Universitat Oberta de Catalunya

uoc.edu

Aplicación multiplataforma para la gestión y seguimiento de entregas a domicilio para empresa de logística

José Luis Rubio Leira
Grado de Ingeniería Informática
Desarrollo multiplataforma de aplicaciones móviles

Consultores: Carlos Sánchez Rosa y Jordi Almirall López
Profesor: Carles Garrigues Olivella

Junio de 2023

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Aplicación multiplataforma para la gestión y seguimiento de entregas a domicilio para empresa de logística*
Nombre del autor:	*José Luis Rubio Leira*
Nombre del consultor/a:	*Carlos Sanchez Rosa* *Jordi Almirall López*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2023
Titulación::	*Grado en Ingeniería Informática*
Área del Trabajo Final:	*Desarrollo multiplataforma de aplicaciones móviles*
Idioma del trabajo:	*Español*
Palabras clave	*Automatización, Gestión logística, movilidad*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Este proyecto propone el desarrollo de un sistema automatizado para mejorar la gestión de pedidos en una empresa de logística que realiza entregas a domicilio para el supermercado Eroski en la ciudad de Zaragoza. El nuevo sistema plantea reemplazar los métodos manuales de gestión, procesado y control de los pedidos, así como los procesos de comunicación con los repartidores, por un sistema automatizado.

La solución propuesta consiste en una aplicación desarrollada en Node.js para la descarga y procesamiento automático de pedidos desde los servidores de Eroski, una base de datos MongoDB alojada en la nube para el almacenamiento de los datos, una aplicación web responsiva HTML5, desarrollada en PHP con el framework Laravel, para la explotación de estos datos y una aplicación nativa en Android para que los repartidores gestionen las entregas.

Para el desarrollo del sistema, se adoptó un enfoque ágil basado en la metodología Kanban, donde las funcionalidades se desarrollaron de forma secuencial, generando entregables incrementales. Además, se aplicó un diseño centrado en el usuario (DCU) para garantizar que el producto final fuera altamente usable y eficiente, satisfaciendo las necesidades de los usuarios.

El resultado es un sistema que libera al personal de la gestión del procesamiento de pedidos y permite a los repartidores conocer en tiempo real sus entregas pendientes y gestionarlas eficientemente durante su jornada de trabajo.

Abstract (in English, 250 words or less):

This project proposes the development of an automated system to improve the management of orders in a logistics company that carries out home deliveries for the Eroski supermarket in

the city of Zaragoza. The new system suggests replacing manual methods of management, processing, and control of orders, as well as communication processes with the delivery staff, with an automated system.

The proposed solution consists of an application developed in Node.js for the automatic download and processing of orders from Eroski's servers, a MongoDB database hosted in the cloud for data storage, a responsive HTML5 web application developed in PHP with the Laravel framework, to exploit this data, and a native Android application for the delivery staff to manage deliveries.

For the system development, an agile approach based on the Kanban methodology was adopted, where functionalities were developed sequentially, generating incremental deliverables. Furthermore, a user-centered design (UCD) was applied to ensure that the final product is highly usable and efficient, meeting users' needs.

The result is a system that frees staff from managing order processing and allows the delivery staff to know their pending deliveries in real-time and manage them efficiently during their workday.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

El presente Trabajo está enfocado a resolver la problemática en la operativa diaria de la empresa Logística Blaniza, S.L., que realiza el Reparto a domicilio de los pedidos del supermercado Eroski en la ciudad de Zaragoza. La empresa está formada por una flota de furgonetas y por un equipo de repartidores que son quienes realizan las recogidas en tienda y el reparto a domicilio.

Eroski pone a disposición de las empresas que realizan el reparto a domicilio en las diferentes ciudades, una carpeta en un servidor FTP a la que hay que conectarse utilizando una VPN corporativa mediante un cliente GlobalProtect.

Con una periodicidad de 10 minutos, Eroski deja en dicha carpeta la información de los pedidos realizados en su tienda online utilizando ficheros de texto (uno por pedido) que contienen los datos de la entrega a domicilio, esto es, datos del destinatario, tienda asociada al pedido, número de cajas y bolsas de seco, de fresco, de congelado, el importe del pedido etc.

La forma que la empresa tiene de acceder a los datos de los pedidos es mediante la descarga manual periódica de los ficheros a través de una conexión mediante un cliente de FTP (WinSCP), en un ordenador con un cliente GlobalProtect instalado y conectado a la VPN de Eroski. Estos ficheros se procesan a mano, y el control de los pedidos se realiza manualmente mediante una hoja Excel. Además, la comunicación con los repartidores para indicarles las recogidas y las entregas se hace a través de mensajes de WhatsApp en un grupo compartido al que acceden todos ellos.

La forma de trabajar actual, tiene varios problemas derivados del procesado manual de la información, lo que puede inducir a que se produzcan errores. El hecho de que se descarguen y se procesen manualmente los pedidos, obliga a que haya una persona dedicada exclusivamente al procesado periódico de los pedidos y a la comunicación con los repartidores mediante el grupo de WhatsApp anteriormente comentado. Dicha comunicación con los repartidores, al ser grupal, hace que todos los repartidores reciban los mensajes de todas las recogidas y entregas de todas las tiendas, y no se comunique específicamente a cada repartidor cuál es su trabajo pendiente.

1.2 Objetivos del Trabajo

Los objetivos que se pretenden conseguir con el trabajo son los siguientes:

Objetivos Funcionales (ordenados de mayor a menor prioridad)

1.- Descarga y tratamiento de los pedidos de manera automática evitando los errores del procesado manual de los pedidos.

2.- La aplicación móvil debe permitir al repartidor modificar el estado de un pedido (Pendiente, entregado, etc.).

3.- Disponer de un sistema de Gestión de los pedidos: saber en tiempo real qué pedidos se han entregado, cuales están pendientes, etc.

4.- Disponer de un sistema que permita a cada repartidor saber en cada momento y en tiempo real qué pedidos tiene que recoger y entregar.

5.- Disponer de un sistema que permita obtener consultas y estadísticas de los pedidos realizados durante un periodo.

6.- El sistema debe permitir generar cuadros de facturación que permitan a la empresa facturar a Eroski mensualmente por los pedidos procesados.

6.- Creación de un sistema de alertas para avisar de los nuevos pedidos que van entrando.

7.- Mediante el sistema se debe poder controlar el desempeño de cada repartidor (Cuantos pedidos ha realizado en un periodo, etc).

Objetivos No Funcionales

- Deslocalización de la gestión para que pueda realizarse desde cualquier lugar.
- La aplicación que carga los pedidos en la base de datos ha de estar operativa en todo momento, y generar avisos en el caso de no estarlo.
- La aplicación móvil ha de ofrecer tiempos de respuesta rápidos ya que se requiere que las consultas y las modificaciones de estado de los pedidos se hagan de una manera ágil.
- La aplicación móvil que utilicen los repartidores ha de ser fácil de usar y con una interfaz de usuario intuitiva, para que se pueda usar con agilidad.

1.3 Enfoque y método seguido

En un primer momento, para evitar el trabajo relacionado con la descarga y procesado de los ficheros de los pedidos, se intentó que Eroski proporcionara los datos de dichos pedidos a través de servicios web de tipo REST.

Ante la negativa de Eroski de cambiar su modo de trabajo, el sistema tenía que ser capaz de descargar y procesar los pedidos y para ello era necesario que estuviera conectado a la VPN corporativa de Eroski. Esto obligaba a que la aplicación que descarga y procesa los pedidos estuviera en un servidor Windows con el cliente GlobalProtect instalado.

Como en la empresa no se dispone de personal técnico de sistemas ni infraestructura, se decidió que la aplicación de descarga estuviera alojada en un servidor Cloud (Clouding.io) que proporcionara la infraestructura para alojar la aplicación y que permitiera el escalado de los recursos en un futuro si fuera necesario.

Para la descarga y procesado de los pedidos se utilizará una aplicación desarrollada en Node.js que chequeará periódicamente la carpeta FTP remota donde se dejan los ficheros de pedidos, descargará dichos ficheros y los procesará insertando los datos correspondientes en una base de datos que posteriormente será explotada por la aplicación web final.

Para la base de datos se ha elegido utilizar una base de datos NoSQL como MongoDB ya que no se prevén que existan transacciones complejas y el escalado es más sencillo que en una

base de datos relacional. Esta base de datos estará alojada en un servidor cloud MongoDB Atlas, lo que permitirá tanto definir reglas de seguridad específicas como el escalado de los recursos en el caso de que las necesidades de la aplicación crezcan en un futuro.

Los datos de los pedidos insertados en el MongoDB serán explotados por una aplicación responsiva HTML5 desarrollada en PHP con el framework Laravel que servirá tanto para que los gestores de la empresa lleven el control de los pedidos, como para que los repartidores puedan estar al tanto de los pedidos que tienen pendientes en cada una de las tiendas.

Los gestores utilizarán la aplicación a través del navegador web. Para los repartidores, se integrará la aplicación web en una aplicación nativa en android. Un esquema descriptivo de todo el sistema en su conjunto sería el siguiente:

