tabla de revisión del Sprint 3

Historia de usuario	Descripción	Criterios de aceptación	Cumple
HE-IN-02	Como vendedor, necesito entregar, dar de baja artículos del pedido, y finalizar el pedido al completar la entrega de manera online y offline.	Se puede entregar, dar de baja artículos.	Si
HE-IN-03	Como vendedor, necesito realizar ventas añadiendo descuentos a los artículos de manera online y offline	Visualizar un campo extra llamado porcentaje de descuento en el formulario de creación de pedido que modifique el valor a pagar de los productos.	Si
HE-RE-01	Como vendedor, necesito recaudar dinero de las ventas realizadas previamente a clientes, facilitando la repartición de los pagos a diferentes facturas mediante pagos en efectivo o transferencias de manera online y offline.	Visualizar una ventana donde se muestren las facturas generadas del cliente. La información de las facturas deberá contener la siguiente información: total, deuda. Cada factura deberá contener los siguientes métodos de pago: Efectivo y Deposito. Al seleccionar una forma de pago se mostrará un popup que permita especificar el monto a abonar.	Si

- **Sprint 4**

Objetivos del Sprint

Implementar descuentos en el formulario de pedidos de la aplicación móvil.

Implementar entrega, dar de baja artículos y completar pedido al terminar de entregar los artículos.

Implementar las gestiones de recaudo, recaudo detalle, tipo forma pago, y forma pago detalle, en la aplicación web de la empresa.

Revisión del sprint

La Tabla 22 muestra la revisión del sprint 4.

Tabla 22. Tabla de revisión del Sprint 4

Historia de usuario	Descripción	Criterios de aceptación	Cumple
HE-IN-04	Como vendedor, necesito conocer el stock de mis clientes de manera online y offline, para poder saber cuándo un cliente va a necesitar artículos.	Mostrar el stock de los artículos de un cliente	Si
HE-O-01	Como vendedor, necesito conocer la información del cliente.	Mostar la siguiente información del cliente: Razón social, teléfono, dirección y ruc.	Si

3. RESULTADOS Y DISCUSIÓN

3.1. Descripción del producto final

3.1.1. Aplicación móvil PWA

Al iniciar el aplicativo móvil PWA, y luego de que un vendedor haya ingresado con sus credenciales, se muestra la pantalla de “Cronogramas” como se ve en la Figura 23, donde se visualiza información de los cronogramas que tiene asignados, estos cronogramas están filtrados por el día de la semana y la fecha actual, pues estos cronogramas pueden ser de dos tipos:

- **Fecha:** cronograma que se asigna solo para una fecha específica.
- **Día:** cronograma que se asigna a un día de la semana.

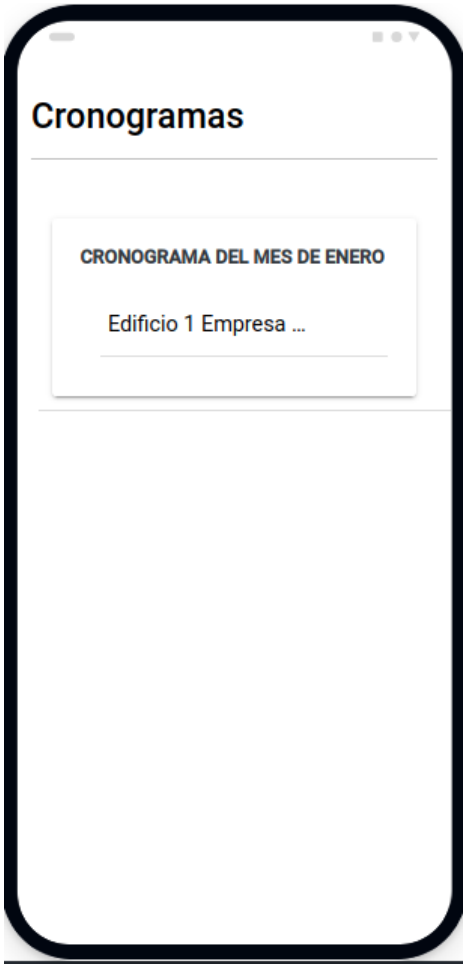


Figura 23. Pantalla cronogramas - aplicativo móvil

Nota: El aplicativo al funcionar como un PWA debe de tener una conexión anticipada para traer toda la información necesaria. Esta información contiene: Cronogramas del vendedor, información del cliente, artículos de bodega, stock del cliente, facturas y pedidos del cliente.

Al seleccionar el cronograma se muestra la pantalla del “Mapa” como en la Figura 24, que contiene un mapa, el cual indica la posición actual del vendedor, y la posición de los clientes que tiene que visitar.

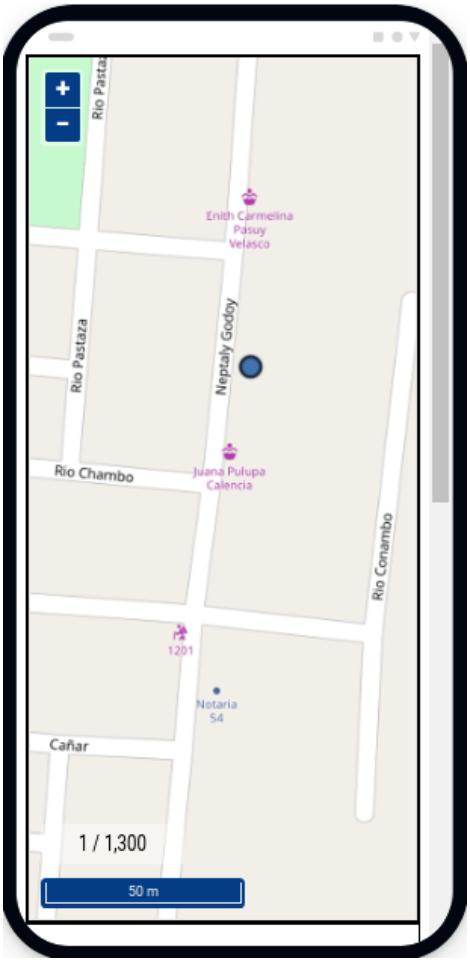


Figura 24. Pantalla mapa – aplicativo móvil

Al seleccionar un cliente en el mapa, se muestra un menú de opciones llamada Gestión Cliente, así como en la Figura 25, en ella se visualiza la información del cliente y las opciones: Empezar visita y Revisar Stock.

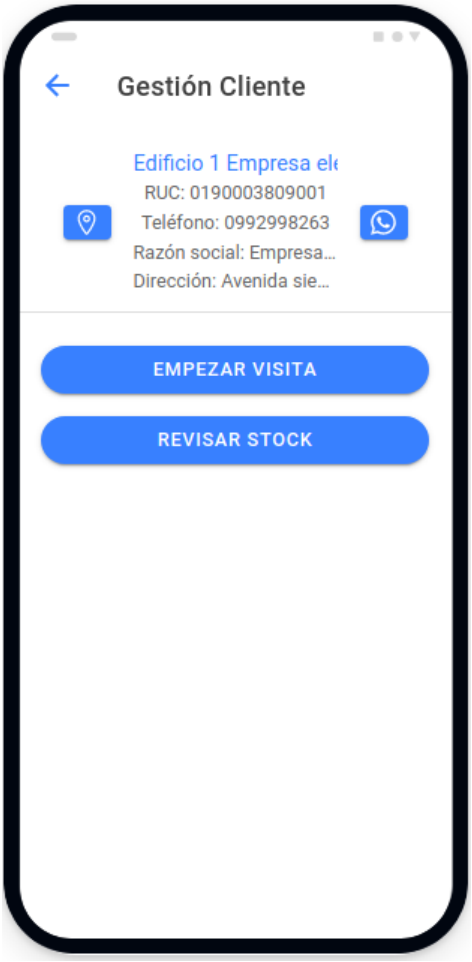


Figura 25. Pantalla gestión cliente – aplicativo móvil

Al seleccionar la opción “Empezar visita” se muestra la pantalla de “Visita”, que se puede ver en la Figura 26, con el resumen de la deuda del cliente, la cantidad de pedidos y recaudos registrados en la visita, adicionalmente se tienen las opciones para realizar pedidos y recaudos.

Nota: La información de deuda total, total de recaudos y pedidos realizados se actualizan conforme el vendedor va realizando pedidos y recaudos. Esta información actualiza cuando vuelva a tener conexión a internet.

Figura 26. Pantalla visita – aplicativo móvil

Cuando el vendedor se encuentre en la pantalla de pedidos podrá crear nuevos pedidos escogiendo el producto, ingresando la cantidad y opcionalmente el descuento. Adicionalmente, se listan los pedidos pendientes del cliente en el caso de tenerlos. La Figura 27 muestra la pantalla de pedidos.

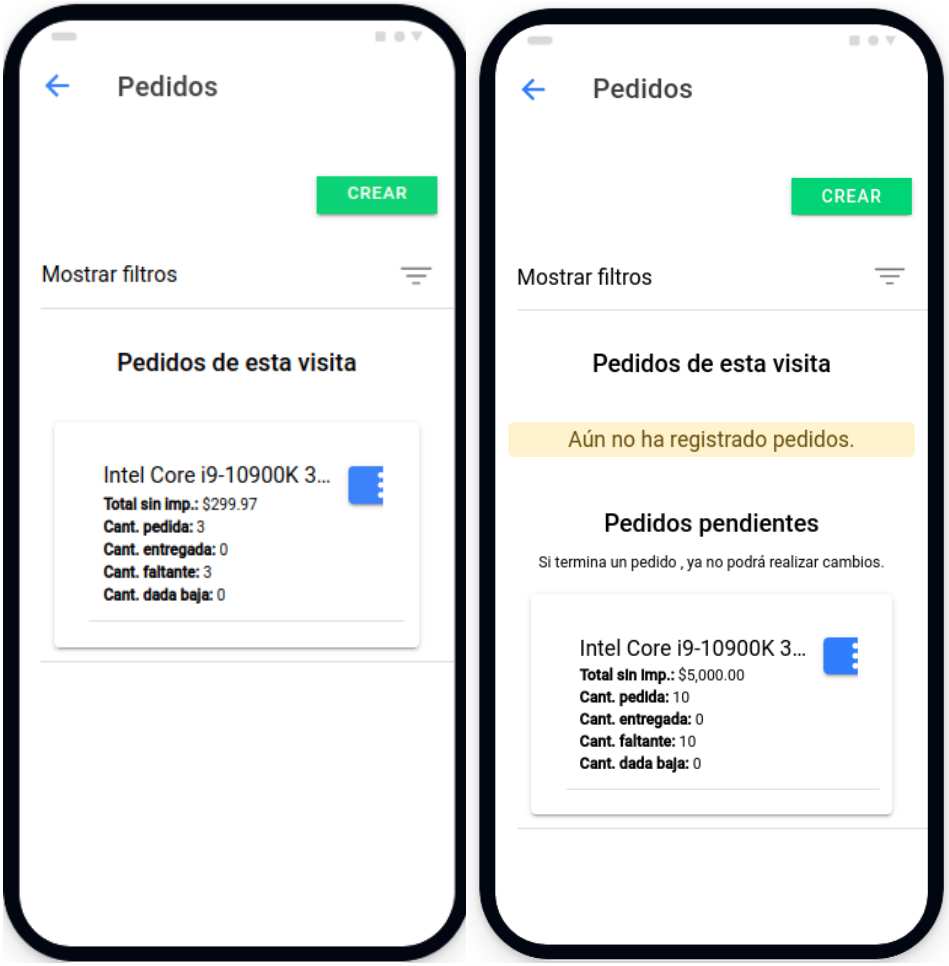


Figura 27. Pantalla pedidos – aplicativo móvil

En la Figura 28 se muestran las acciones entregar, dar de baja y eliminar las cuales se son posibles únicamente cuando un pedido se ha registrado. Para los pedidos pendientes se tiene la opción de reestablecer en lugar de eliminar.

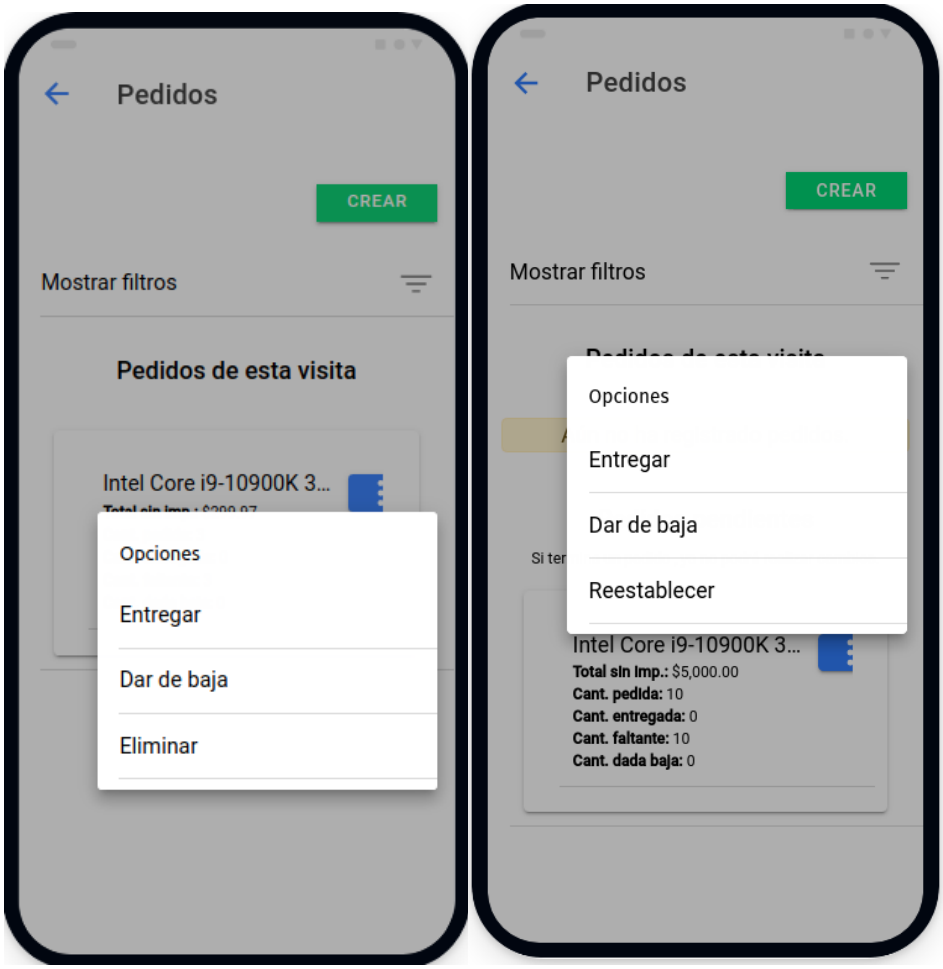


Figura 28. Opciones de pedidos – aplicativo móvil

La opción Dar de baja de un pedido nuevo y la opción Reestablecer de un pedido pendiente muestran un mensaje de confirmación para evitar equivocaciones del usuario Figura 29.