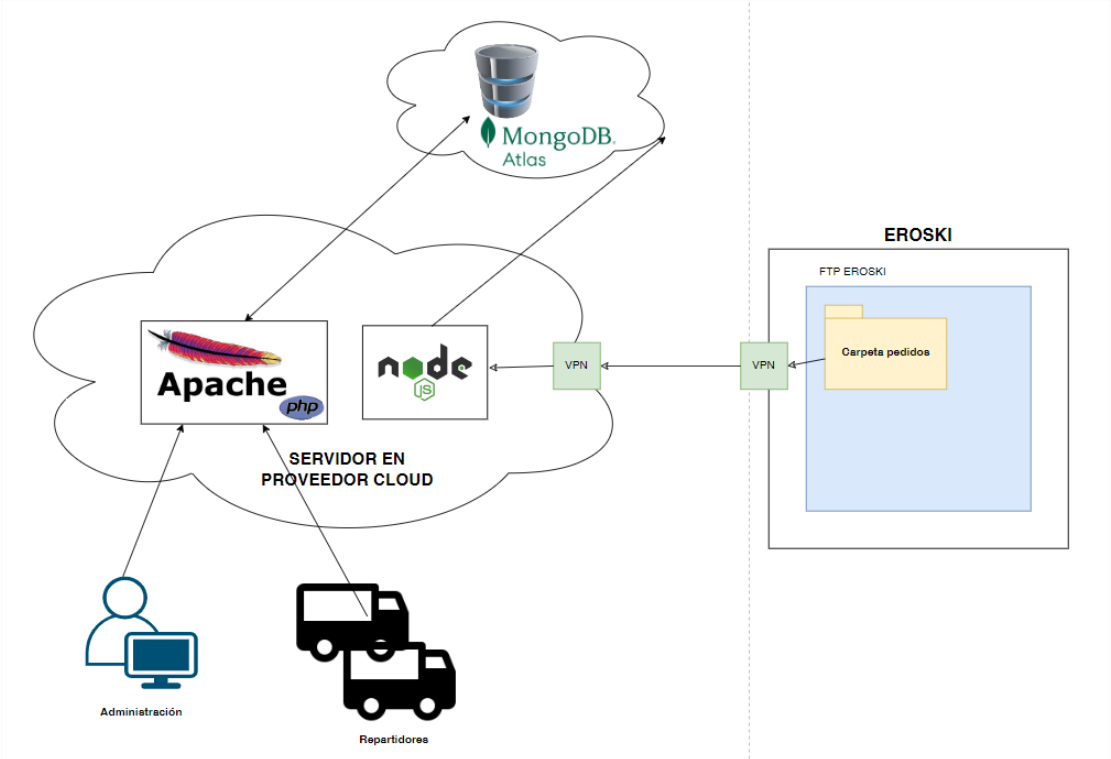


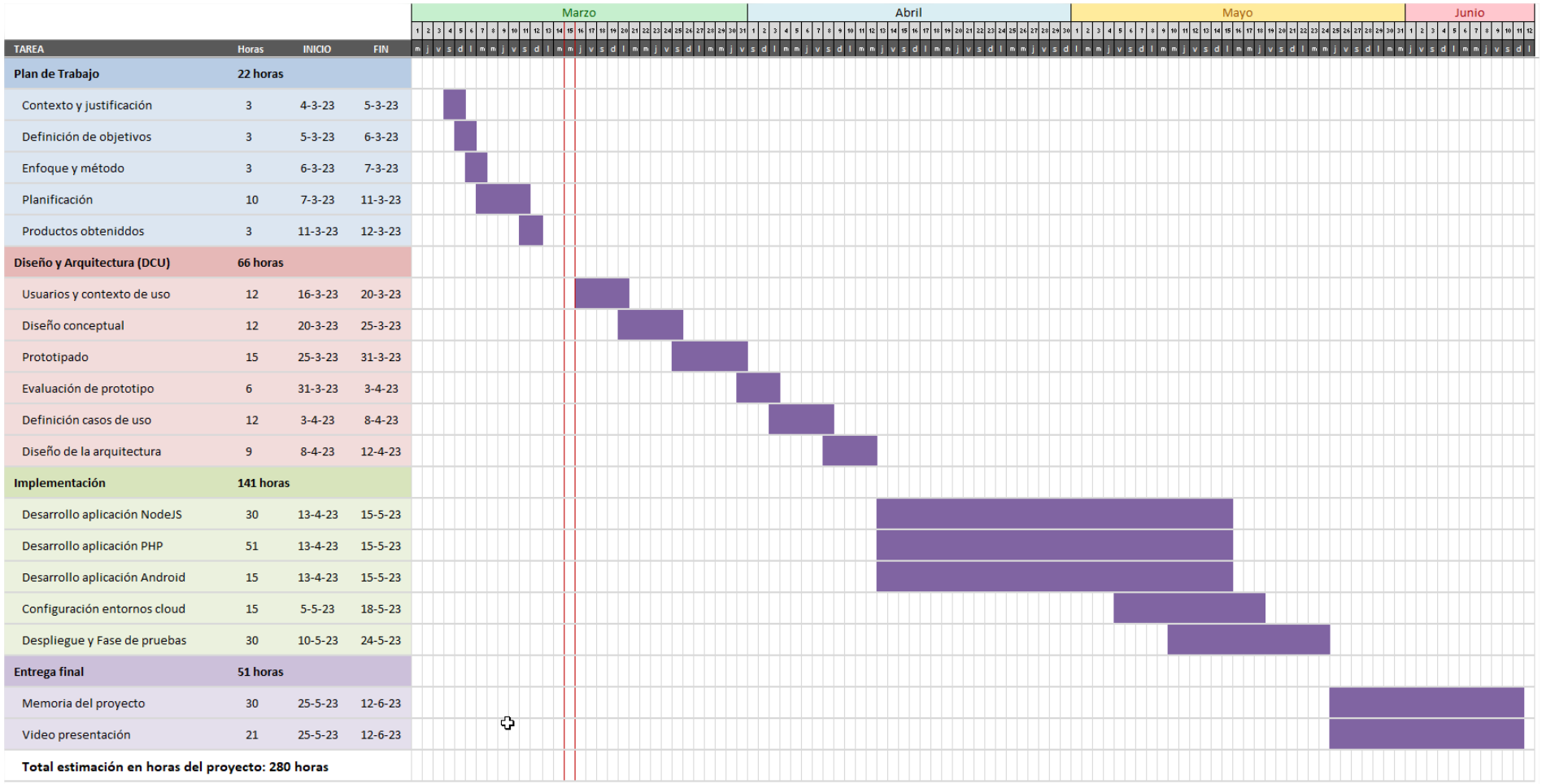
Figura 1 – Diagrama general del sistema

1.4 Planificación del Trabajo y estimación de costes

Para realizar la planificación se ha establecido que cada jornada de trabajo equivale a 3 horas de dedicación, y se estima que la dedicación semanal al trabajo será de 20 horas (15 horas correspondientes al trabajo de lunes a viernes y 5 horas correspondientes al sábado y al domingo). En la fase de implementación, debido al incremento en la carga de trabajo se prevé que si es necesario se aumente el tiempo de dedicación durante los fines de semana hasta alcanzar las 30 horas semanales de dedicación total.

Según el detalle de la planificación que puede verse en el diagrama de Gantt de la *Ilustración2,* la estimación del total del proyecto en horas de trabajo asciende a un total de 280 horas.

Teniendo en cuenta que el proyecto abarca diferentes disciplinas (desarrollo full stack, desarrollo móvil, despliegue en cloud, etc.) se ha establecido un coste estimado de 30 euros por hora de trabajo. Por lo tanto, el coste total estimado del proyecto sería de 8400 euros antes de impuestos.



TAREA	Horas	INICIO	FIN
Plan de Trabajo	**22 horas**		
Contexto y justificación	3	4-3-23	5-3-23
Definición de objetivos	3	5-3-23	6-3-23
Enfoque y método	3	6-3-23	7-3-23
Planificación	10	7-3-23	11-3-23
Productos obteniddos	3	11-3-23	12-3-23
Diseño y Arquitectura (DCU)	**66 horas**		
Usuarios y contexto de uso	12	16-3-23	20-3-23
Diseño conceptual	12	20-3-23	25-3-23
Prototipado	15	25-3-23	31-3-23
Evaluación de prototipo	6	31-3-23	3-4-23
Definición casos de uso	12	3-4-23	8-4-23
Diseño de la arquitectura	9	8-4-23	12-4-23
Implementación	**141 horas**		
Desarrollo aplicación NodeJS	30	13-4-23	15-5-23
Desarrollo aplicación PHP	51	13-4-23	15-5-23
Desarrollo aplicación Android	15	13-4-23	15-5-23
Configuración entornos cloud	15	5-5-23	18-5-23
Despliegue y Fase de pruebas	30	10-5-23	24-5-23
Entrega final	**51 horas**		
Memoria del proyecto	30	25-5-23	12-6-23
Video presentación	21	25-5-23	12-6-23
Total estimación en horas del proyecto: 280 horas			

Figura 2 - Diagrama de Gantt de la planificación del trabajo

1.5 Breve sumario de productos obtenidos

Los productos obtenidos a nivel de desarrollo una vez finalizado el trabajo son:

- Aplicación Node.js de descarga, proceso y almacenamiento de pedidos
- Base de datos MongoDB que contendrá los datos de los pedidos
- Aplicación web realizada en PHP
- Aplicación nativa Android

Los productos obtenidos a nivel de documentación son:

- Código fuente de las distintas aplicaciones
- Memoria final
- Presentación en video

1.6 Breve descripción de los otros capítulos de la memoria

Los capítulos de los que va a constar la siguiente memoria son los siguientes:

Capítulo 2: Diseño centrado en el usuario: Se describirán las fases del DCU (usuarios y contexto de uso, diseño conceptual, prototipado y evaluación del prototipo).

Capítulo 3: Diseño técnico: Contendrá la definición de los casos de uso y el diseño de la arquitectura del sistema.

Capítulo 4: Implementación: Descripción de las herramientas usadas y justificación de las decisiones de implementación.

Capítulo 6: Pruebas: Descripción de las pruebas realizadas para determinar el correcto funcionamiento del sistema.

Capítulo 7: Conclusiones: Descripción de las conclusiones del trabajo, así como reflexión crítica sobre el cumplimiento de los objetivos inicialmente planteados y sobre el cumplimiento de la planificación del trabajo.

Capítulo 8: Glosario: Definición de términos utilizados durante la redacción de esta memoria.

Capítulo 9: Bibliografía: Referencias bibliográficas utilizadas.

Capítulo 10: Anexos: Manual de instalación y uso de la aplicación móvil

2. Diseño centrado en el Usuario

2.1 Usuarios y contexto de uso

En esta fase inicial, el objetivo es obtener una idea lo más fiel posible de las necesidades y expectativas de los usuarios con el sistema final que finalmente se desarrolle. Para llevar a cabo esta fase, se realizará una fase de análisis en la que se utilizarán datos obtenidos a través de diferentes métodos de indagación como observación, y entrevistas. Todo esto con el fin de obtener los distintos perfiles de usuarios del sistema y sus contextos de uso.

Se ha seleccionado el método de shadowing ya que se ha considerado que era fundamental conocer de primera mano el flujo de trabajo diario de los repartidores de la empresa para identificar problemas y puntos de mejora, ya que el proceso de reparto de pedidos constituye el núcleo central del negocio. El método de entrevistas se ha elegido porque permite comprender el contexto, opiniones y necesidades de los usuarios desde su punto de vista, además de permitir un diálogo que permite profundizar en detalles y resolver dudas.

2.1.1 Métodos de indagación

- Observación e investigación contextual (Shadowing)

El objetivo del proceso de shadowing se llevó a cabo con el objetivo de analizar los sistemas de trabajo y comprender las necesidades de los repartidores de la empresa Logística Blaniza durante un día de trabajo típico. Se ha seguido a uno de los repartidores durante una parte de su jornada laboral, observando de cerca sus actividades, interacciones y herramientas utilizadas en el proceso de recogida y entrega de pedidos.

Se comenzó revisando las comunicaciones recibidas a través del grupo de WhatsApp para obtener información sobre los pedidos que debían recoger y entregar. Luego, se dirigieron a las tiendas Eroski asignadas para recoger los pedidos. Se nos comentó que cada repartidor está asignado a una o dos tiendas, pero que en situaciones de mucha carga de trabajo pueden recoger pedidos de otras tiendas.

Una vez en la tienda, el repartidor interactuó con el personal de Eroski para localizar y recoger los pedidos. Organizaron las cajas y bolsas en sus furgonetas según el tipo de producto (seco, fresco, congelado) y la secuencia de entregas. Posteriormente, se realizaron las entregas a los clientes siguiendo la ruta planificada, comunicándose con los clientes en caso de retrasos o problemas en la entrega.

Durante el proceso, los repartidores utilizaron sus dispositivos móviles para acceder a la información a través del grupo de WhatsApp. También utilizaron aplicaciones GPS para la navegación y realizaron llamadas para contactar a los clientes cuando fue necesario.

Se observaron varias dificultades y problemas durante el día, como la necesidad de consultar varias fuentes de información (WhatsApp, hoja de pedidos, etc.), la falta de información específica sobre las tareas asignadas a cada repartidor, ya que cada repartidor debía discriminar qué mensajes eran para él y cuales no, y dificultades para coordinar las entregas de manera eficiente debido a cambios en los horarios o problemas imprevistos.

El proceso reveló oportunidades significativas de mejora en la gestión y coordinación de las actividades de recogida y entrega de pedidos. Estas oportunidades incluyen la centralización y

simplificación de la información accesible para los repartidores, la mejora de la comunicación entre repartidores y gestores, y la optimización de la planificación y coordinación de las entregas a través de un sistema que aborde las necesidades específicas de los repartidores en su trabajo diario.

- Entrevistas en profundidad

Se realizó una entrevista a la persona que se ocupa de las tareas administrativas de Logística Blaniza. La entrevista se realizó en persona y duró aproximadamente 30 minutos. Las preguntas que se hicieron fueron las siguientes:

P: ¿En qué consiste su trabajo dentro de la empresa?
La entrevistada indicó que sus responsabilidades principales incluyen la supervisión de los repartidores, la coordinación de las recogidas y entregas de pedidos, y la comunicación con el supermercado Eroski y los clientes. También indicó que se encarga de la facturación a Eroski de las entregas realizadas.

P: ¿Qué herramientas utiliza para llevar a cabo sus responsabilidades?
La entrevistada mencionó que utilizan una combinación de herramientas y sistemas, como la hoja de cálculo de Excel para la gestión de pedidos, el cliente de FTP (WinSCP) para descargar los archivos de pedido, el cliente GlobalProtect para conectarse a la VPN de Eroski y WhatsApp para la comunicación con los repartidores.

P: ¿Qué dificultades encuentra principalmente en su trabajo diario?
La entrevistada indicó que lo que más dificultades le supone es la carga de trabajo manual y repetitiva involucrada en el procesamiento de descarga de los ficheros de pedido y elaboración de las hojas Excel, así como la dificultad para coordinar a los repartidores.