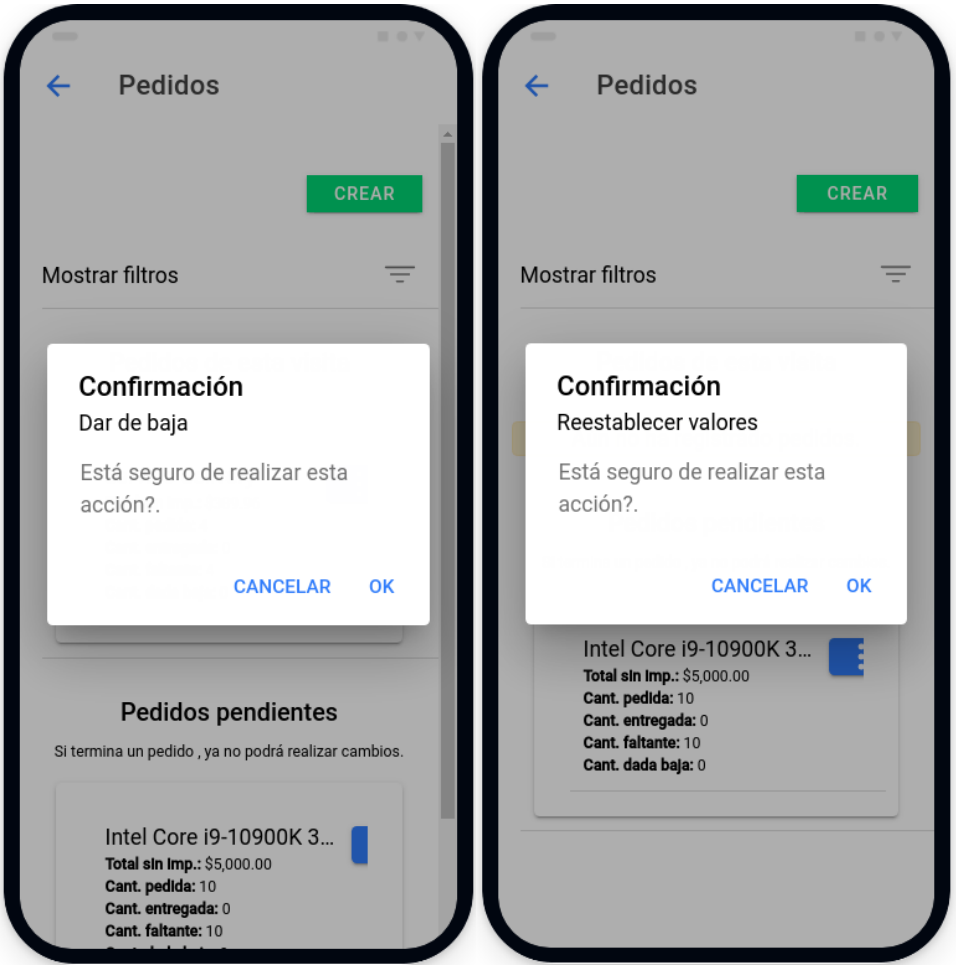


Figura 29. Mensajes de confirmación dar de baja y restablecer - aplicativo móvil

Al entregar todos los productos de un pedido se genera una factura, la cual se puede visualizar al escoger la opción “Recaudos” en la pantalla de “Visita”. Las facturas tienen la opción de abonar mediante dos formas de pago, las cuales puede ser efectivo o depósito, como se muestra en la Figura 30.

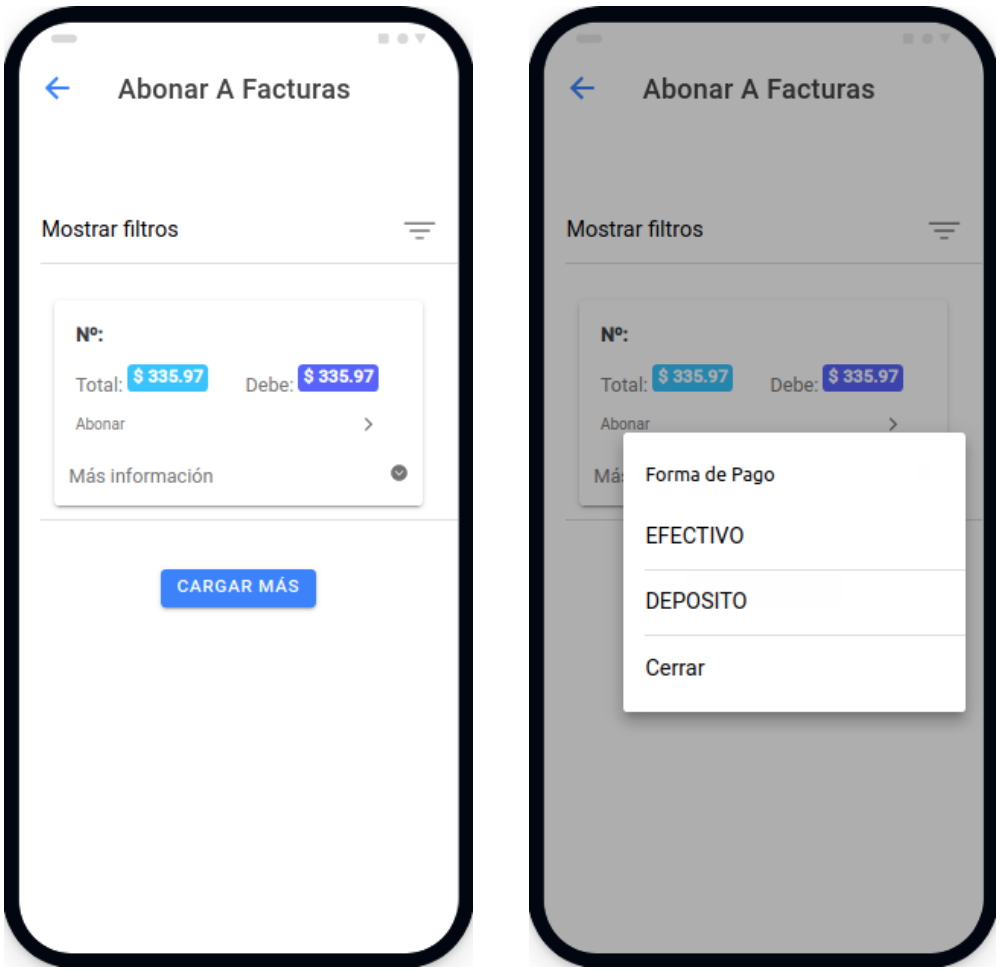


Figura 30. Opciones de abono – aplicativo móvil

Al seleccionar la opción “Abonar”, se muestran las opciones de pago ya mencionadas, y a continuación un modal para ingresar el monto del abono, como en la Figura 31.

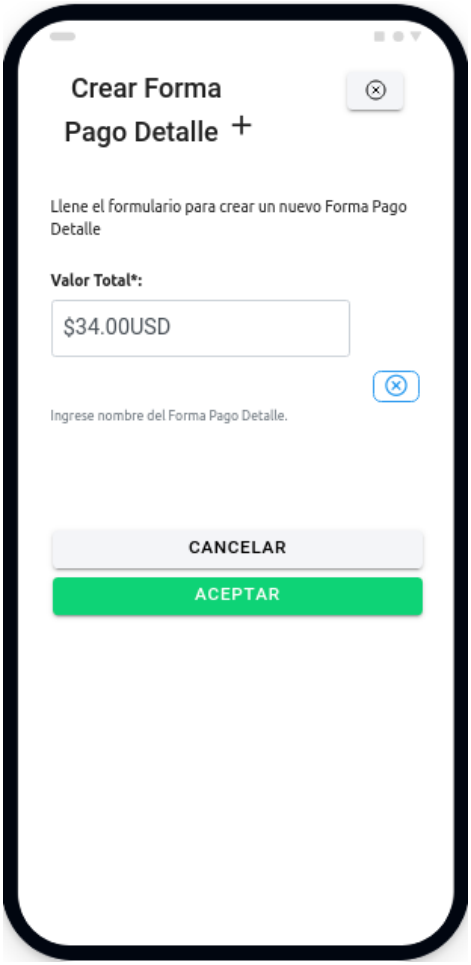


Figura 31. Formulario de abono – aplicativo móvil

Una vez registrado el abono se actualiza el campo Deuda en el resumen de la pantalla de Visita.

Al finalizar la visita desde la pantalla de visita, y si el dispositivo tiene una conexión activa a internet se mostrará el botón “Enviar”, dentro del menú de envío de datos como se muestra en la Figura 32. Al seleccionar este botón todos los datos locales de las visitas se guardarán en el backend usando el servicio web para guardar datos de la aplicación móvil.

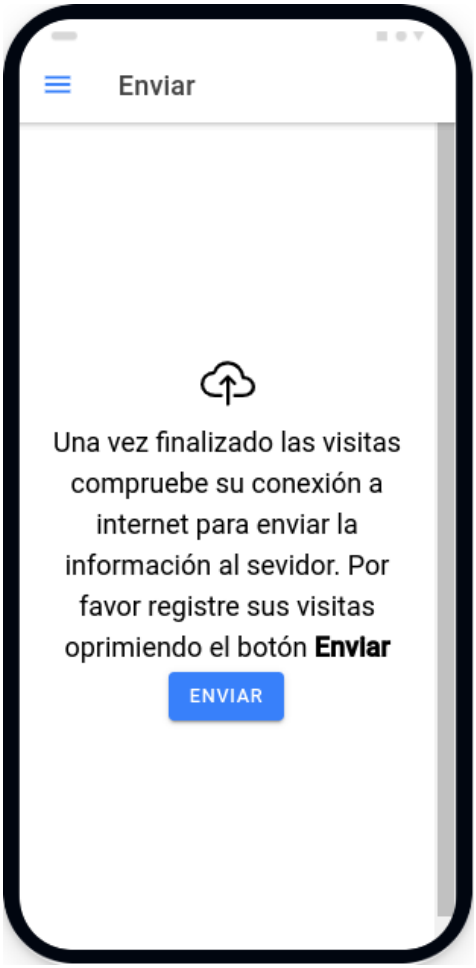


Figura 32. Botón guardar visitas – aplicativo móvil

A continuación, la Figura 33 muestra que la PWA puede ser instalada como una aplicación en un dispositivo, después de aceptar el mensaje de confirmación, y su icono aparece en la pantalla principal.

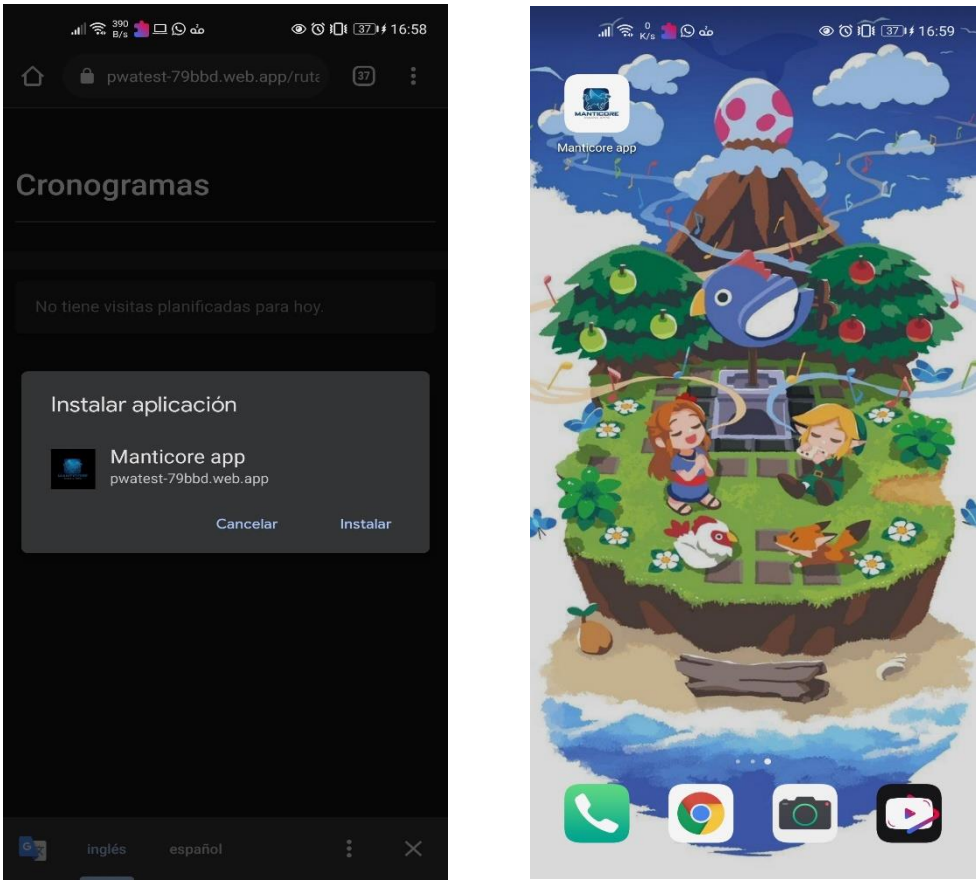


Figura 33. Instalación de la PWA

3.1.2. Aplicación web

Cuando un usuario administrador inicia sesión en la aplicación web permite administrar los datos necesarios para el funcionamiento de la aplicación móvil, en esta se crean usuarios, se asignan roles, se crean clientes, se crean rutas a las que se les asignan clientes, se crean cronogramas a los que se les asignan rutas, y se asignan rutas a vendedores. Adicionalmente, se puede revisar las visitas realizadas por los vendedores, con sus pedidos, y recaudos. A continuación, se muestran algunas de las pantallas de la aplicación web.

La gestión de rutas en la Figura 34 permite crear rutas y asignarles una bodega de la empresa, la cual proveerá los artículos para los pedidos de los clientes.

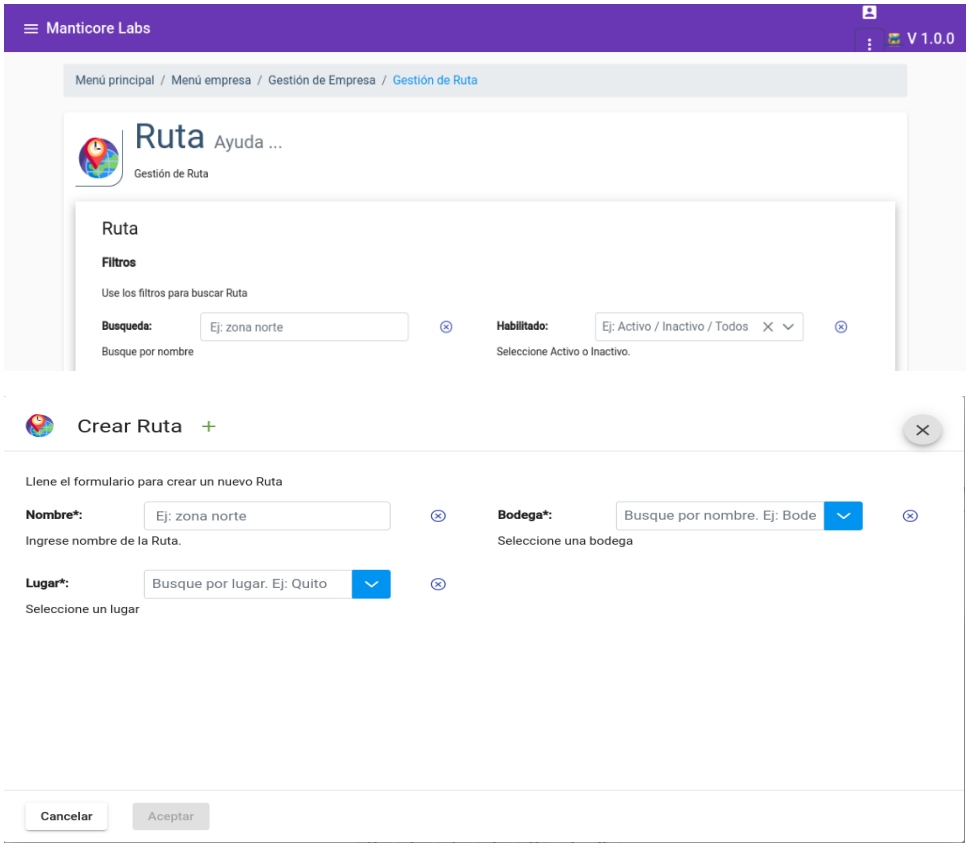


Figura 34. Pantalla ruta – aplicativo web

En la Figura 35 se muestra la gestión de edificio cliente ruta la cual permite asignar edificios de clientes a una ruta.

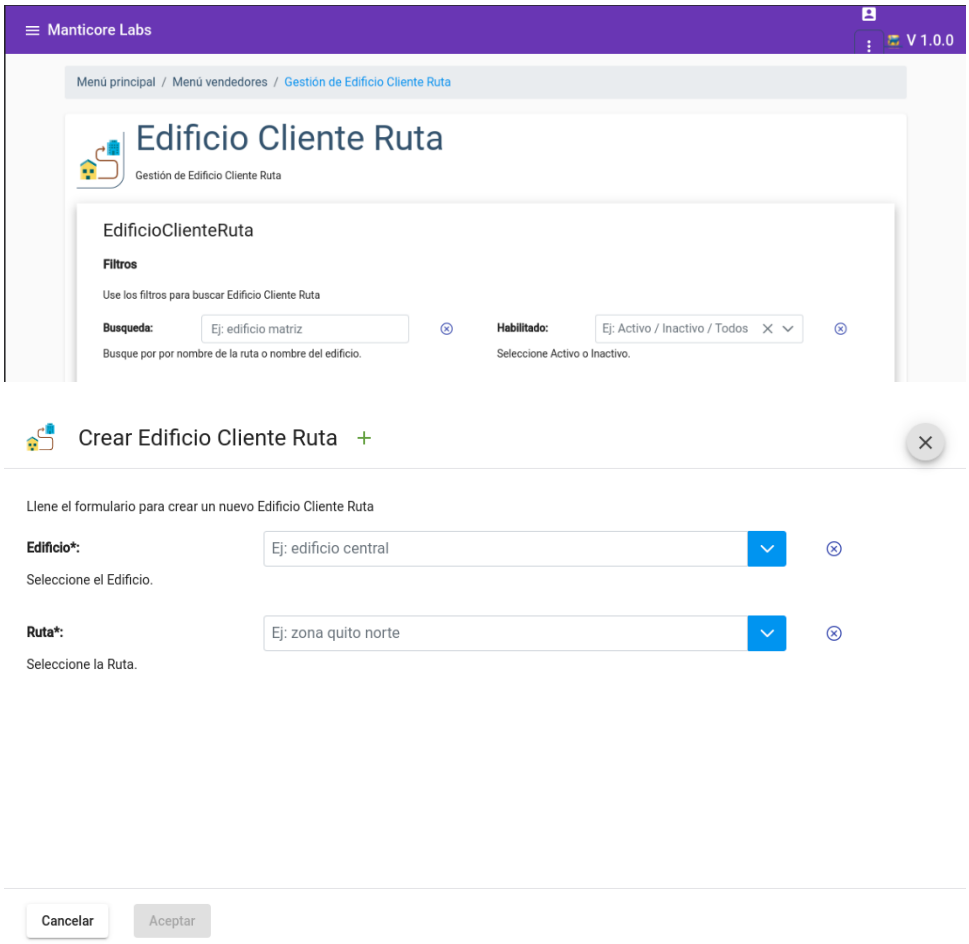


Figura 35. Pantalla edificio cliente ruta – aplicativo web

En la Figura 36 se muestra la pantalla “Cronograma cabecera” la cual permite crear cronogramas y asignarles una ruta.

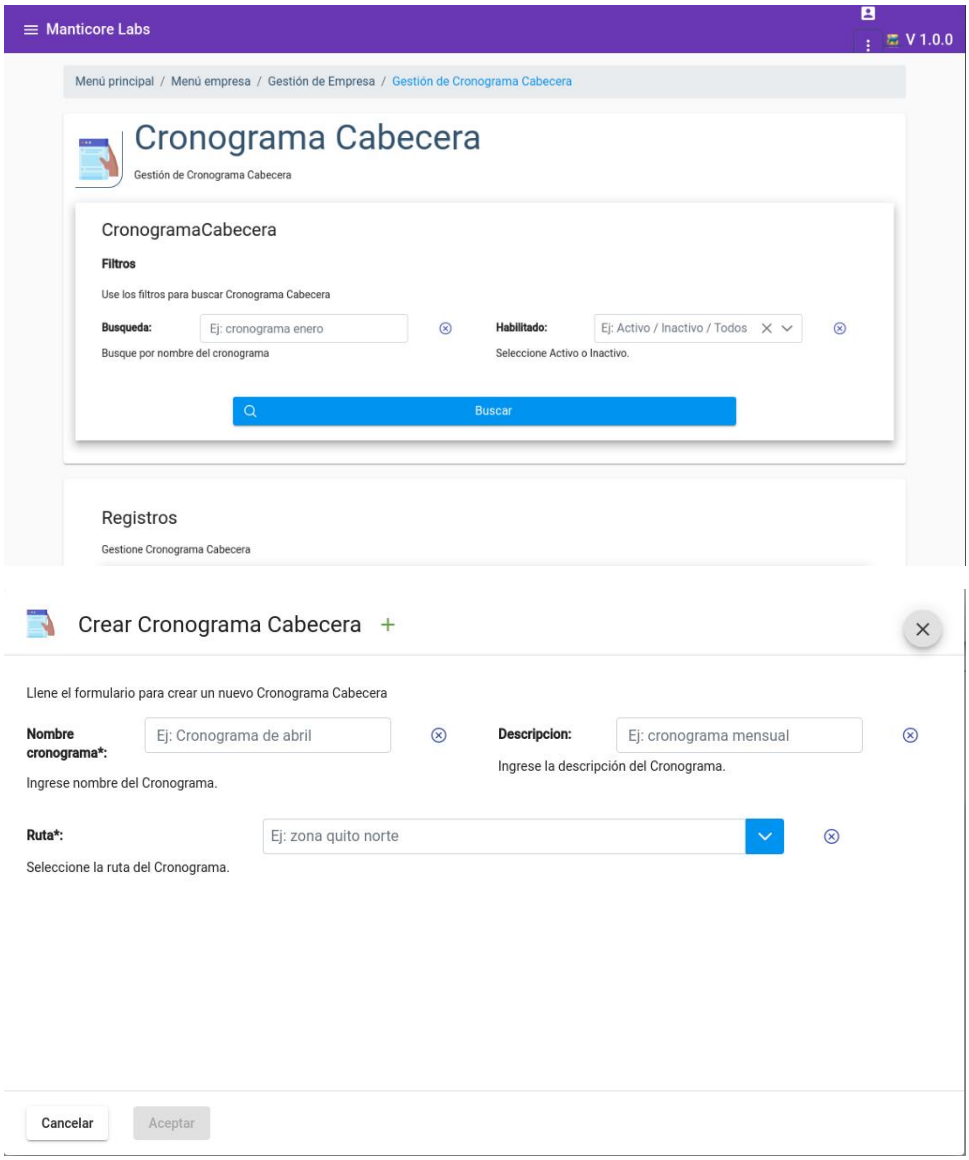


Figura 36. Pantalla cronograma cabecera – aplicativo web

En la Figura 37 se muestra la pantalla “Cronograma detalle” la cual permite asignar los días que se debe realizar un cronograma o una fecha específica.

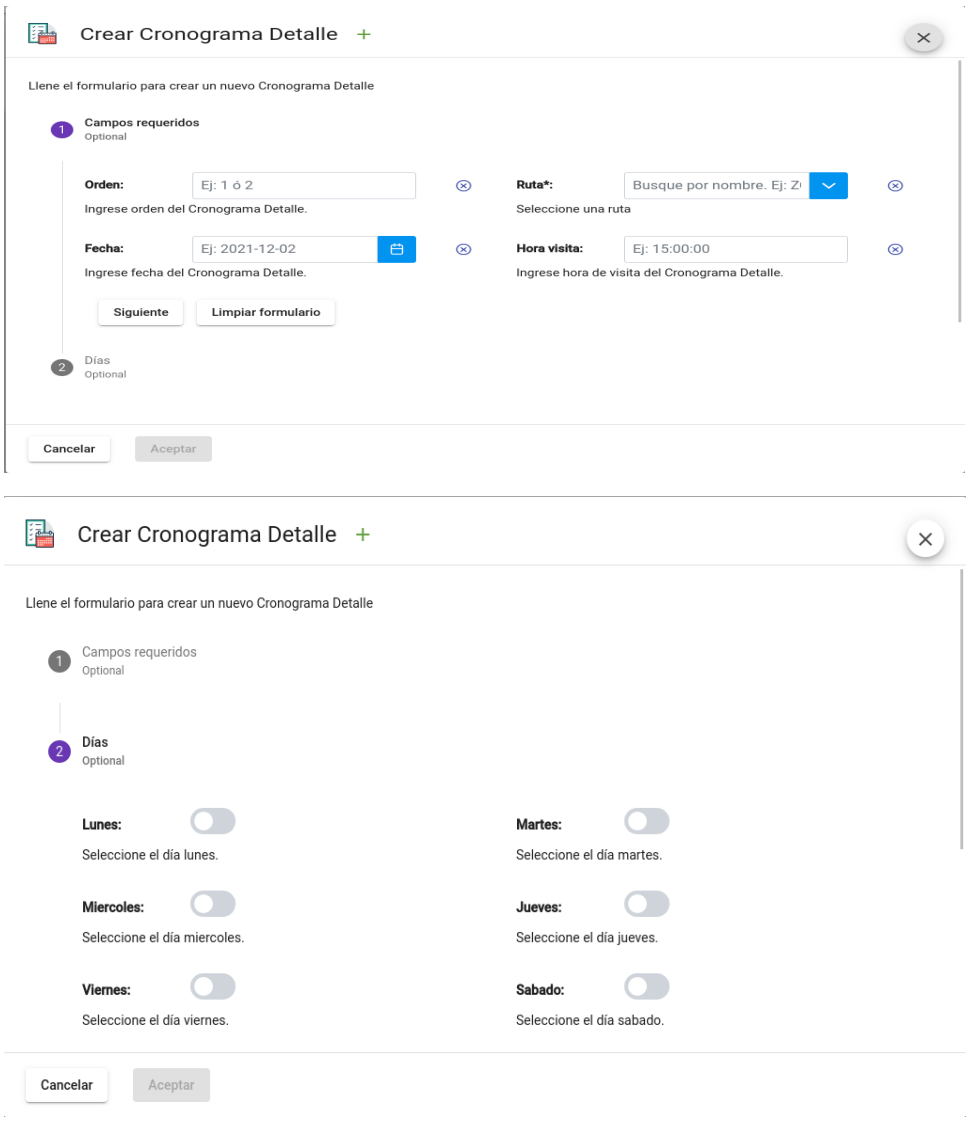


Figura 37. Pantalla cronograma detalle – aplicativo web

En la Figura 38 se muestra la pantalla de “Rutas de vendedor” la cual permite asignar rutas a un vendedor.