P: ¿Qué mejoras le gustaría que el nuevo sistema tuviera?
La entrevistada sugirió varias mejoras, como la automatización del proceso de descarga y procesamiento de ficheros de pedidos, la centralización de la información en un sistema al que los repartidores pudieran acceder con un teléfono móvil y al que también pudiera acceder ella desde su escritorio para sus labores de control, gestión y facturación.

También indicó que el sistema debía proporcionar a los repartidores, en la medida de lo posible, acceso rápido y claro a la información de sus pedidos asignados, y que les permitiera de una forma ágil el modificar el estado de los pedidos cuando estos fueran entregados ya que no disponen de mucho tiempo entre entrega y entrega, y, además, desde las oficinas se quiere supervisar las entregas según se vayan produciendo. También incidió en la necesidad de que el nuevo sistema generara informes de pedidos y que ofreciera la posibilidad de exportar todos los reportes en formato Excel ya que dispone de conocimientos avanzados en la herramienta.

2.1.2 Perfiles de Usuario

A través de la observación y las entrevistas se han podido identificar patrones de comportamiento y necesidades de información del personal de la organización, así como sus dificultades en el trabajo diario y las expectativas que tienen del nuevo sistema a Implantar. Esto nos ha permitido la elaboración de los dos perfiles existentes que reflejan los comportamientos y las necesidades de los usuarios reales.

Por ejemplo, mediante la observación, se ha puesto de manifiesto que los repartidores necesitan un método eficiente de consultar los pedidos pendientes y actualizar los estados de los pedidos. Actualmente, la utilización de los grupos de WhatsApp se presenta como un método poco operativo. Asimismo, a través de la entrevista con un gestor de la empresa, se ha visto que es necesario automatizar la descarga y procesamiento de los pedidos de los servidores de Eroski, para hacer que los gestores de la empresa se centren en la propia gestión de los pedidos y la elaboración de los informes pertinentes Los dos perfiles identificados son el perfil de repartidor y el perfil de gestión. Se detallan a continuación:

Perfil Repartidor

Características	
Edad	25 – 60
Sexo	Masculino
Nacionalidad	España, Ecuador, Colombia
Experiencia laboral	Desde 1 a 20 años
Uso tecnología	Todos utilizan móvil a nivel de usuario básico

Contextos de uso

Utilizarán la aplicación para determinar qué entregas tienen pendientes para recoger en cada tienda. Una vez recogido el género y cargado en la furgoneta, usaran la aplicación para determinar la dirección de entrega. Tras conducir hasta la dirección de destino, y una vez entregada la mercancía, utilizaran la aplicación para cambiar el estado del pedido a "Entregado". Si ocurre algún problema, como una ausencia del cliente en el domicilio, utilizarán la aplicación para llamar al cliente.

Análisis de tareas a realizar con la aplicación

- Identificarse en la aplicación mediante el formulario de login
- Seleccionar la tienda de la que se quiere obtener la información
- Revisar datos de pedidos pendientes
- Modificar el estado de un pedido (a "Pendiente" o a "Entregado")
- Posibilidad de contactar telefónicamente con un cliente en caso de que ocurra un problema
- Posibilidad de consultar estados de pedidos de otras fechas
- Cerrar sesión

Características a incorporar tras fase de indagación

- Dado que un repartidor tiene asignada una o varias tiendas, pero puede tener que ir a cualquier otra que no tenga asignada, en la lista de tiendas aparecerán en primer lugar sus tiendas asignadas por motivos de agilidad, pero también podrá consultar los pedidos del resto.
- Debido al bajo perfil tecnológico de los usuarios y a la necesidad de operar de forma rápida, la aplicación ha de ser sencilla y ágil en su manejo.

Perfil Gestión	
Características	
Edad	40-45
Sexo	Masculino y Femenino
Nacionalidad	España
Experiencia laboral	Más de 20 años
Uso tecnología	Experiencia con herramientas y aplicaciones ofimática (Excel, Word, etc.)
Contextos de uso	
Utilizará la aplicación para supervisar y coordinar los repartos, lo que incluye el seguimiento del progreso de las entregas diarias y el cumplimiento de las mismas. También utilizará la aplicación para dar de alta a nuevos repartidores o de baja a aquellos que dejen la empresa. Generará informes basados en consultas extraídas de los datos de la aplicación, así como los cuadros de facturación que se utilizarán para facturar a Eroski mensualmente por las entregas realizadas.	
Análisis de tareas	
▪ Identificarse en la aplicación mediante el formulario de login ▪ Comprobar el estado de los pedidos de cada tienda ▪ Posibilidad de realizar informes de pedidos, por centro, fechas, etc. ▪ Generar cuadros de facturación mensual ▪ Cerrar sesión	
Características a incorporar tras fase de indagación	
▪ Debido al uso recurrente de Excel en la organización, todas las consultas y resultados de cuadros de facturación han de poderse exportar a ese formato.	

2.2 Diseño conceptual

2.2.1 Personas

Para comprender mejor las necesidades, aspiraciones y frustraciones de los usuarios, se van a personificar los perfiles identificados en Nelson (repartidor) y Pilar (gestión). La personificación de ambos perfiles se basa en la información recopilada en la fase de indagación y, por ejemplo, Nelson representa a un empleado tipo de la empresa que realiza entregas diarias. Por otro lado, Pilar representa a los usuarios que realizan la gestión de la operativa diaria.

Nelson	Perfil Repartidor

Nelson tiene 28 años, está casado desde hace 3 y tiene una hija de 1 año. Llegó a España hace ya 10 años y desde entonces ha trabajado en diferentes empresas relacionadas con el reparto. Antes de trabajar en Blaniza, estuvo trabajando 3 años en una empresa de reparto de comida a domicilio, por lo que conoce bien la ciudad y sus entresijos.

Tiene un móvil Android de gama media que utiliza de forma personal principalmente para comunicarse vía WhatsApp, pero no utiliza ninguna otra aplicación porque no se considera muy "tecnológico" y prefiere mantener su uso de tecnología al mínimo.

Nelson comienza la jornada revisando los pedidos en el grupo de WhatsApp para ver qué trabajo tiene pendiente. A veces tiene dificultades para saber el trabajo que le corresponde y eso le genera cierta confusión. Le gustaría cambiar de forma de trabajar, para tener claro qué es lo que tiene que hacer y para completar sus entregas de forma más eficiente y poder volver antes a casa para pasar más tiempo con su familia.

Pilar	Perfil Gestión

Pilar tiene 45 años, es de Zaragoza, está casada y tiene 2 hijos. Ha estado realizando trabajos administrativos en diferentes empresas de sectores muy diversos y desde hace 3 años trabaja en Logística Blaniza realizando la carga de los pedidos, la coordinación de los repartidores y la facturación.

Pilar utiliza un ordenador portátil y un Smartphone de gama alta proporcionados por la empresa para llevar a cabo sus tareas diarias. Está familiarizada con paquetes de software ofimático y de gestión de facturación y contable.

Pilar emplea gran parte de su jornada en descargar los ficheros de los pedidos nuevos y traspasarlos a hojas Excel para su gestión. Esto le genera bastante estrés porque además de ser una tarea repetitiva, es algo que debe realizar de manera periódica y necesita de una buena organización para que no se le pase ningún pedido, lo cual a veces ocurre. No entiende que no haya un sistema automático que haga la carga de los pedidos y eso a veces le genera frustración.

También le toca lidiar con los repartidores, ya que con el grupo de WhatsApp que ella gestiona y donde se colocan los pedidos de cada tienda, se generan confusiones y problemas de coordinación. Muchas veces, los repartidores le llaman enfadados y ella tiene que gestionar estas situaciones, lo que le genera más estrés.

Cuando llega la hora de realizar la facturación mensual, es la encargada de generar con las Excel los cuadros de facturación que luego se presentan a Eroski. Esto lo hace de forma manual y en algún momento se ha producido algún error que le ha costado algún disgusto.

Pilar ya se ha quejado en varias ocasiones a la dirección de la empresa para que se busque una solución que le permita gestionar y supervisar de manera eficiente los repartos, reducir su carga de trabajo administrativa y mejorar la comunicación con los repartidores.

2.2.2 Escenarios de uso y flujos de interacción

Escenario de uso 1 – Consulta de pedidos pendientes

Perfil de usuario	Repartidor
Contexto	Al comienzo de la jornada, Nelson recoge su furgoneta preparándose para recoger y entregar los pedidos del día.
Objetivos	Identificar y priorizar los pedidos pendientes que debe recoger y entregar
Tareas	Acceder a la aplicación e iniciar sesión Seleccionar una de las tiendas Revisar la lista de pedidos pendientes de la tienda seleccionada
Necesidades de Información	Detalles de los pedidos, principalmente dirección de entrega y hora de entrega mínima y máxima.
Funcionalidades	Consulta de pedidos pendientes
Desarrollo de las tareas	Nelson abre la aplicación en su smartphone, inicia sesión y ve la lista de todas las tiendas, en las que aparecen las que tiene asignadas en primer lugar. Despliega la información de la tienda que corresponda y ve los pedidos que hay pendientes de entregar en esa tienda y al final de la lista, los pedidos entregados. Nelson planifica el orden y las ruta de recogida y entrega en base a la información proporcionada.

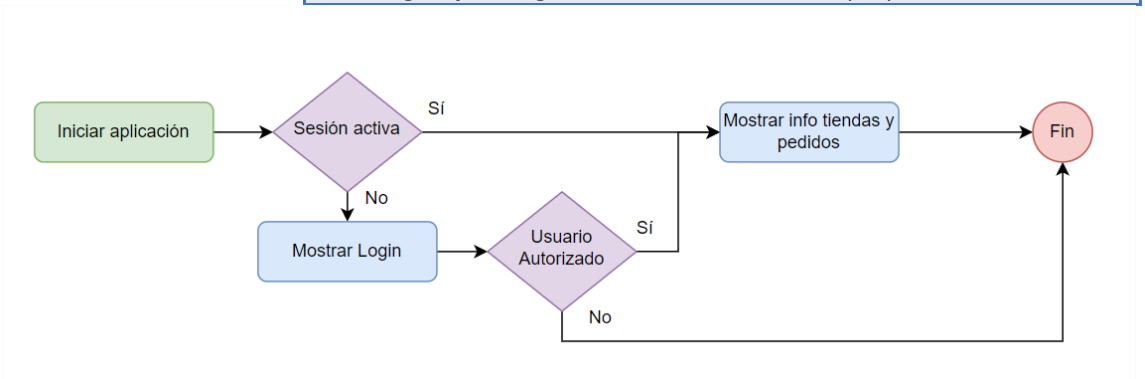


Figura 3 – Diagrama de flujo de consulta de pedidos pendientes

Escenario de uso 2 – Entrega realizada

Perfil de usuario	Repartidor
Contexto	Nelson ha llegado al domicilio del cliente para entregar un pedido
Objetivos	Completar la entrega y actualizar el estado del pedido en la aplicación
Tareas	Entregar el pedido al cliente Obtener confirmación de entrega Acceder a la aplicación e iniciar sesión Localizar pedido del cliente Actualizar el estado del pedido en la aplicación
Necesidades de Información	Detalles del pedido y contacto del cliente.
Funcionalidades	Actualización del estado del pedido
Desarrollo de las tareas	Nelson entrega la mercancía al cliente y obtiene una confirmación de la entrega. Después Nelson abre la aplicación en su smartphone, busca la tienda y el pedido correspondiente y utiliza el botón de cambio de estado para cambiar el estado del pedido a “Entregado”.