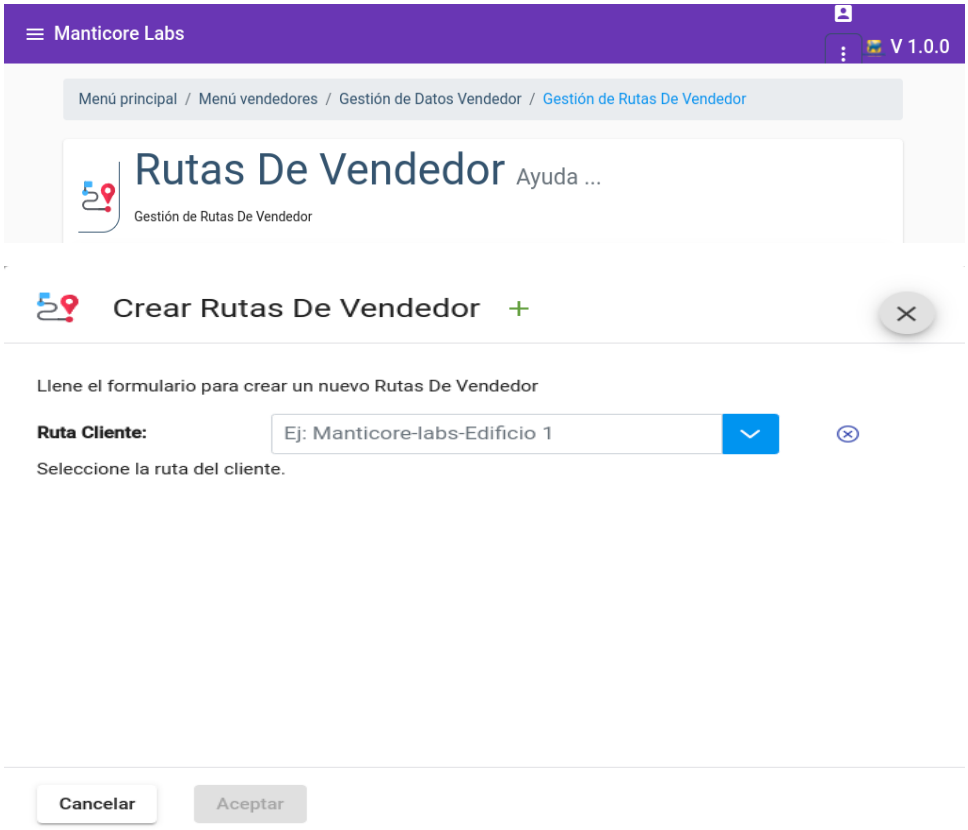


Figura 38. Pantalla rutas de vendedor – aplicativo web

En la Figura 39 se muestra la pantalla de “Visitas” la cual permite revisar las visitas realizadas por todos los vendedores, con sus pedidos, y recaudos. Estas pantallas se manejan en forma de gestiones, en las que se tiene que crear manualmente cada uno de los registros necesarios a diferencia del diseño más sencillo de la aplicación móvil.

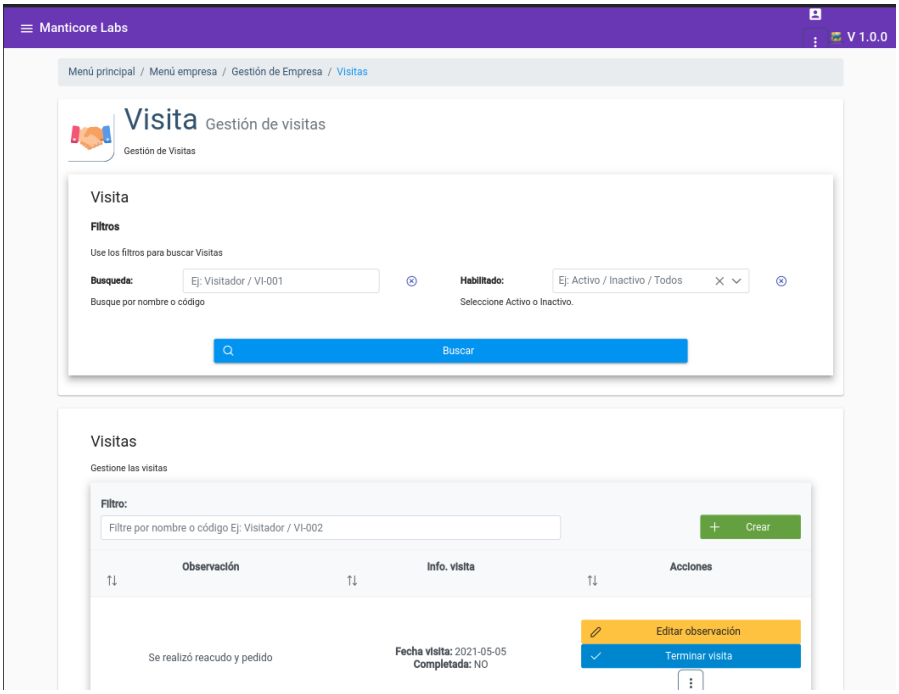


Figura 39. Pantalla visita – aplicativo móvil

En la Figura 40 se muestra la pantalla cabecera documento la cual permite crear cabeceras similares a las que tienen las facturas.

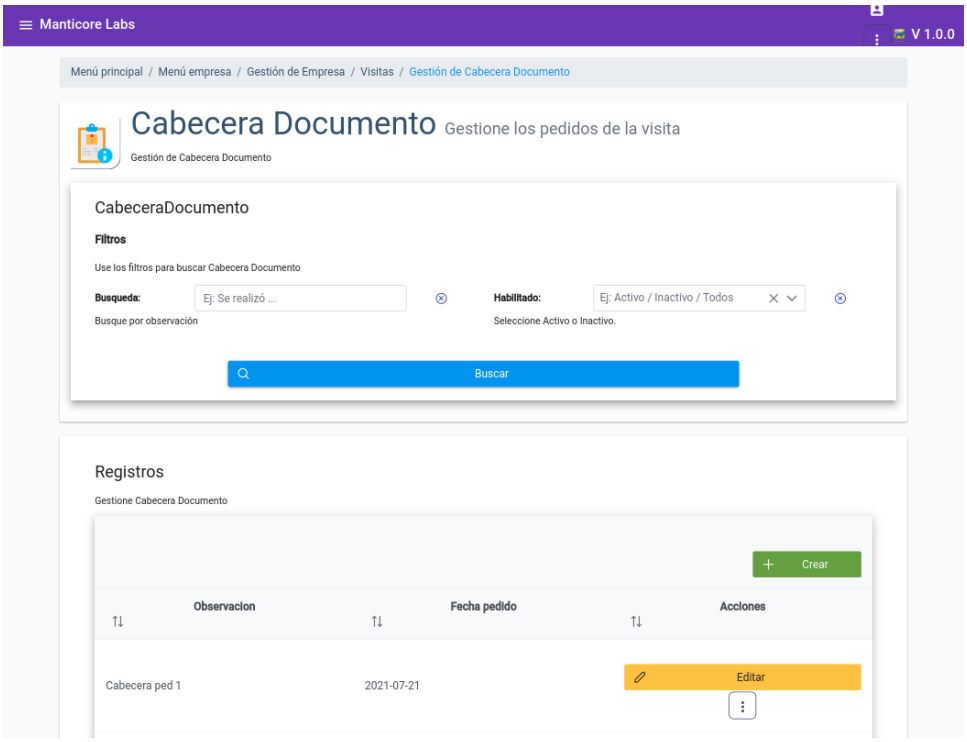


Figura 40. Pantalla cabecera documento – aplicativo web

En la Figura 41 se muestra la pantalla de detalles de pedido la cual permite crear, editar detalles de cada cabecera. Cada detalle viene a representar un artículo que ha hecho un cliente. Cada cabecera puede darse de baja seleccionando el botón Dar de baja, o entregar seleccionando el botón Entregar.

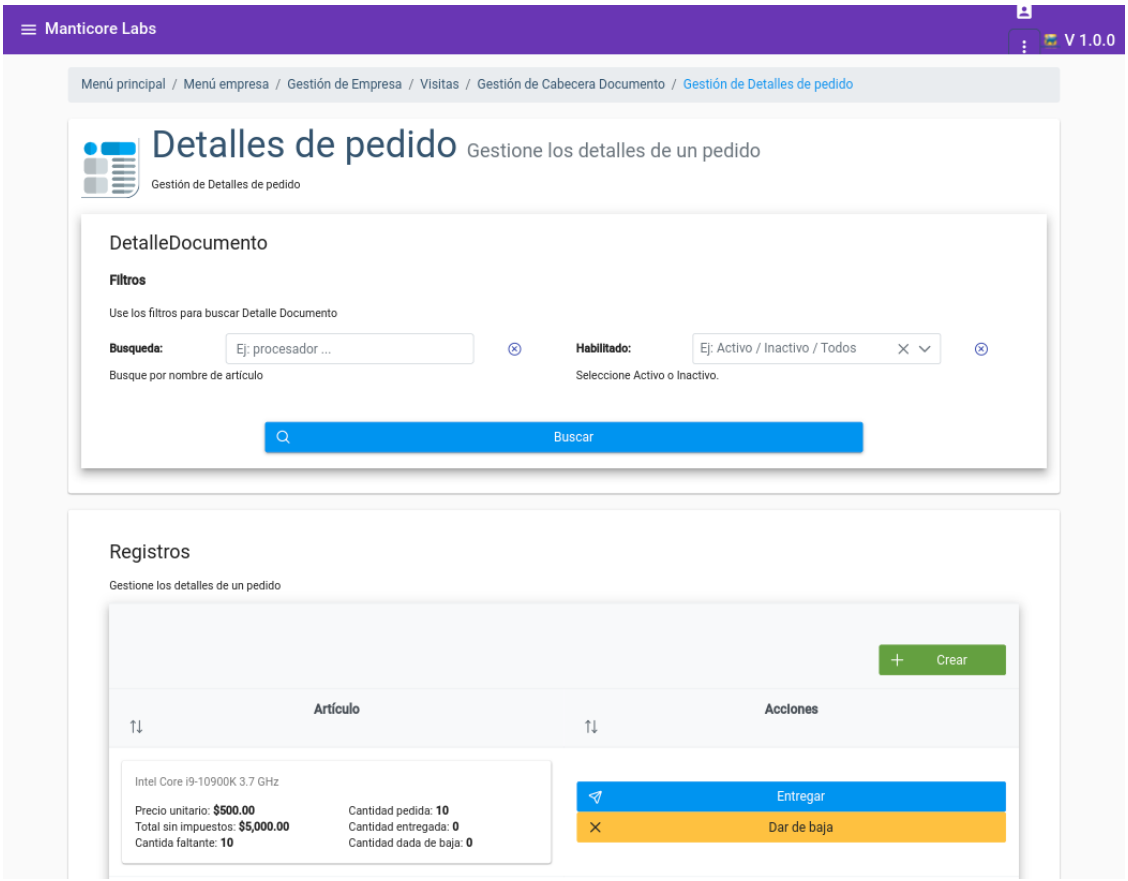


Figura 41. Pantalla detalles de pedido – aplicativo web

En la Figura 42 se muestra la pantalla de recaudos la cual permite crear un recaudo por visita. Al finalizar una visita de recaudos se actualizarán los valores: Total recaudado, observaciones y el estado completado.

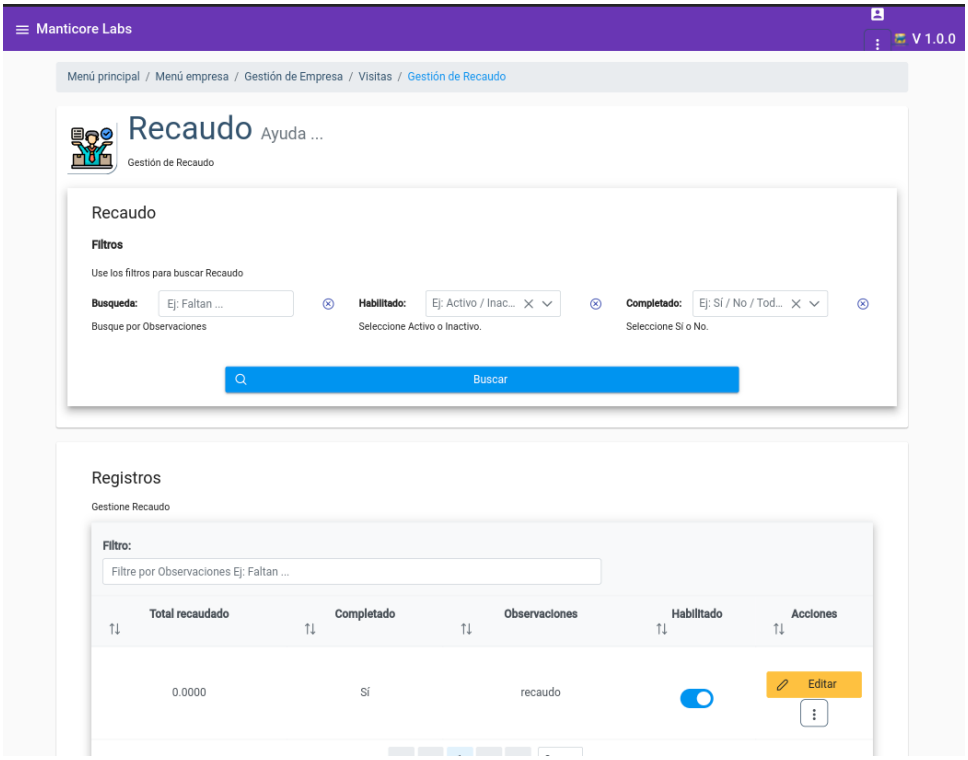


Figura 42. Pantalla recaudo – aplicativo web

En la Figura 43 se muestra la pantalla forma pago detalle la cual permite a un vendedor realizar abonos en los distintos medios de pago.

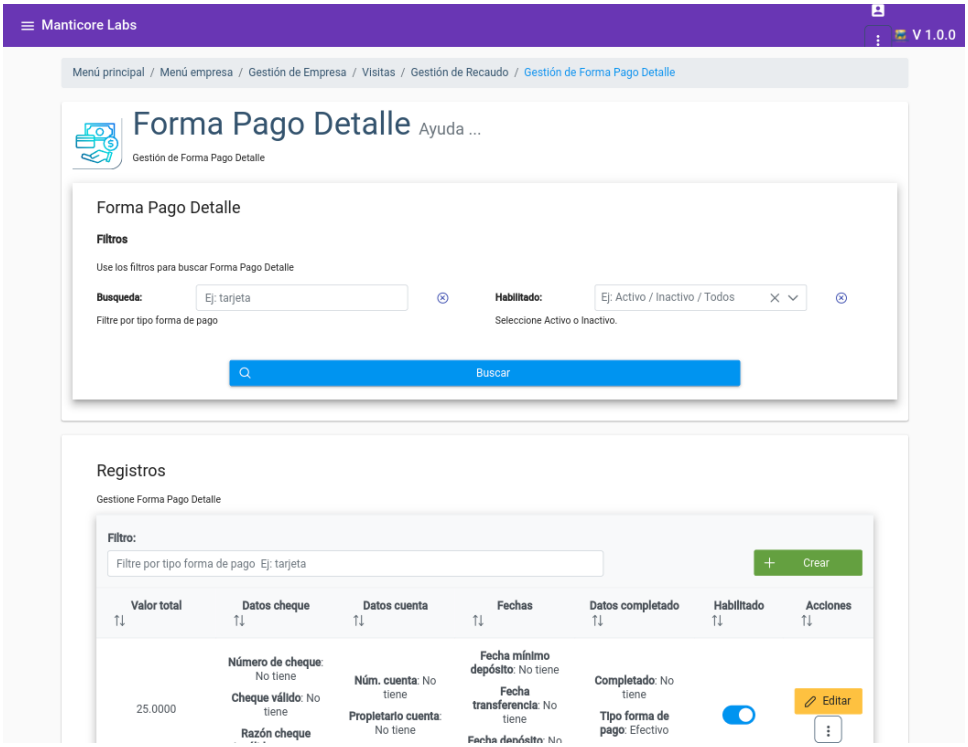


Figura 43. Pantalla forma pago detalle – aplicativo web

En la Figura 44 se muestra la pantalla de recaudo detalle la cual permite asignar los abonos que se detallaron en la Figura 43 a facturas. Lo que importa de esta gestión es que es posible separar un abono para varias facturas.

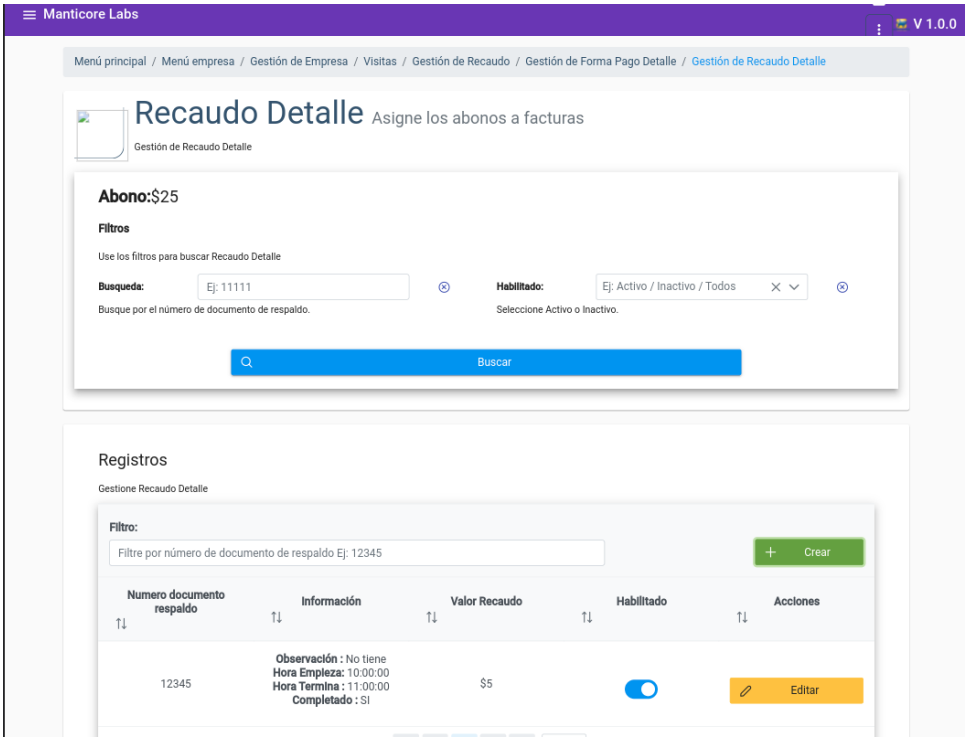


Figura 44. Pantalla recaudo detalle – aplicativo web

3.2. Resultados de la evaluación de la aplicación

Para la evaluación de la aplicación se utilizó SUS, que permite medir la usabilidad de objetos, dispositivos y aplicaciones. Este sistema de evaluación cuenta con un formulario de 10 preguntas, que son puntuadas de 1 a 5, en las que 1 significa total desacuerdo y 5 total acuerdo. El resultado es el promedio de todas las encuestas realizadas.

La evaluación fue realizada a 12 personas, antes de realizar la encuesta, cada persona probó la aplicación, realizando flujos de pedidos, recaudos y mostrar inventario de los clientes. El evaluador realizó las siguientes acciones: escoger un cronograma, añadir un pedido con descuento, un pedido sin descuento, entregar los productos, realizar el recaudo, finalizar la visita, y guardar los datos de la visita. Al finalizar el evaluador realizó la encuesta.

Los resultados de la encuesta por cada persona se pueden ver en la Figura 45.

Figura 45. Resultados de la encuesta por persona

En las siguientes figuras se muestran los resultados por cada pregunta que se realizó.

Las opciones de respuesta en cada pregunta fue la siguiente escala de Linkert:

- Totalmente en desacuerdo.
- En desacuerdo.
- Ni de acuerdo, ni en desacuerdo.
- En acuerdo.
- Totalmente de acuerdo.