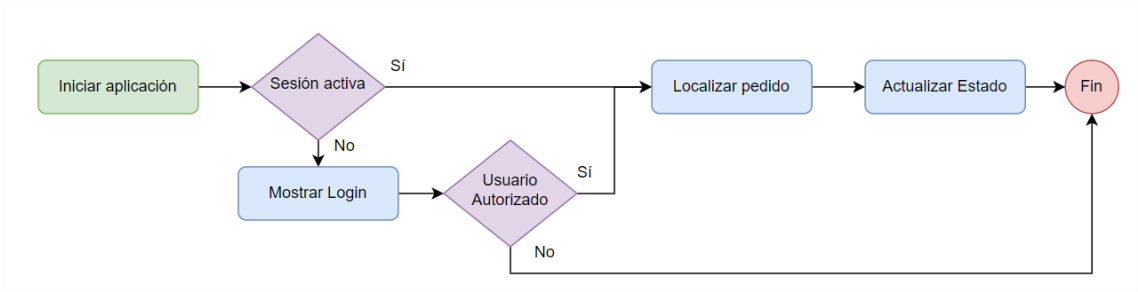


Figura 4 – Diagrama de flujo de entrega realizada

Escenario de uso 3 – Informe de pedidos

Perfil de usuario	Gestión
Contexto	Pilar se encuentra en la oficina revisando el rendimiento y la eficiencia del reparto en el periodo horario que se establezca (diario, semanal, mensual, etc.)
Objetivos	Evaluar el desempeño del equipo de reparto y analizar los datos de pedidos realizados
Tareas	Acceder a la aplicación e iniciar sesión Acceder al menú de administración (solo visible para Gestores) Seleccionar la opción de Administración de pedidos Seleccionar un intervalo de fechas (o todas) Seleccionar una tienda (o todas) Pulsar el botón de generación de informe Revisar datos de informe Exportar datos a Excel (si se desea)
Necesidades de Información	Datos de pedidos realizados y no realizados. Datos de repartidores involucrados.
Funcionalidades	Generación de informes
Desarrollo de las tareas	Pilar abre la aplicación desde su ordenador, accediendo a la URL desde su navegador. Tras iniciar sesión, accede al menú de administración y pulsa sobre la opción de administración de pedidos. Pilar selecciona una fecha de inicio y otra de fin para acotar los resultados del listado y posteriormente decide si quiere obtener el listado de todas las tiendas o de una en concreto. Tras pulsar el botón para generar el informe, Pilar decide exportar los datos de la consulta a Excel con el botón correspondiente.

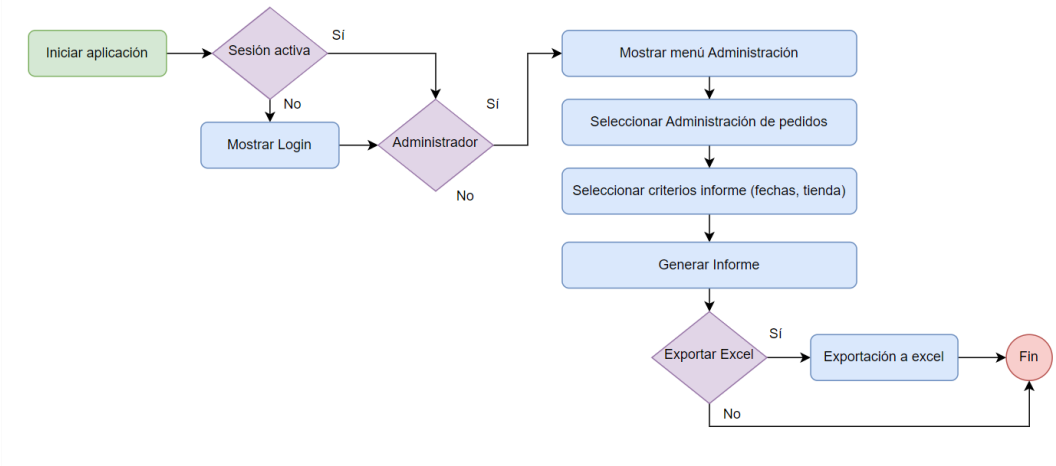


Figura 5 – Diagrama de flujo de informe de pedidos

Escenario de uso 4 – Cuadro de facturación

Perfil de usuario	Gestión
Contexto	Pilar se encuentra en la oficina a final de mes y es el momento de generar los cuadros de facturación para Eroski.
Objetivos	Obtener los datos para generar la factura correspondiente a las entregas realizadas en el último mes.
Tareas	Acceder a la aplicación e iniciar sesión Acceder al menú de administración (solo visible para Gestores) Seleccionar la opción de Cuadro de facturación Seleccionar un intervalo de fechas (o todas) Seleccionar una tienda (o todas) Pulsar el botón de generación de informe Revisar datos de informe Exportar datos a Excel (si se desea)
Necesidades de Información	Datos de pedidos realizados en el último mes
Funcionalidades	Consulta de pedidos entregados en un periodo específico en formato de cuadro mensual
Desarrollo de las tareas	Pilar abre la aplicación desde su ordenador, accediendo a la URL desde su navegador. Tras iniciar sesión, accede al menú de administración y pulsa sobre la opción de cuadro de facturación. Pilar selecciona una fecha de inicio y otra de fin para acotar los resultados del listado (Normalmente de 26 de mes anterior a 25 de mes actual) y posteriormente decide si quiere obtener el cuadro de todas las tiendas o de una en concreto. Tras pulsar el botón para generar el informe, Pilar decide exportar los datos del cuadro de facturación a Excel con el botón correspondiente.

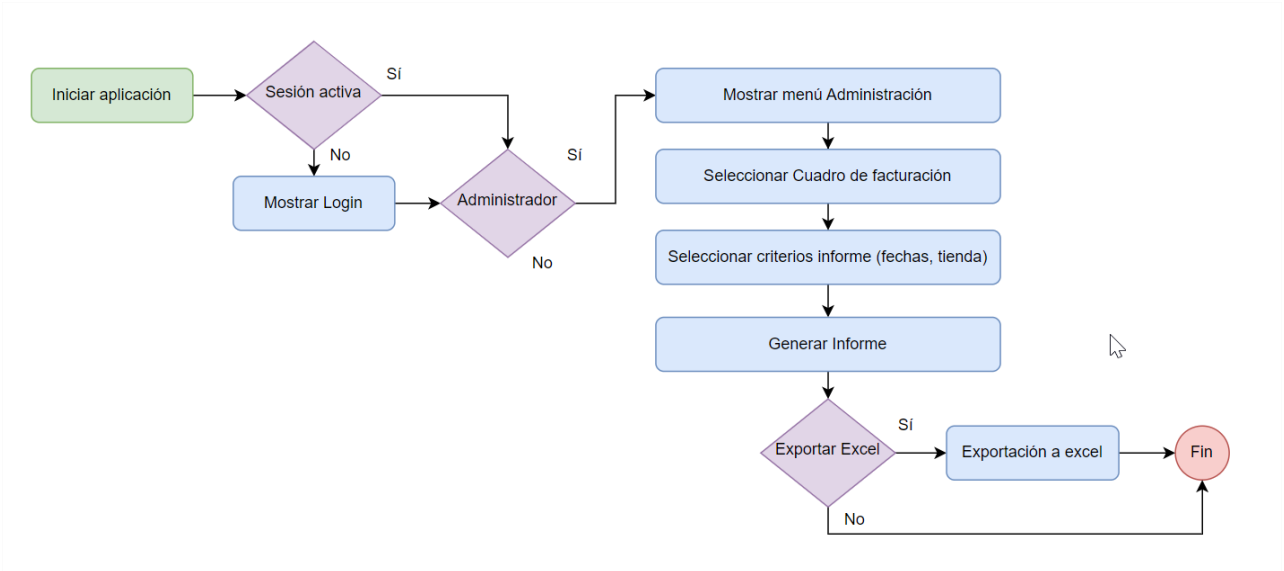


Figura 6 – Diagrama de flujo de cuadro de facturación

2.3 Prototipado

2.3.1 Sketches

A continuación, se detallan los primeros sketches de diseño hechos a mano alzada de las diferentes opciones de la aplicación. Como dicha aplicación está diseñada para ser utilizada tanto a través de una aplicación móvil (Android) como desde un entorno de navegador de escritorio, se detallarán las partes correspondientes a la aplicación móvil (que utilizarán los repartidores) en formato de pantalla de dispositivo móvil, y las pantallas correspondientes a las opciones de administración, en formato de pantalla panorámica de escritorio:

Login de Usuario

Primera pantalla de la aplicación que contendrá dos cajas de texto, (una para el usuario y otra para la contraseña) además de un botón que validará los datos.

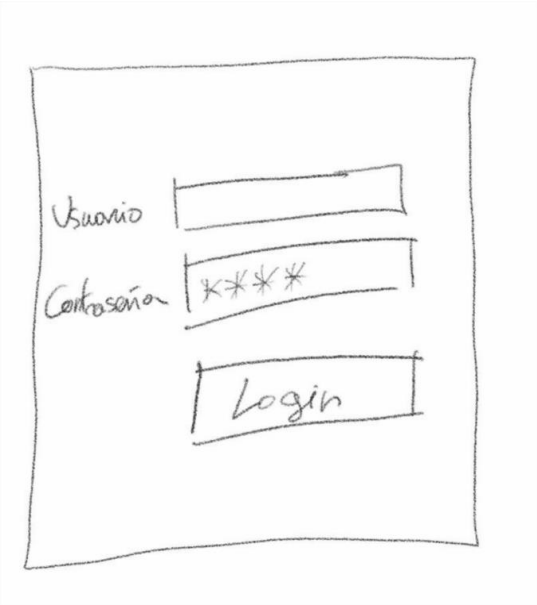


Figura 7 – Sketch de pantalla de inicio de sesión

Pantalla principal de pedidos

Tras hacer login, aparecerán las tiendas con los pedidos del día. Pulsando en una tienda, se desplegarán sus pedidos asociados

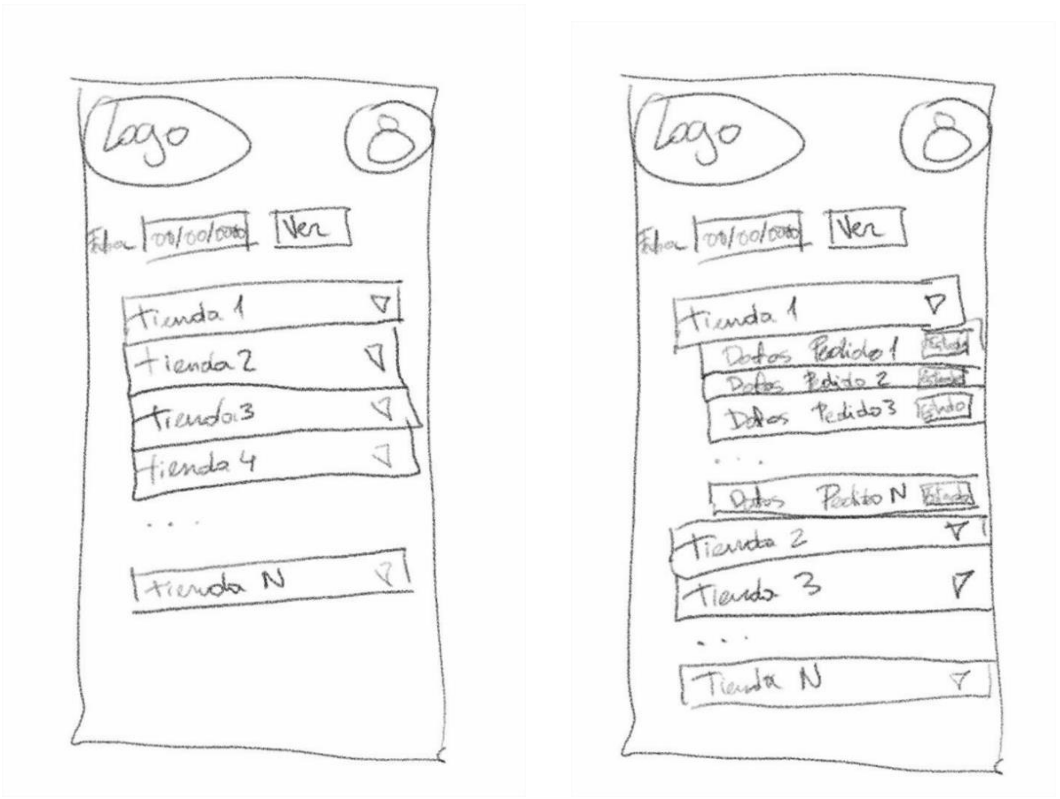


Figura 8 – Sketches de pantalla principal de pedidos

Consulta de pedidos de Administración

Esta ventana permitirá al personal de administración consultar pedidos entregados en un periodo definido entre dos fechas, así como filtrar los pedidos por una tienda en concreto o ver los de todas.