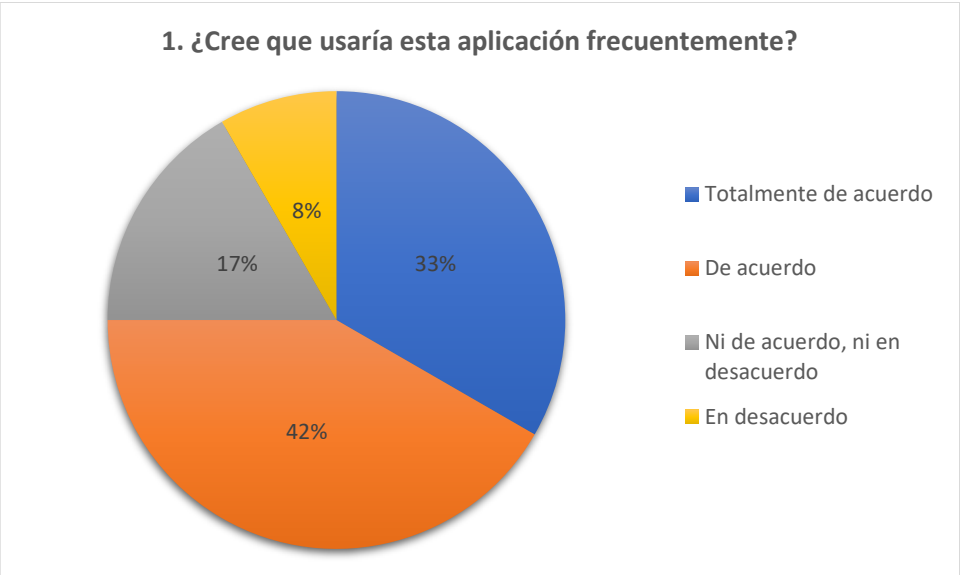


Figura 46. Resultados pregunta 1

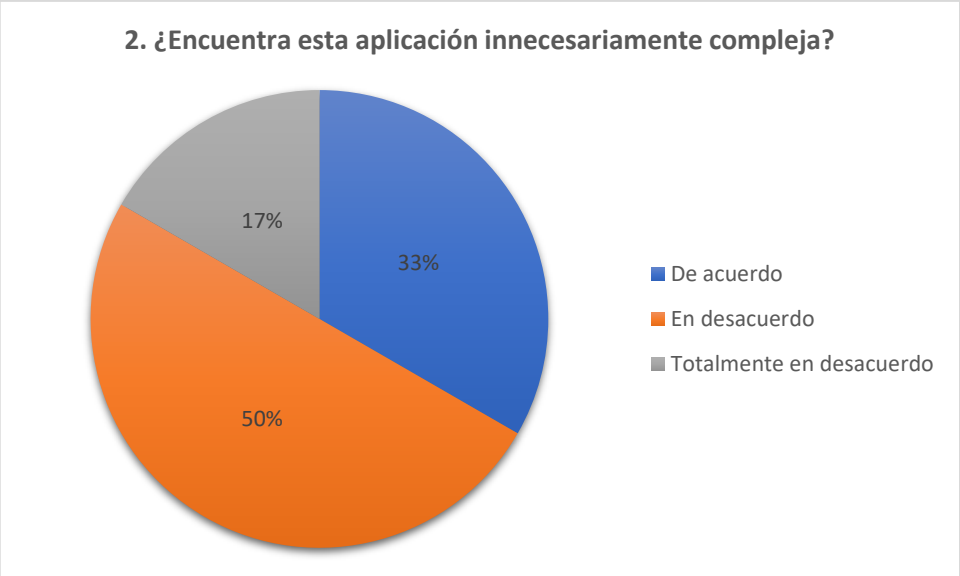


Figura 47. Resultados pregunta 2

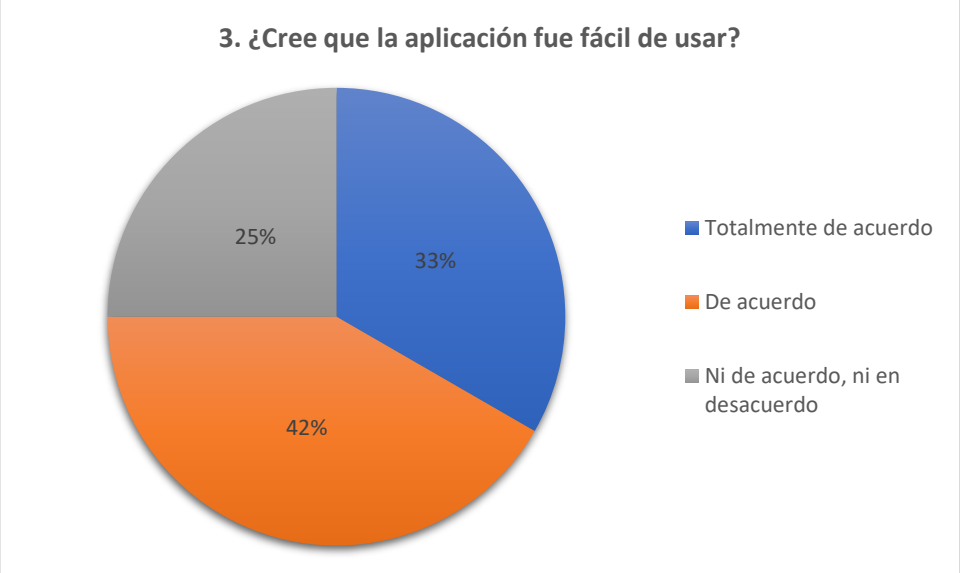


Figura 48. Resultados pregunta 3

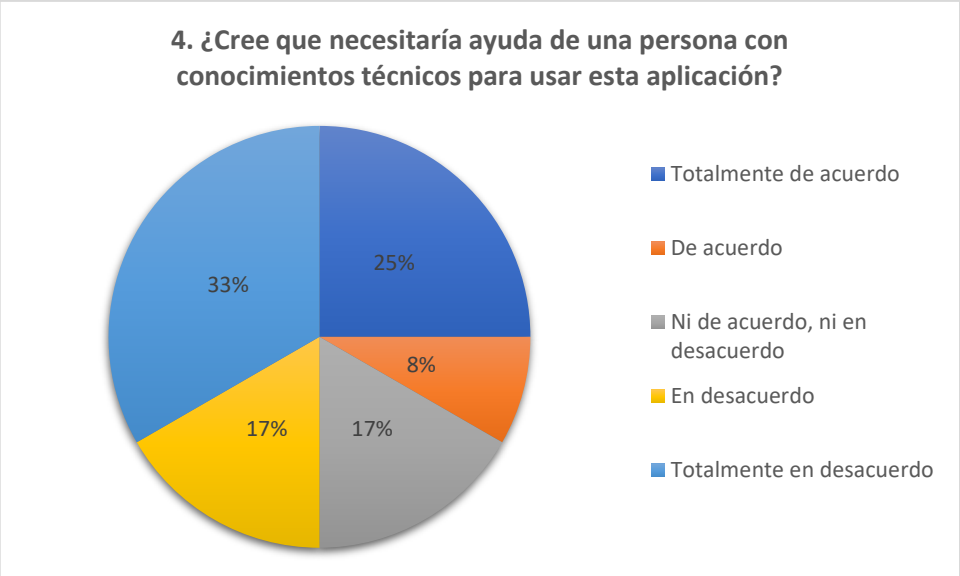


Figura 49. Resultados pregunta 4

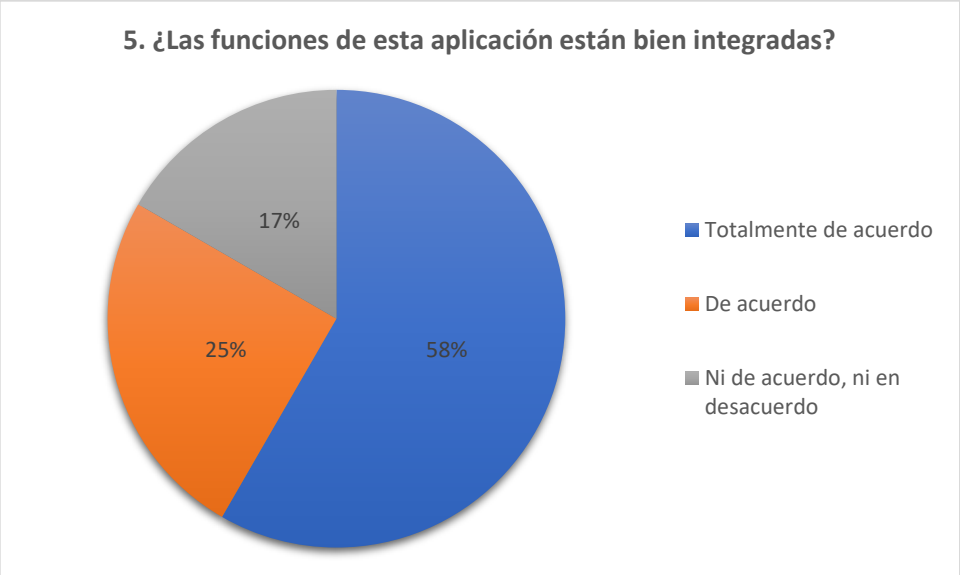


Figura 50. Resultados pregunta 5

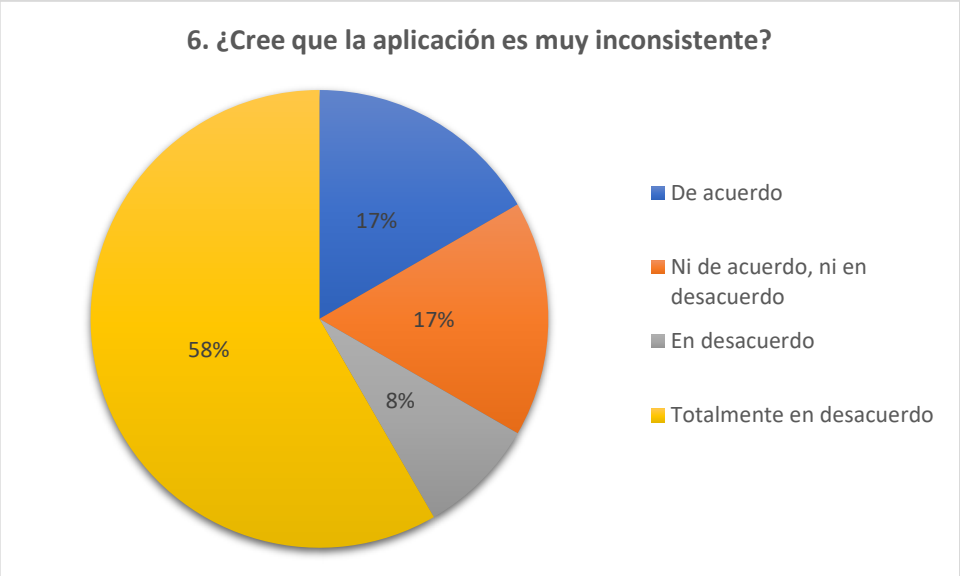


Figura 51. Resultados pregunta 6

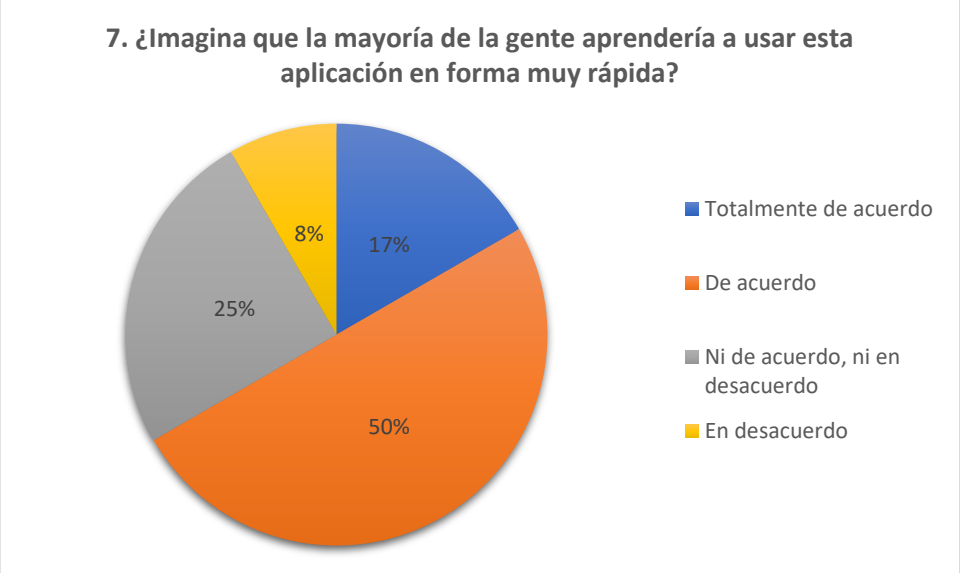


Figura 52. Resultados pregunta 7

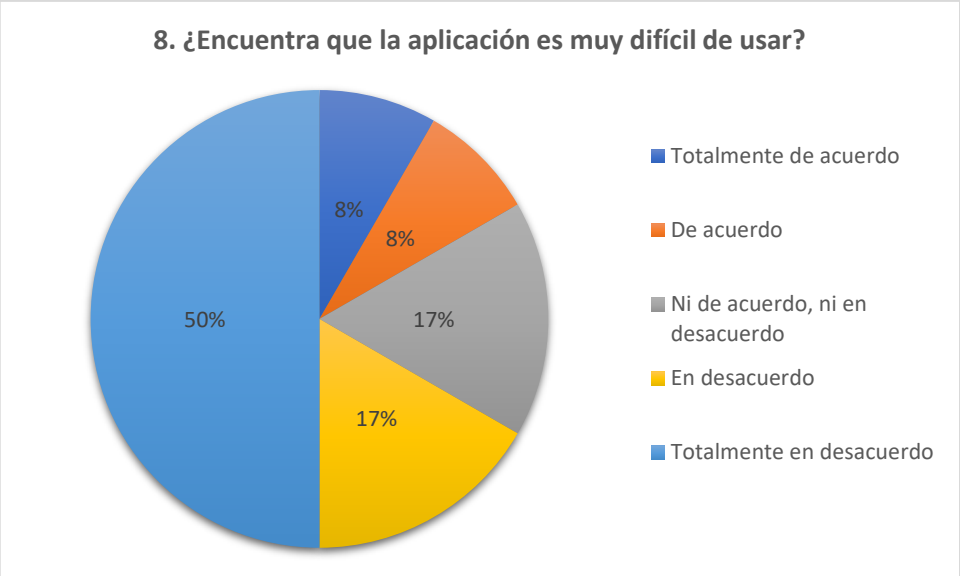


Figura 53. Resultados pregunta 8

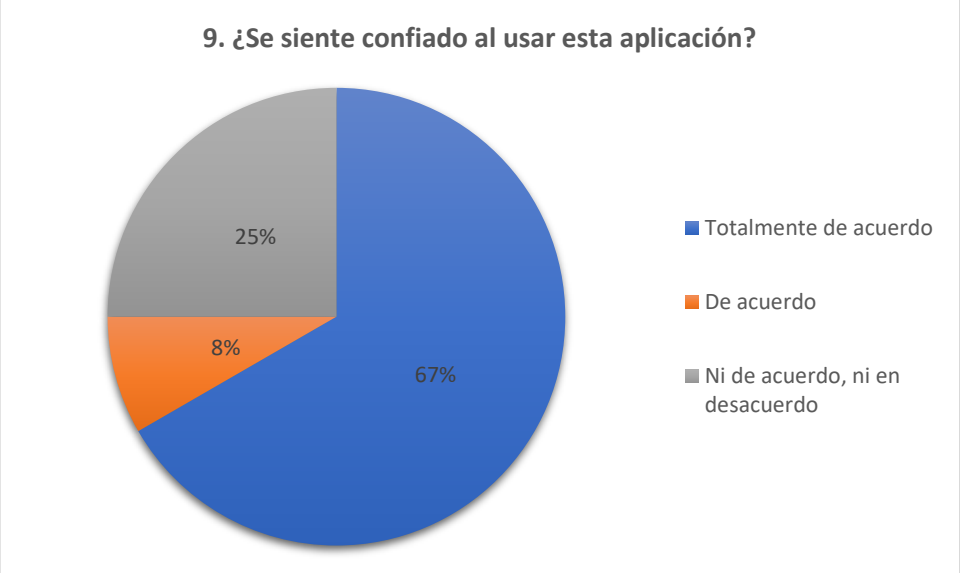


Figura 54. Resultados pregunta 9

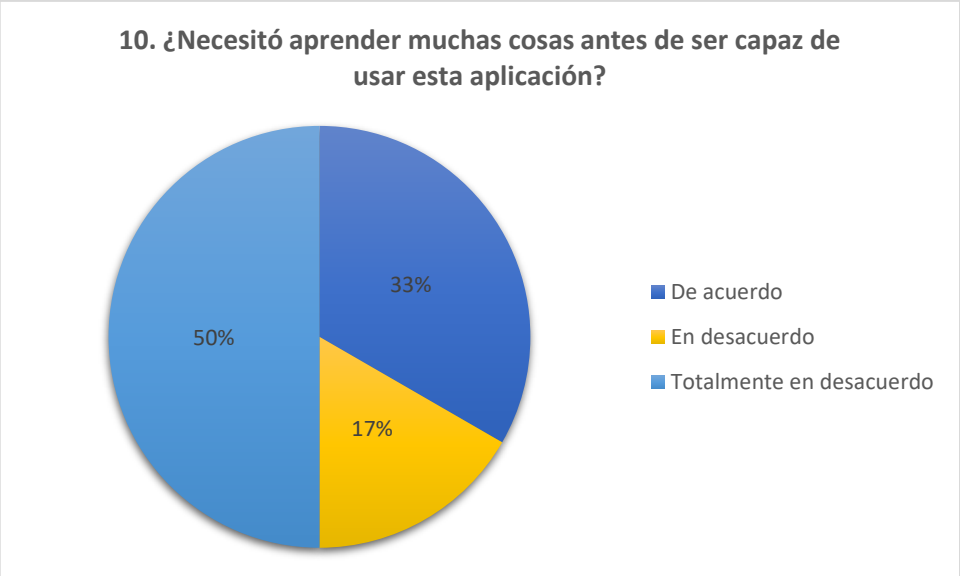


Figura 55. Resultados pregunta 10

3.3. Discusión

En base a los resultados por pregunta de SUS, se obtuvo un puntaje de 73,12 de 100 que según se indica en [32], se puede afirmar que el presente trabajo está por encima del promedio de las aplicaciones, que es un 68. Por tanto, se puede asumir que la aplicación resulta algo compleja de utilizar para personas que no tienen experiencia en el contexto de uso de la aplicación, por lo que piensan que necesitan aprender muchas cosas para poder usarla, por otro lado, existen personas que tienen experiencia en ventas y abonos a las cuales les resultó sencillo usarla, y por lo que su calificación fue alta.

Dado que solamente se obtuvo 5 puntos encima del promedio, pudo deberse al uso de librerías proporcionadas por la empresa desarrolladora que ponen un grado de complejidad mayor a cuando se realiza con componentes modificables. No obstante, las librerías ingresan características de seguridad y código heredado que hicieron que el desarrollo sea eficiente.

4. CONCLUSIONES

4.1. Conclusiones

- El alcance de la reingeniería aumentó en la implementación debido a que hizo falta tener el aplicativo web en la empresa en los submódulos de ventas, recaudos e inventario que permita a el usuario administrador realizar modificaciones de registros de pedidos, ventas o abonos cuando un vendedor comete errores en el ingreso de datos.
- Los requerimientos para el desarrollo de la PWA se enfocaron en el modo de uso en modo offline del aplicativo para los procesos de pedidos y recaudos en una visita a un cliente.
- El rediseño de la arquitectura y base de datos permitió que los submódulos de inventario, y recaudos se simplifiquen de tal modo que la comprensión de la base de datos sea más sencilla, pues se redujo la cantidad de tablas. La arquitectura MVVM permitió que la aplicación tenga otra manera de comunicar los datos al servidor. La Modularización permitió implementar prácticas recomendadas por Angular para mejorar la estructura del proyecto. Tanto en distribución, agrupación y nombrado de archivos y carpetas.
- La implementación de los flujos CI/CD permitió una mayor facilidad para la integración de nuevas características del aplicativo y el despliegue de este, al automatizar estos procesos.
- La implementación del submódulo de inventario y la implementación de una PWA junto con estrategias de manejo de cache eficientes permitieron una mejor experiencia al usuario final, sin la necesidad de una conexión a internet, lo cual es la base de una PWA.
- El puntaje alcanzado mediante la evaluación de usabilidad indica que la aplicación se encuentra por encima de la calificación promedio obtenida por 500 sistemas evaluados en un estudio realizado por Jeff Sauro [32]. Lo que la encaja en un nivel de usabilidad aceptable.

4.2. Recomendaciones

- La aplicación debe estar en un hosting que brinde conexión segura para proteger el service worker.
- Realizar futuras implementaciones en el archivo de service worker para manejar más funcionalidades en modo sin conexión.

- Probar el aplicativo en dispositivos con sistema operativo IOS para obtener retroalimentación de problemas visuales. El aplicativo fue probado únicamente en dispositivos con sistema operativo Android.

REFERENCIAS BIBLIOGRÁFICAS

[1] MDN Web Docs, «Progressive web apps (PWAs),» 2020. [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps. [Último acceso: 26 Junio 2020].

[2] A. Osmani, «Getting Started with Progressive Web Apps,» Google Developers, [En línea]. Available: //developers.google.com/web/updates/2015/12/getting-started-pwa. [Último acceso: 10 Junio 2020].

[3] P. Lepage, «Tu primera Progressive Web App,» Google Developers, 16 Mayo 2019. [En línea]. Available: https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/?hl=es. [Último acceso: 10 Junio 2020].

[4] «How to use the Play Console - Play Console Help,» Support Google, [En línea]. Available: https://support.google.com/googleplay/android-developer/answer/6112435. [Último acceso: 07 Julio 2020].

[5] R. Islam, T. Arafhin y R. Islam, «Mobile Application and Its Global Impact,» Diciembre 2010. [En línea]. Available: http://ijens.org/107506-0909%20IJET-IJENS.pdf. [Último acceso: 2021 Agosto 10].

[6] J. Cuello y J. Vittone, Diseñando apps para móviles, Catalina Duque Giraldo ed., 2013, pp. 20-25.

[7] I. Sommerville, Ingeniería de Software, Novena ed., Mexico: Pearson Educations, 2011.

[8] K. Schwaber, Agile Project Management, Microsoft Press, 2004.

[9] Scrum, «What is Scrum?,» [En línea]. Available: https://www.scrum.org/resources/what-is-scrum. [Último acceso: 4 Julio 2022].

[10] Gitlab, «Unify the entire DevOps lifecycle with GitLab,» Gitlab, [En línea]. Available: https://about.gitlab.com/stages-devops-lifecycle/. [Último acceso: 25 Marzo 2022].

[11] Atlassian, « What is DevOps?,» [En línea]. Available: https://www.atlassian.com/devops. [Último acceso: 28 Abril 2021].

[12] Microsoft Azure, «What is DevOps,» [En línea]. Available: https://azure.microsoft.com/en-us/overview/what-is-devops. [Último acceso: 28 Abril 2021].

[13] Servicio de Informática de la Universidad de Alicante, «Modelo Vista Controlador,» [En línea]. Available: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html. [Último acceso: 08 Junio 2021].

[14] Microsoft, «The MVVM Pattern,» [En línea]. Available: The MVVM Pattern. [Último acceso: 08 Junio 2021].

[15] Digital, «System Usability Scale (SUS): Improving Products Since 1986,» 25 Febrero 2022. [En línea]. Available: https://digital.gov/2014/08/29/system-usability-scale-improving-products-since-1986/. [Último acceso: 10 Abril 2022].

[16] MDN Web Docs, «HTML5,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Glossary/HTML5. [Último acceso: 21 Mayo 2021].

[17] MDN Web Docs, «CSS: Cascading Style Sheets,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS. [Último acceso: 28 Junio 2021].

[18] V. Antani, Mastering JavaScript: Explore and master modern JavaScript techniques in order to build large-scale web applications, PACKT, 2016.

[19] Typescriptlang, «What is TypeScript?,» [En línea]. Available: https://www.typescriptlang.org/.

[20] Angular, «What is Angular?,» [En línea]. Available: https://angular.io/guide/what-is-angular. [Último acceso: 29 Mayo 2021].

[21] Nestjs, «Nestjs,» [En línea]. Available: https://docs.nestjs.com/. [Último acceso: 29 Mayo 2021].

[22] Nodejs, «Acerca de Node.js,» [En línea]. Available: https://nodejs.org/es/about/. [Último acceso: 29 Mayo 2021].

[23] Ionic Framework, «Introduction to Ionic,» [En línea]. Available: https://ionicframework.com/docs/. [Último acceso: 2021 Mayo 29].

[24] MySQL, «MySQL Documentation,» [En línea]. Available: https://dev.mysql.com/doc/. [Último acceso: 29 Mayo 2022].

[25] JetBrains, «WebStorm: The Smartest JavaScript IDE by JetBrains,» [En línea]. Available: https://www.jetbrains.com/webstorm/. [Último acceso: 29 Mayo 2022].

[26] Git, «Git,» [En línea]. Available: https://git-scm.com/. [Último acceso: 29 Mayo 2022].

[27] GitKraken, «Best Git Client for Windows, Mac, Linux,» [En línea]. Available: https://www.gitkraken.com/git-client. [Último acceso: 29 Mayo 2022].

[28] Lucidchart, «Visualiza tus tus procesos, personal y sistemas con diagramas,» [En línea]. Available: Available: https://www.lucidchart.com/pages/es/producto. [Último acceso: 29 Mayo 2022].

[29] Figma, «Figma,» [En línea]. Available: https://www.figma.com/. [Último acceso: 29 Mayo 2022].

[30] Angular, «Angular coding style guide,» [En línea]. Available: https://angular.io/guide/styleguide. [Último acceso: 10 Febrero 2022].

[31] V. Wu, «How to use GitLab for Agile software development,» How Agile artifacts map to GitLab features and how an Agile iteration looks in GitLab, 5 Marzo 2018. [En línea]. Available: https://about.gitlab.com/blog/2018/03/05/gitlab-for-agile-software-development/. [Último acceso: 19 Julio 2020].

[32] J. Sauro, «Measuring U,» [En línea]. Available: https://measuringu.com/sus/. [Último acceso: 28 Abril 2022].

ANEXOS

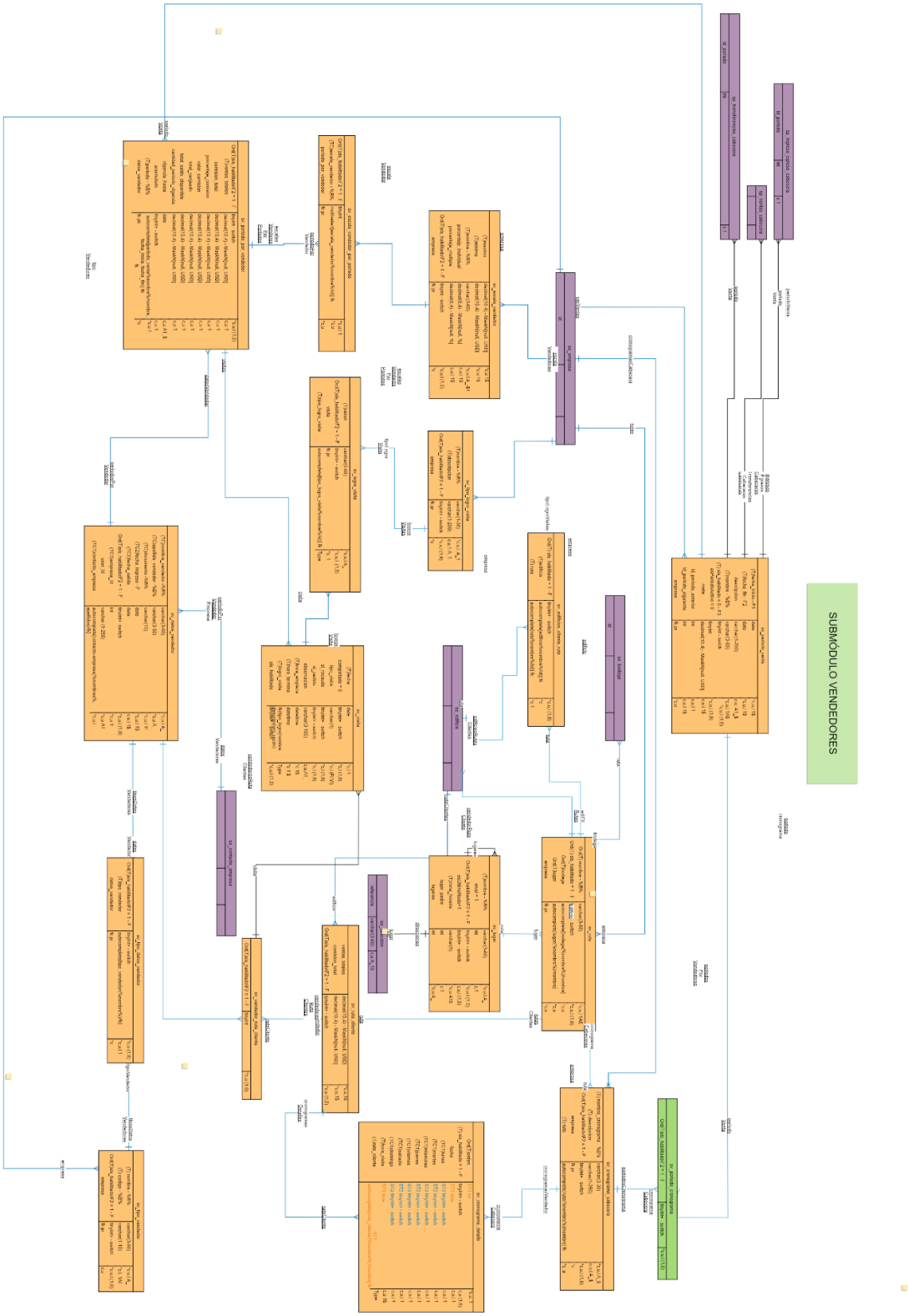
Anexo 1: Diagramas de la base de datos anterior

Anexo 2: Diagramas de la base de datos actualizada

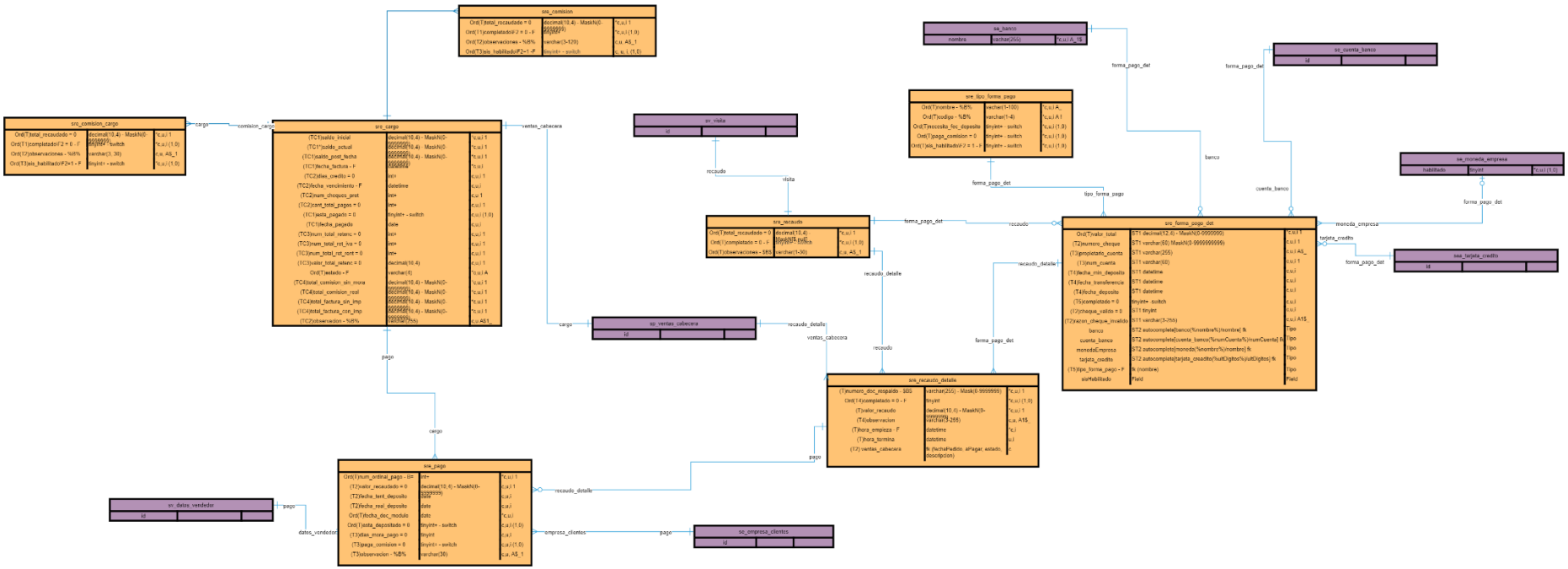
Anexo 3: gitlab-ci.yml file

Anexo 4: Link video aplicación móvil

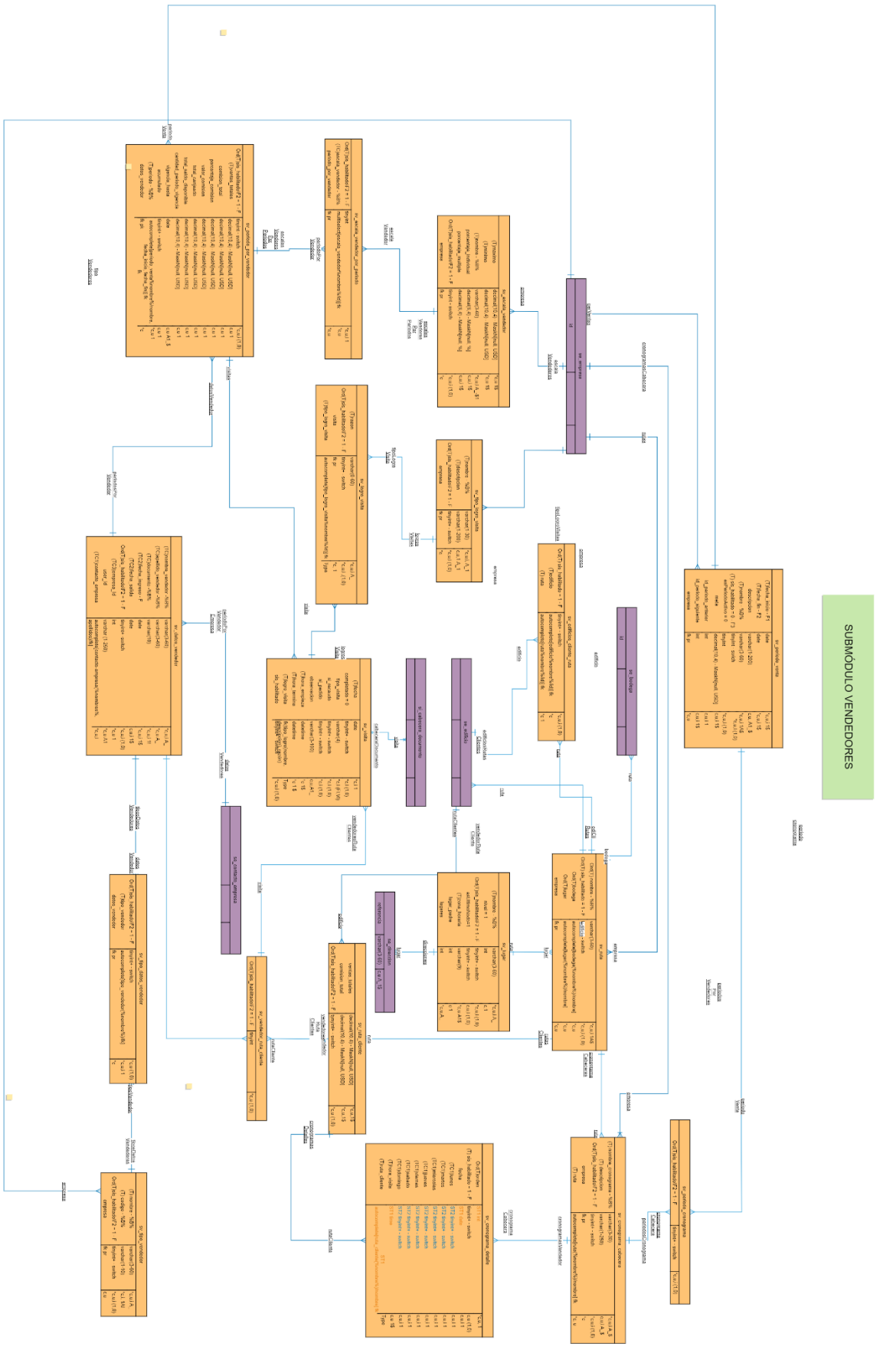
Anexo 1. Diagramas de la base de datos anterior



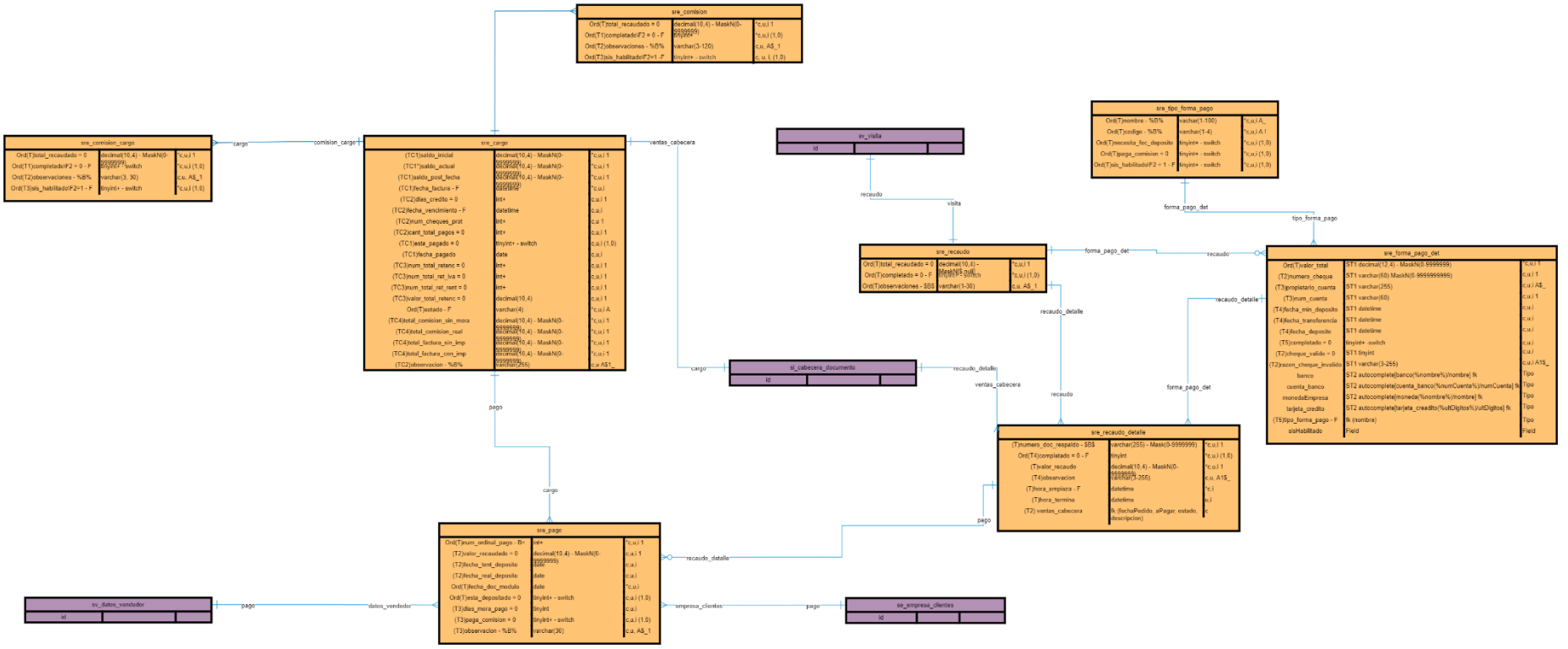
SUBMÓDULO RECAUDOS



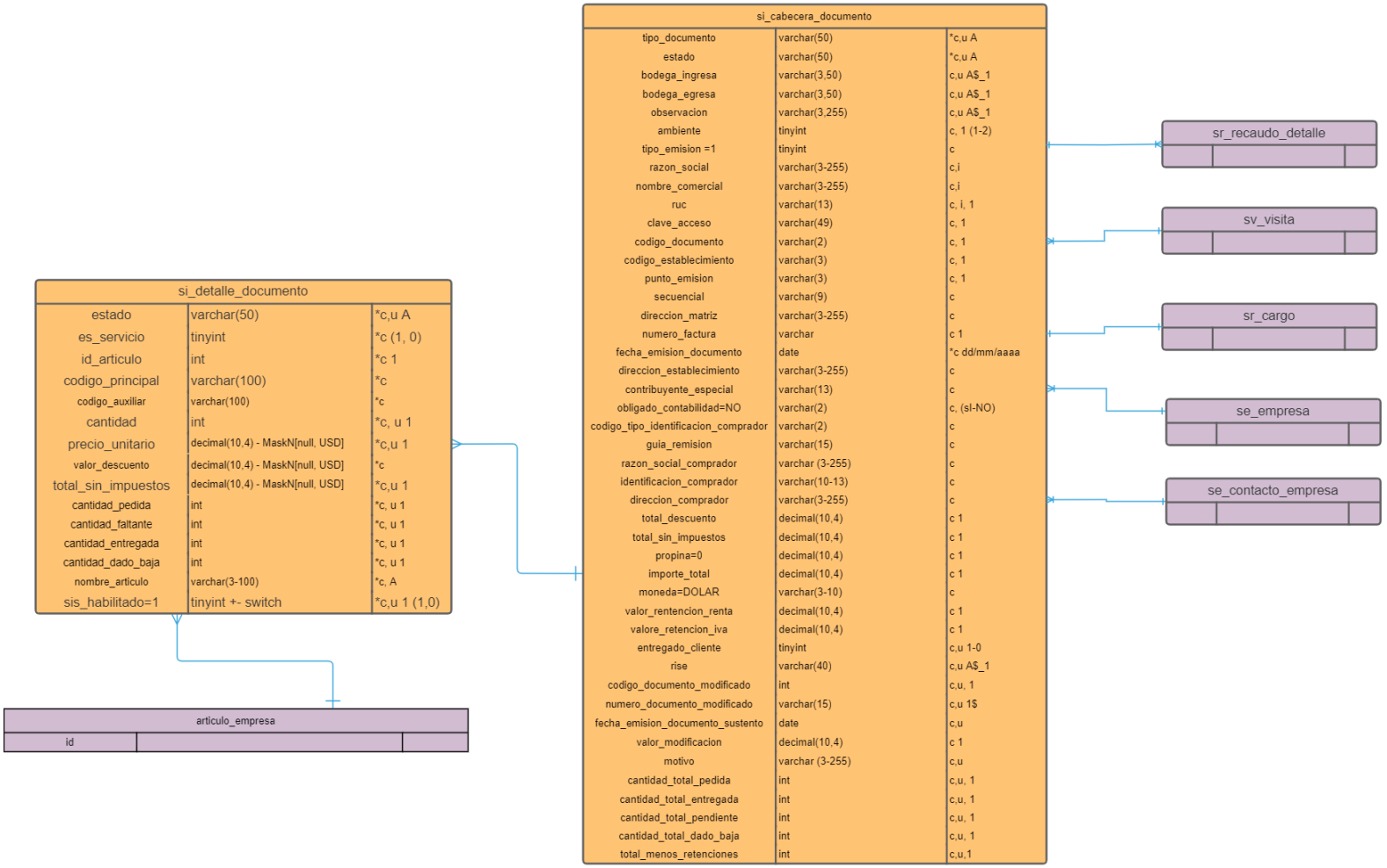
Anexo 2. Diagramas de la base de datos actualizada



SUBMÓDULO RECAUDOS



SUBMÓDULO INVENTARIO



Anexo 3. gitlab-ci.yml file

```
image: node:10.15.1

variables:
  GIT_SUBMODULE_STRATEGY: recursive


before_script:
  - cd front-prueba-submodulos && npm i
  - ls
  - 'which ssh-agent || ( apt-get update -y && apt-get install openssh-client -y )'
  - eval $(ssh-agent -s)
  - echo "$SSH_BACKEND" | tr -d '\r' | ssh-add - > /dev/null
  - mkdir -p ~/.ssh
  - chmod 700 ~/.ssh
  - git config --global user.email "eadepto@hotmail.com"
  - git config --global user.name "Adrian Eguez"
  - ssh-keyscan gitlab.com >> ~/.ssh/known_hosts
  - chmod 644 ~/.ssh/known_hosts


stages:
  - test
  - build
  - deploy
  - prod
```

```
build:
  stage: build
  only:
    - desarrollo
  script:
    - npm run prestart:build

lint:
  stage: test
  only:
    - desarrollo
  script:
    - npm run lint


build_test:
  stage: deploy
  only:
    - fear-test
  script:
    - cd ci-cd
    - node copiar-contenido.js test
    - cd ..
    - pwd
    - npm run prestart:prod -- --base-href=/fear-test/front-app/ && npm run compress
```

- ssh -Tv git@gitlab.com

- echo "Clonando repo"

- rm -rf backend/

- git clone git@gitlab.com:manticore-labs/trabajo/proyectos/submodulos/fear-mpc-v2/fear-mpc-v2-backed.git backend

- pwd

- cd backend

- git checkout fear-test

- rm -rf backend-prueba-submodulos/publico/front-app/

- cd backend-prueba-submodulos/publico/

- ls

- mkdir front-app

- cd front-app

- pwd

- cp -a ../../../../dist/front-prueba-submodulos/. ../front-app/

- pwd

- ls

- rm index.html

- cd .. # /frontEnd/backend/backend/publico

- pwd

- cd .. # /frontEnd/backend/backend

- pwd

- cd .. # /frontEnd/backend

- pwd

- cd .. # /frontEnd/

```
    - pwd
    - cd ci-cd
    - node copiar-contenido-backend.js test
    - cd ..
    - cd backend
    - cd "$PROJECT_NAME_BACKEND"
    - git add .
    - git commit -m "Deploy front test" || true
    - git push origin fear-test || true



build_prod:
  stage: prod
  only:
    - master
  script:
    - cd ci-cd
    - node copiar-contenido.js prod
    - cd ..
    - pwd
    - npm run prestart:prod
    - ssh -Tv git@gitlab.com
    - echo "Clonando repo"
    - rm -rf backend/
    - git clone git@gitlab.com:manticore-labs/trabajo/proyectos/submodulos/fear-mpc-v2/fear-mpc-v2-backend.git backend
```

```
- pwd
- cd backend
- git checkout master
- git status
- rm -rf "$PROJECT_NAME_BACKEND"/publico/front-app/
- git status
- git add .
- git commit -m "Eliminando archivos" || true
- cd "$PROJECT_NAME_BACKEND"/publico/
- mkdir front-app
- cd front-app
- pwd
- cp -a ../../../../dist/front-prueba-submodulos/. ../front-app/
- pwd
- rm index.html
- cd .. # /frontEnd/backend/backend/publico
- pwd
- cd .. # /frontEnd/backend/backend
- pwd
- cd .. # /frontEnd/backend
- pwd
- cd .. # /frontEnd/
- pwd
- cd ci-cd
- node copiar-contenido-backend.js prod
```

```
- cd ..
- cd backend
- cd "$PROJECT_NAME_BACKEND"
- git add .
- git commit -m "Deploy front produccion" || true
- git push origin master || true
```

Anexo 4. Link video aplicación móvil

https://www.youtube.com/watch?v=CVb0Ce2fzLY