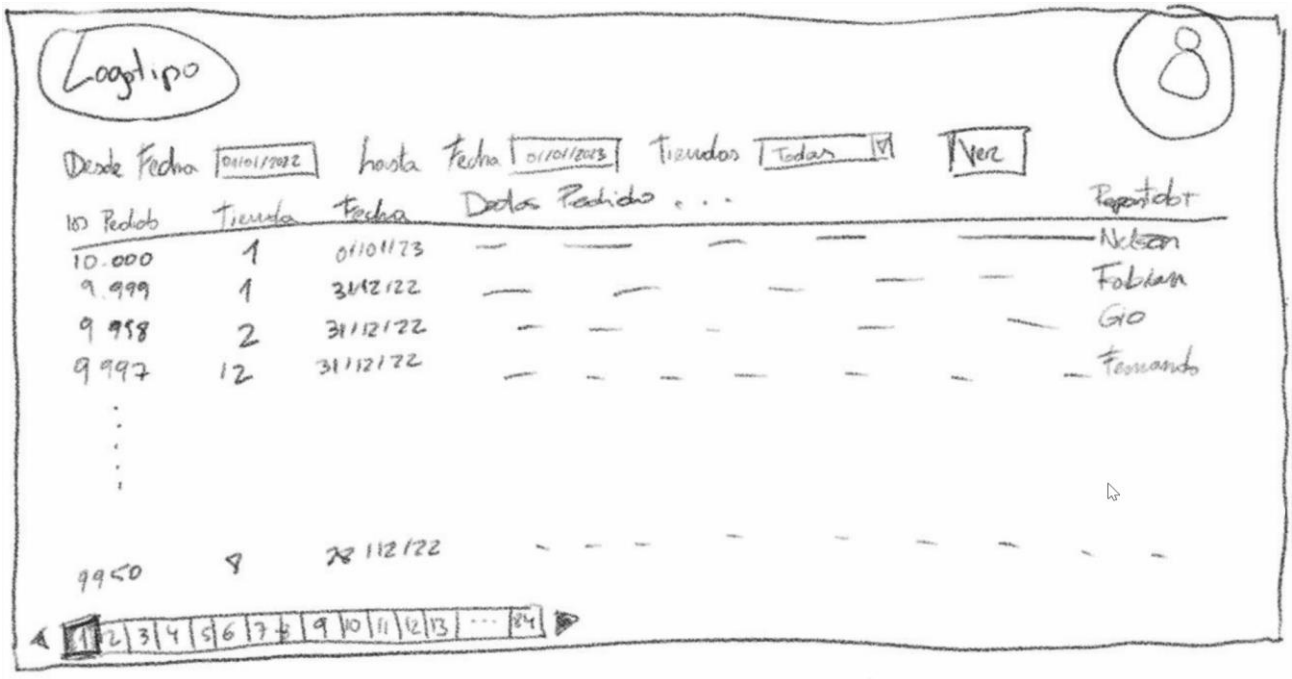


Figura 9 – Sketch de pantalla de pedidos de administración

Cuadro de facturación

Esta opción permitirá generar un cuadro con el desglose de los pedidos de cada tienda en un periodo determinado, que por defecto será mensual. Esta será la información que se entregue con la facturación que se le haga a Eroski.

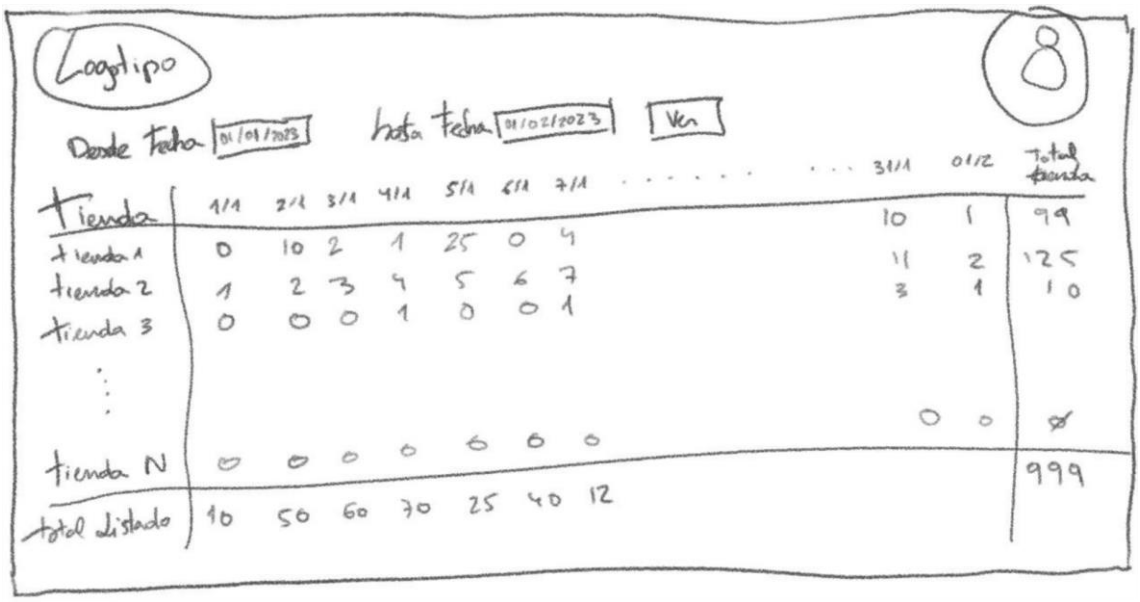


Figura 10 – Sketch de cuadro de facturación

Gestión de usuarios

Desde esta opción se podrá consultar los datos de los usuarios activos de la aplicación, así como utilizar el botón 'Nuevo Usuario' para crear un nuevo usuario en el sistema. También desde esta opción se darán de baja un usuario existente.

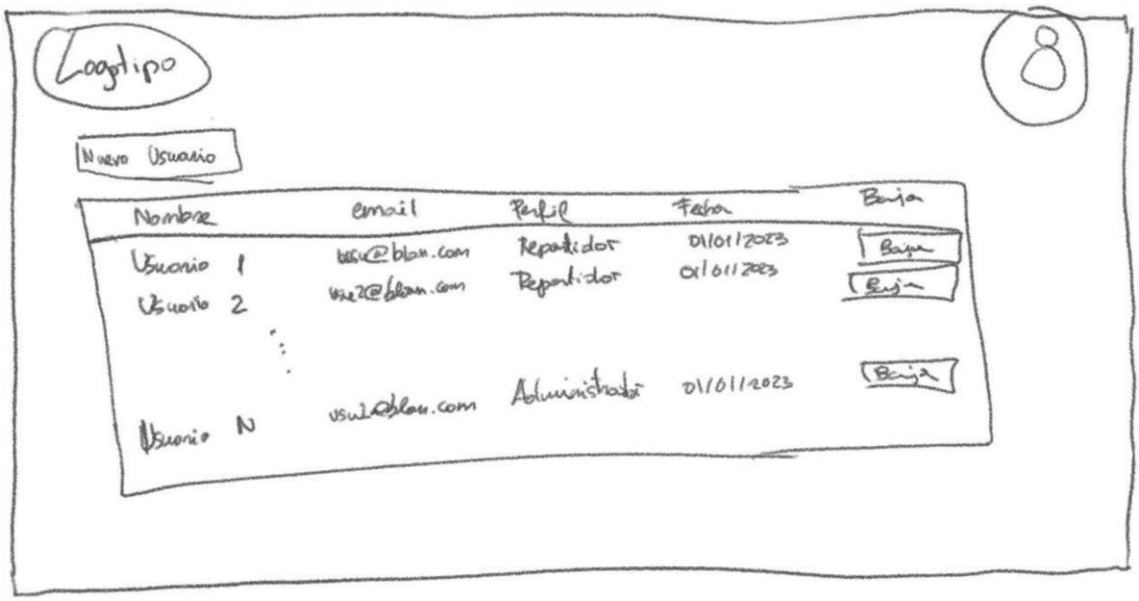


Figura 11 – Sketch de pantalla de gestión de usuarios

Al pulsar el botón 'Nuevo usuario', nos llevará a esta pantalla donde rellenar los datos de un nuevo usuario:

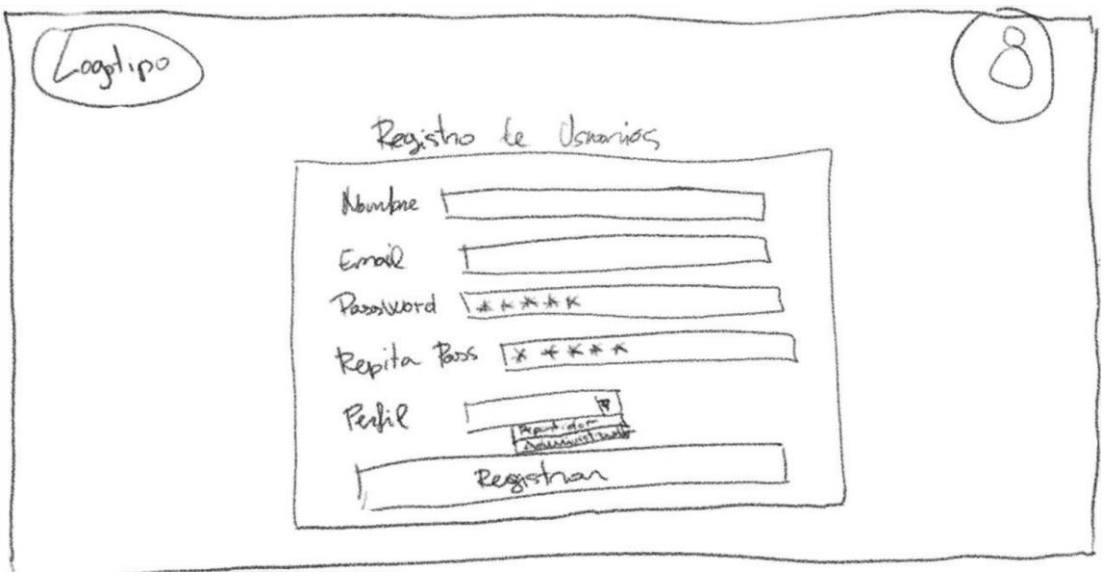


Figura 12 – Sketch de pantalla de registro de usuarios

2.3.2 Prototipo horizontal de alta fidelidad

Pantalla de inicio de sesión en versión aplicación móvil:

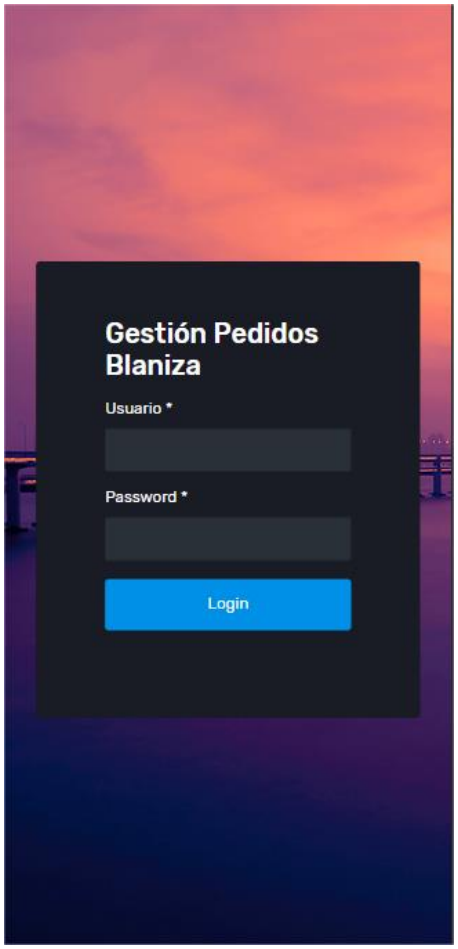


Figura 13 – Prototipo horizontal: Pantalla de inicio de sesión versión móvil

Ventana de inicio de sesión en versión escritorio:

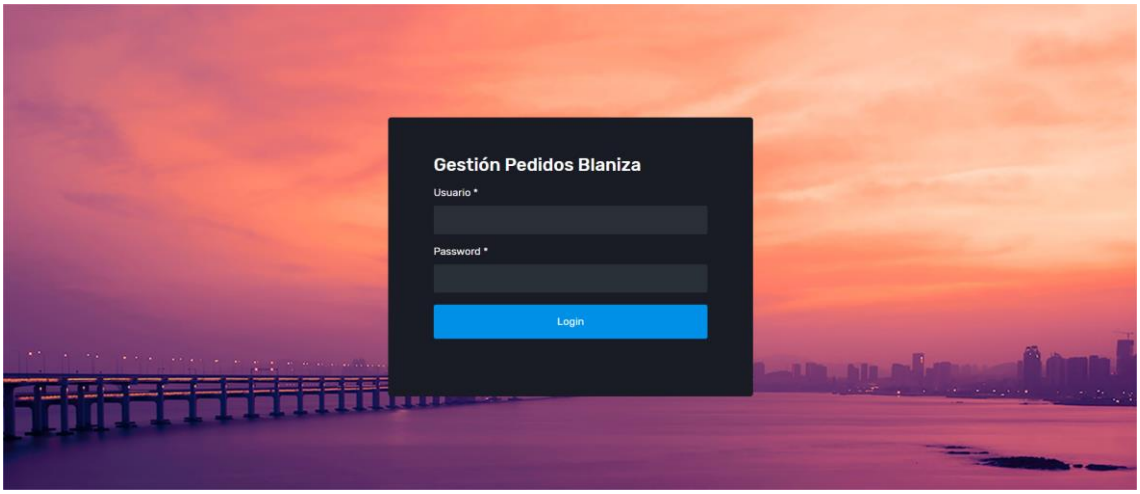


Figura 14 – Prototipo horizontal: Pantalla de inicio de sesión versión escritorio

Pantalla principal de pedidos en versión aplicación móvil:

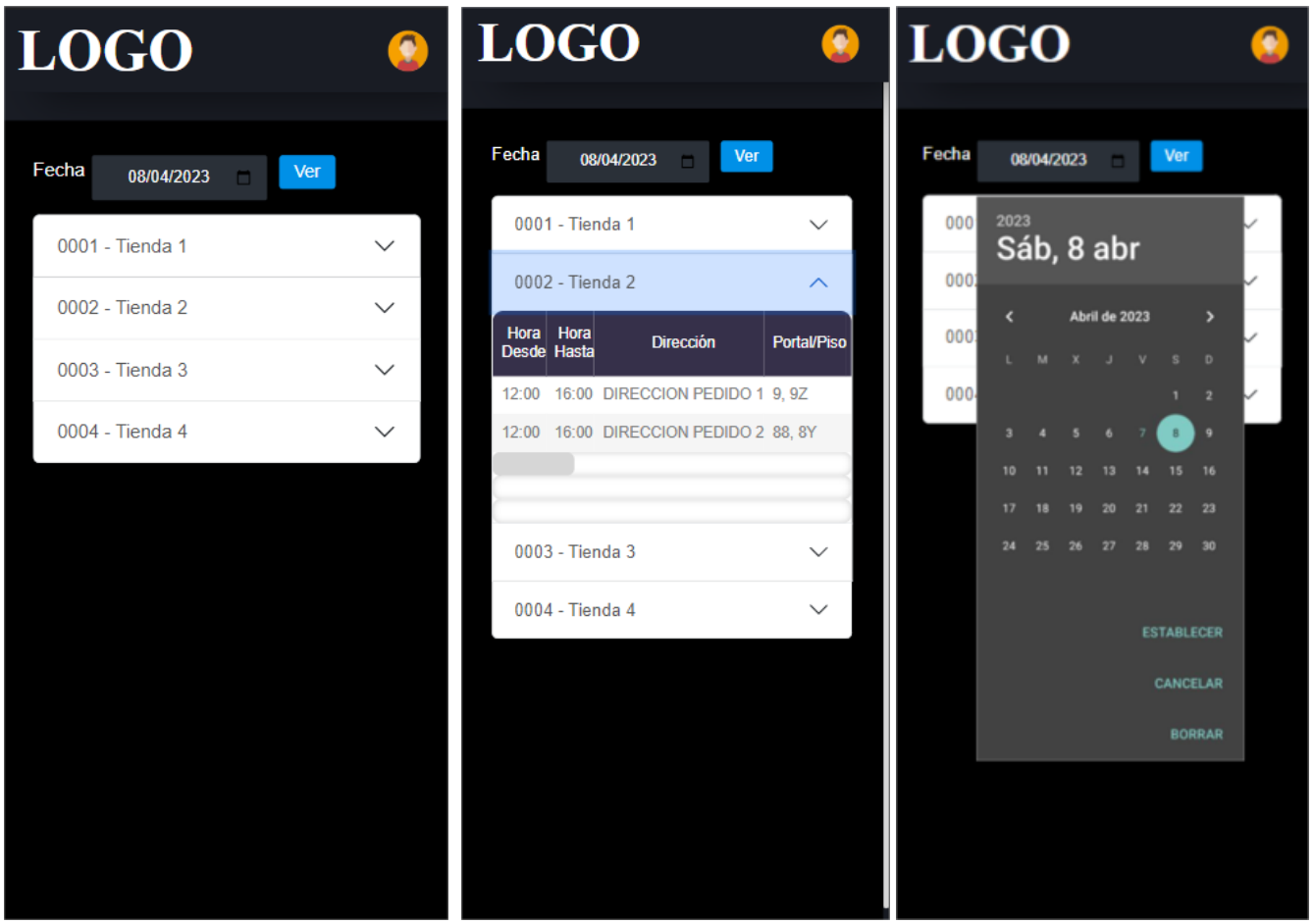


Figura 15 - Prototipo horizontal: Pantalla principal de pedidos versión móvil

Pantalla principal de pedidos en versión escritorio:

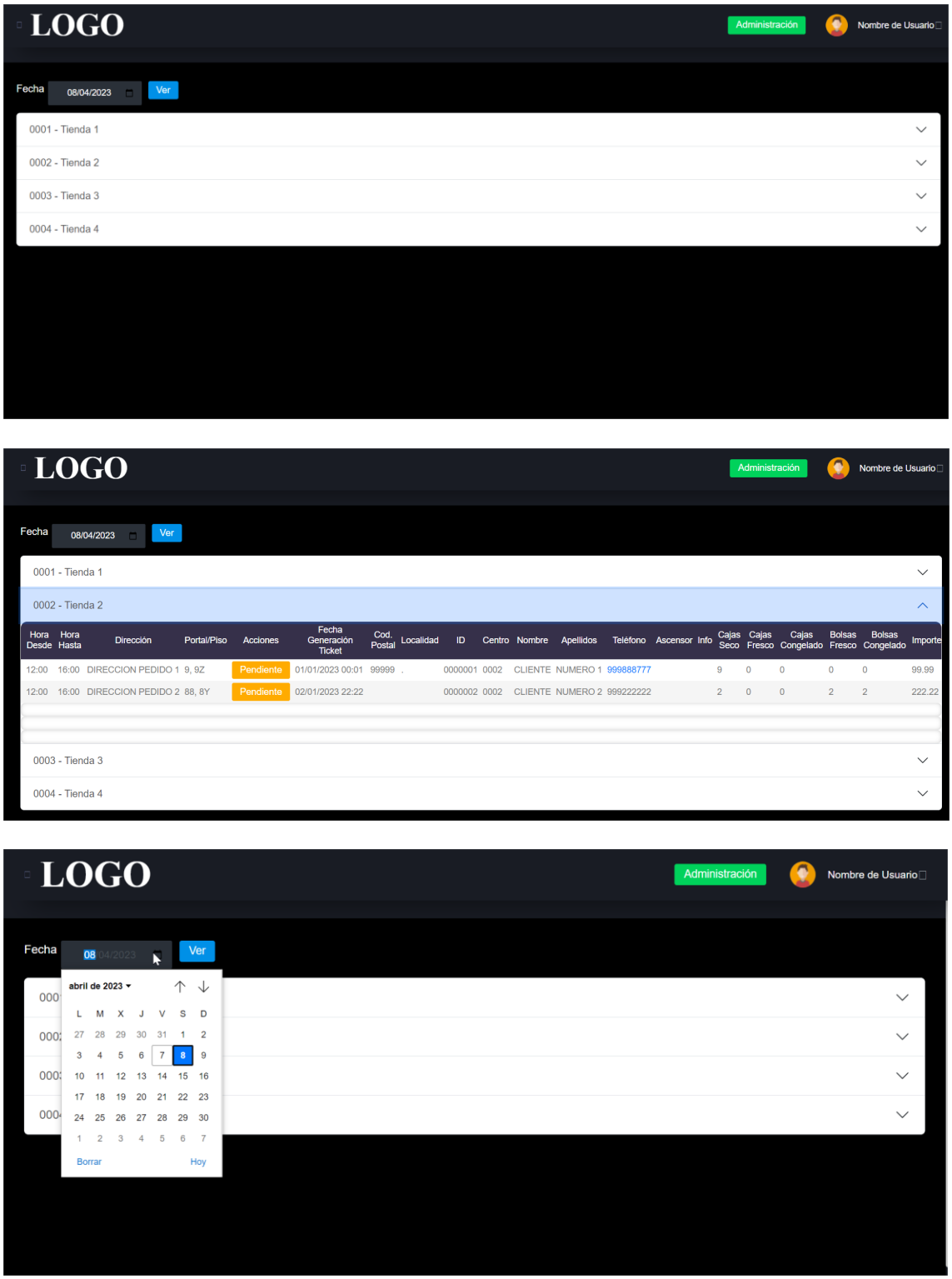


Figura 16 - Prototipo horizontal: Pantalla principal de pedidos versión escritorio

Modificación de estado de pedido en versión móvil:

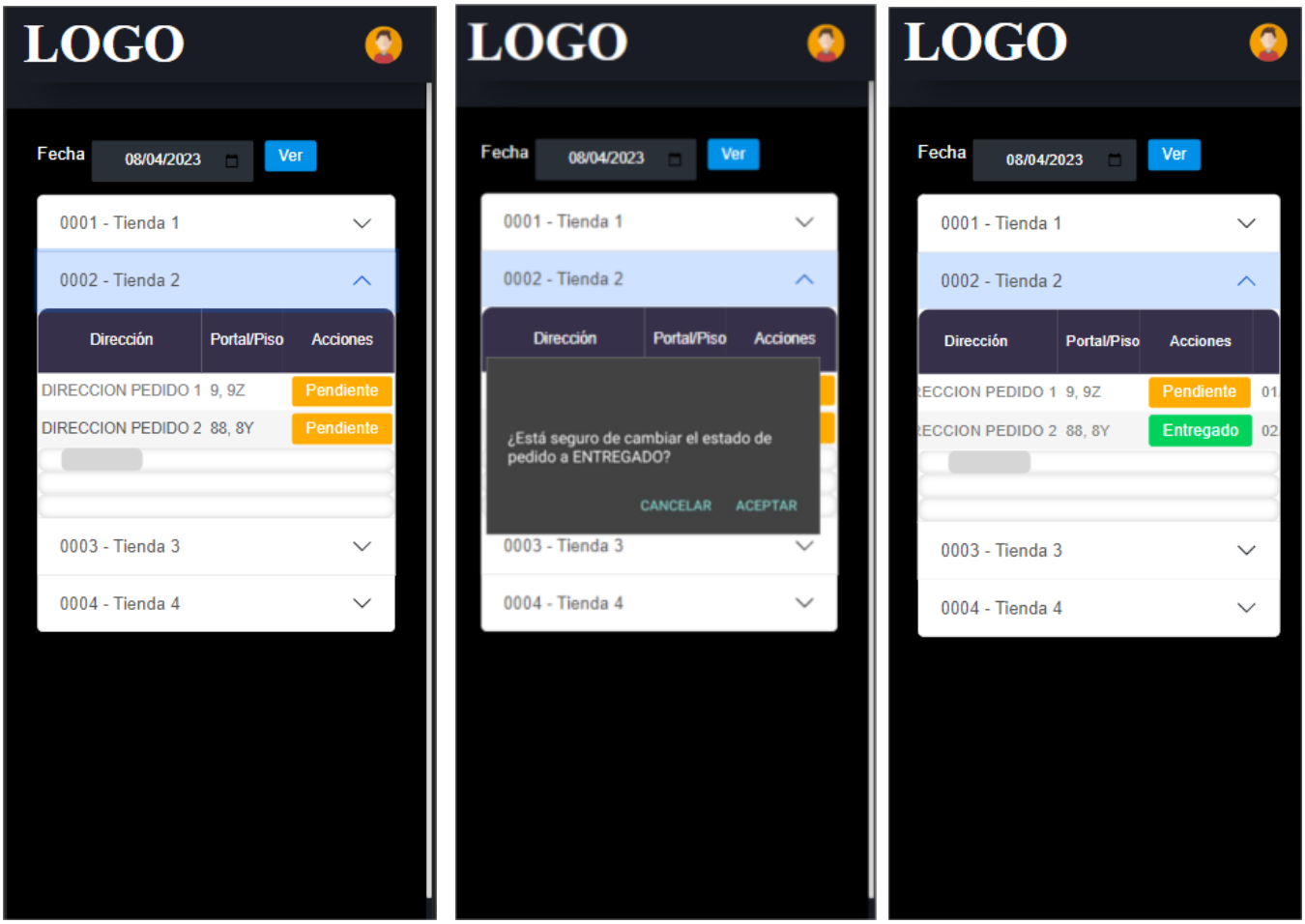
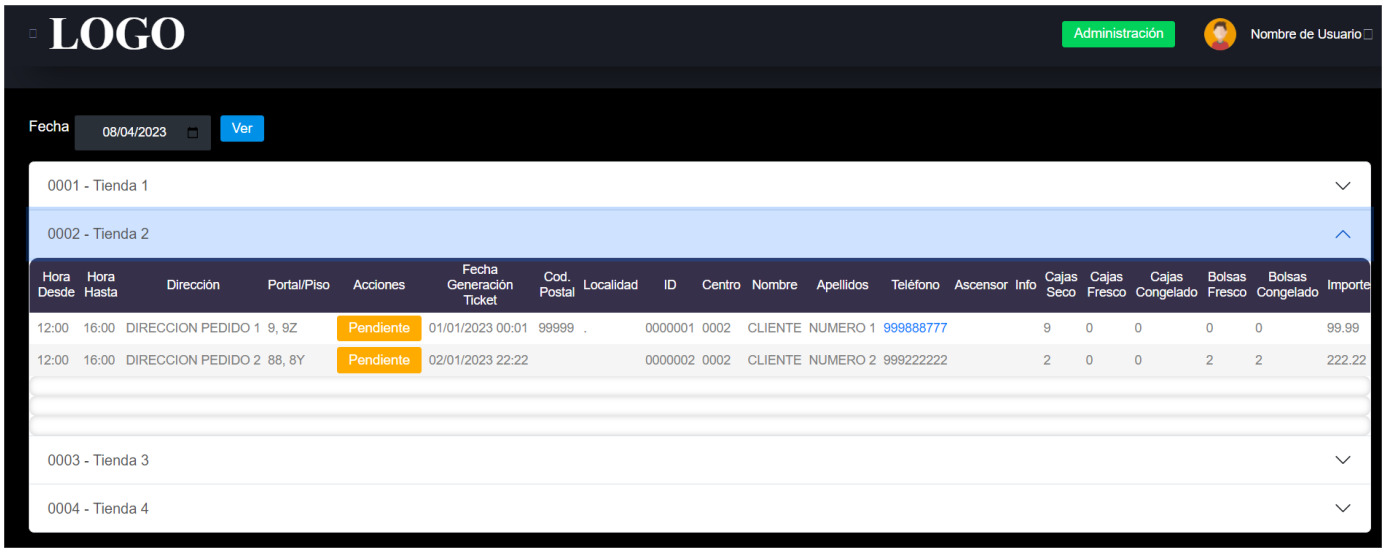


Figura 17 – Prototipo horizontal: Modificación de estado de pedido versión móvil

Modificación de estado de pedido en versión escritorio:



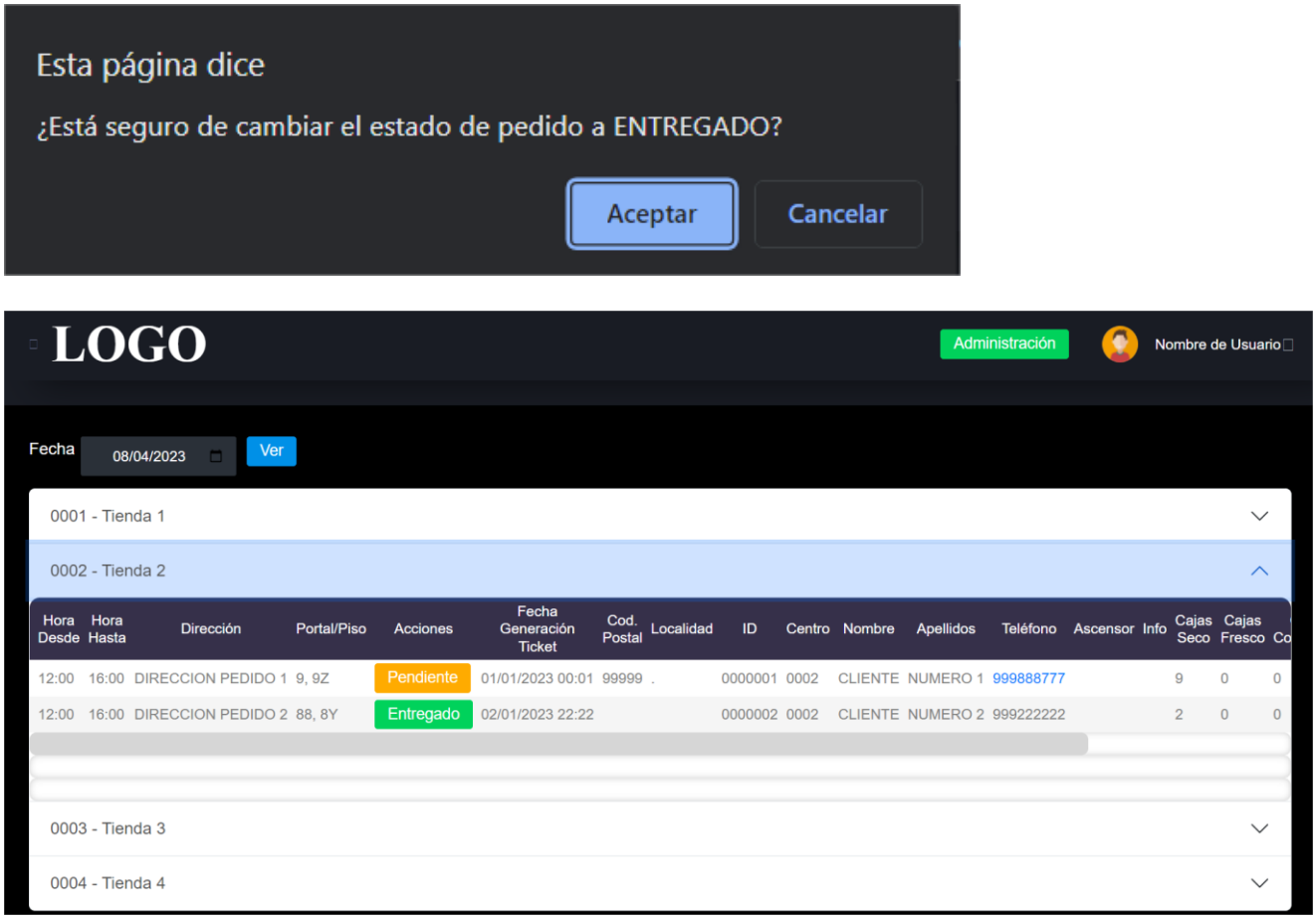


Figura 18 - Prototipo horizontal: Modificación de estado de pedido versión escritorio

Opciones de administración disponibles para usuarios con perfil ADMINISTRADOR:

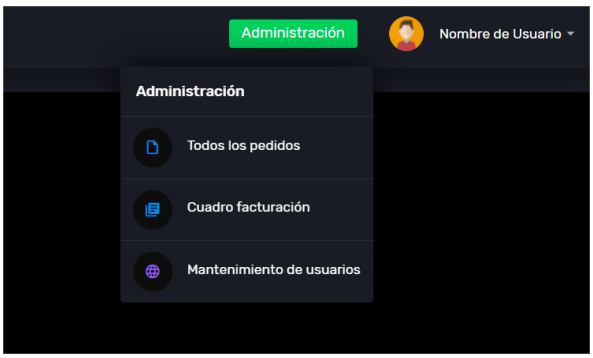


Figura 19 – Prototipo horizontal: Opciones menú de administración

Cuadro de facturación:

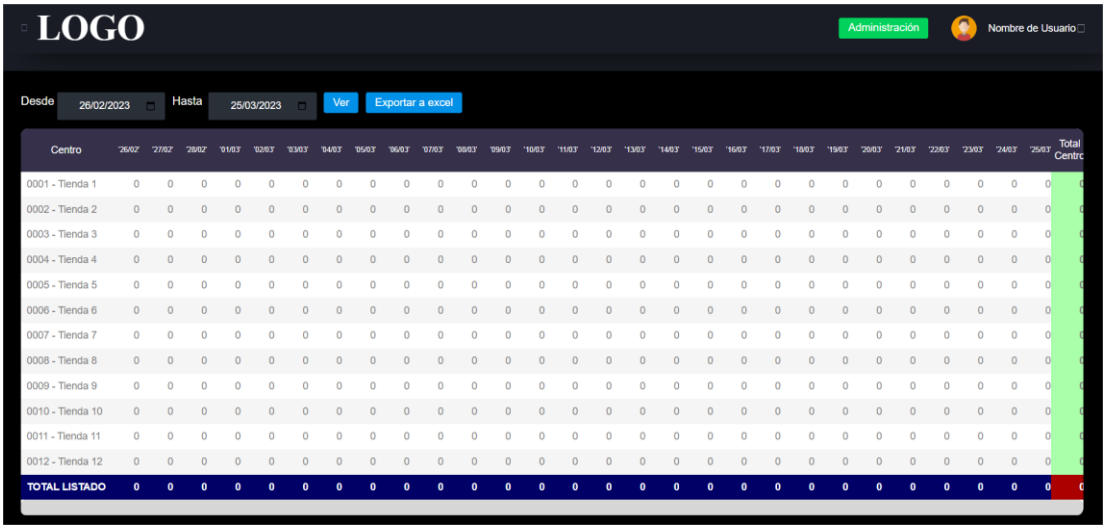


Figura 20 - Prototipo horizontal: Cuadro de facturación

Mantenimiento de usuarios del sistema:

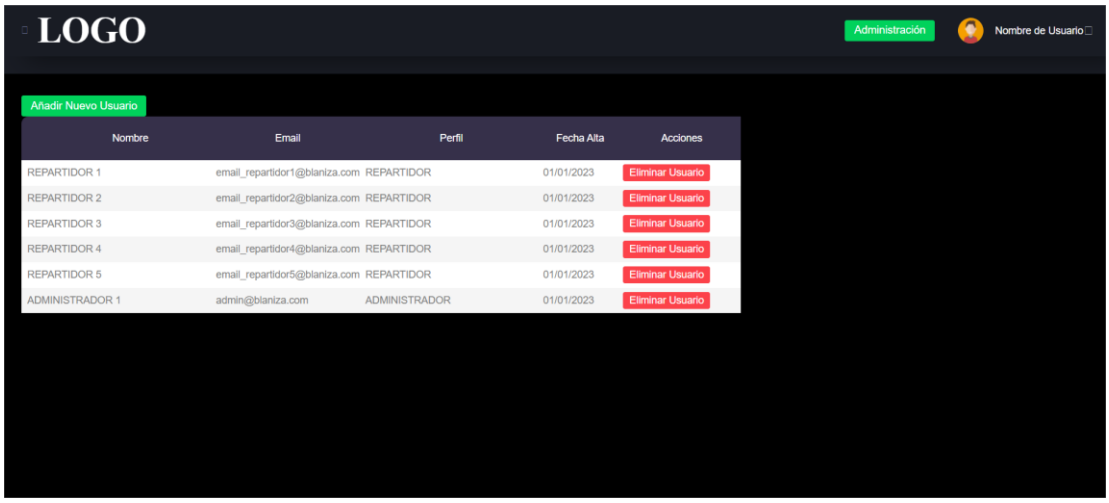


Figura 21 – Prototipo horizontal: Gestión de usuarios

Registro de nuevo usuario:

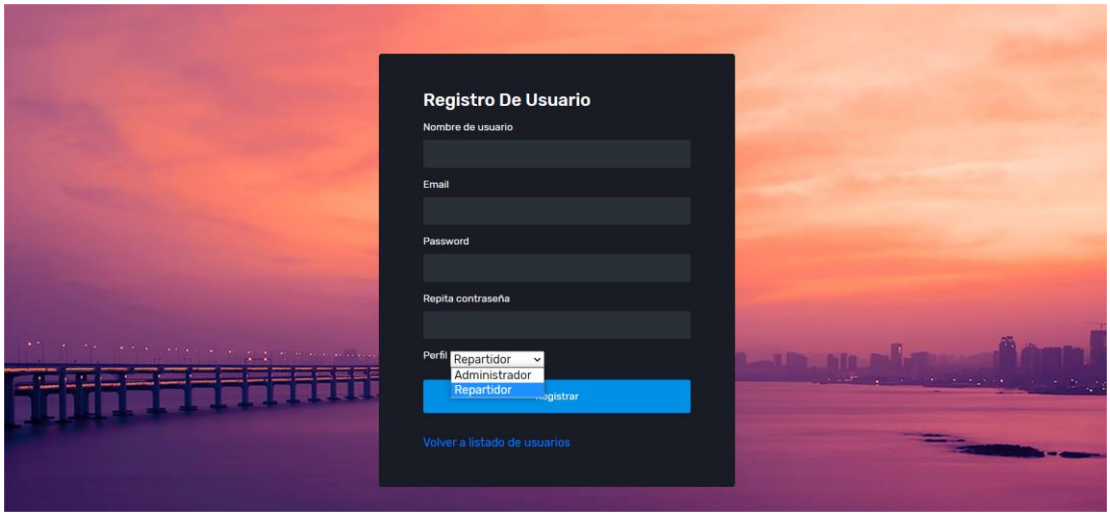


Figura 22 – Prototipo horizontal: Registro de nuevo usuario

2.4 Evaluación

Durante la fase de evaluación se pondrá a prueba la usabilidad y funcionalidades del prototipo, a través de una serie de actividades con los usuarios finales. El objetivo es identificar posibles problemas de usabilidad, detectar puntos de mejora y validar que el diseño cumpla con las expectativas y necesidades de los usuarios finales de la aplicación.

Para realizar esta fase, los usuarios tendrán que contestar a una serie de preguntas que permitirán obtener información específica del usuario. Posteriormente, en el laboratorio, se le pedirá a cada usuario que realice ciertas funciones con el prototipo y finalmente se le harán una serie de preguntas para obtener las impresiones del usuario.

Finalmente se analizarán los resultados obtenidos y se propondrán mejoras en función de las sugerencias y comentarios recibidos por parte de los usuarios. Esto permitirá el refinamiento del prototipo en una nueva iteración del proceso de diseño.

2.4.1 Preguntas de información de usuario

Las preguntas para obtener información específica de los usuarios serán las siguientes:

- ¿Cuál es su nombre?
- ¿Cuál es su edad?
- ¿Qué nivel de estudios tiene?
- ¿Qué puesto desempeña en la empresa?
- ¿Cuánto tiempo lleva trabajando en el sector del reparto o gestión logística?
- ¿Qué tipo de dispositivos móviles y sistemas operativos utilizas habitualmente?
- ¿Con qué frecuencia utilizas aplicaciones en tu dispositivo?

2.4.2 Tareas

Diferenciaremos dos tipos de tareas a realizar con el prototipo. Por una lado las de los repartidores y por otro las correspondientes al personal de gestión.

Tareas a realizar por los repartidores:

- Iniciar sesión
- Consultar los pedidos entregados en una tienda en una fecha determinada
- Consultar los pedidos pendientes de entregar
- Localizar un pedido concreto
- Modificar el estado de un pedido

Tareas a realizar por el personal de gestión:

- Iniciar sesión
- Consultar los pedidos entregados en una tienda en una fecha determinada
- Obtener un informe de pedidos entre un intervalo de fechas en una tienda determinada
- Obtener un cuadro de facturación de un mes determinado y exportarlo a excel
- Dar de alta un usuario nuevo
- Dar de baja un usuario existente

2.4.3 Preguntas referentes a las tareas

Una vez los usuarios hayan completado sus tareas con el prototipo, se les realizarán las siguientes preguntas:

- ¿Pudo completar todas las tareas?
- ¿Qué opinión general tiene del prototipo? ¿Lo considera atractivo?
- ¿Fue facil de usar?
- ¿Hubo alguna tarea que le pareció más complicada?
- ¿Qué opina de la forma de presentar la información?
- ¿Qué mejoras propone?

2.4.4 Análisis de resultados

Tras llevar a cabo la evaluación del prototipo, se ha recopilado información para la mejora de la experiencia de uso y funcionalidad del sistema. Los usuarios han sugerido que sería útil incluir en el listado de tiendas el número de pedidos entregados sobre el total de pedidos del día, ya que eso les permitirá ver de un vistazo el progreso de las entregas. También creen que podría ser bueno utilizar códigos de colores para resaltar en verde las tiendas que tienen todos sus pedidos entregados. Además, han comentado que el horario de entrega y la dirección de entrega son los datos más útiles para ellos a la hora de ver los pedidos y que está bien que aparezcan en primer lugar. También valoran positivamente que el botón de cambio de estado esté justo después de esos datos y no al final del resto de datos del pedido, ya que ellos apenas usan el resto de datos.
En cuanto a la parte de gestión, los usuarios están conformes con las funcionalidades y no han hecho ningún comentario reseñable.

En base a estas conclusiones, se incorporarán los cambios sugeridos y se volverá a realizar una nueva ronda de evaluación para validar dichas mejoras.

3. Diseño técnico

3.1 Casos de uso

3.1.1 Diagrama UML de casos de uso

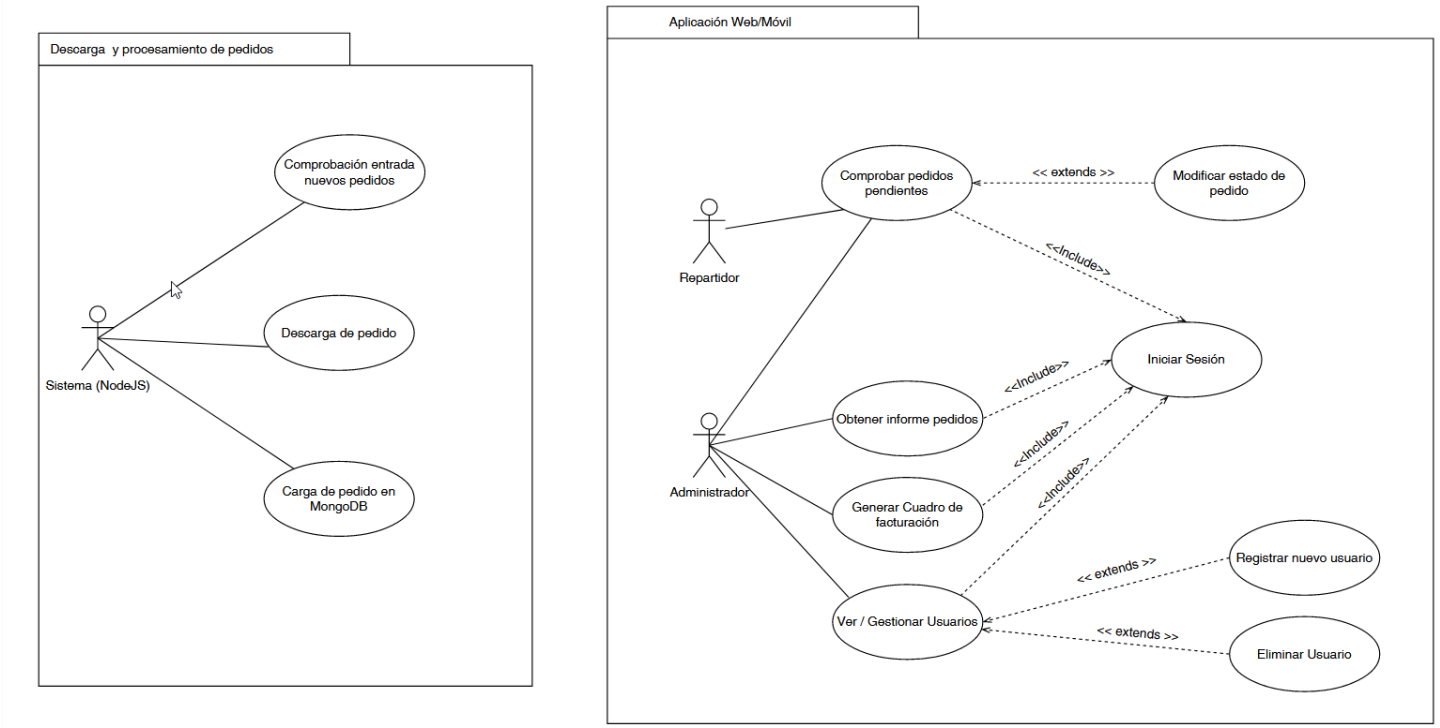


Figura 23 – Diagrama de casos de uso

3.1.2 Listado de casos de uso

ID: CU-001	
Nombre	Comprobación entrada nuevos pedidos
Prioridad	Alta
Descripción	Verificar si existen nuevos pedidos disponibles en la carpeta de entrada de pedidos del servidor FTP de Eroski.
Actores	Sistema (NodeJS)
Precondiciones	Estar conectado a la VPN de Eroski.
Iniciado por	Administrador del sistema
Flujo	1. La aplicación establece conexión con el servidor FTP de Eroski 2. Se comprueba la ruta de entrada de pedidos 3. Si existen ficheros en dicha carpeta, se obtiene una lista de los mismos 4. Si no existen pedidos nuevos, se cierra la conexión con el servidor FTP y se espera al siguiente ciclo
Postcondiciones	Se determina si existen pedidos nuevos para descargar o no. En caso de que existan se descarga una lista de los ficheros a descargar.
Notas	Este proceso se repite pasado un tiempo determinado que en principio se ha establecido en 1 minuto.

ID: CU-002	
Nombre	Descarga de pedido
Prioridad	Alta
Descripción	Descargar los ficheros de pedidos pendientes
Actores	Sistema (NodeJS)
Precondiciones	Estar conectado a la VPN de Eroski. Se dispone conexión con el servidor de FTP de Eroski. Se ha comprobado que han entrado pedidos nuevos y se dispone de la lista de ficheros remota.
Iniciado por	Administrador del sistema
Flujo	1. Se recorre la lista de ficheros nuevos 2. Para cada uno de los ficheros se solicita la descarga al servidor
Postcondiciones	Los ficheros de pedido se han descargado correctamente.
Notas	-

ID: CU-003	
Nombre	Carga de pedido en MongoDB
Prioridad	Alta
Descripción	Cargar los datos de pedidos nuevos descargados en la base de datos MongoDB
Actores	Sistema (NodeJS)
Precondiciones	Se dispone de los ficheros nuevos descargados
Iniciado por	Administrador del sistema
Flujo	1. Se procesa cada fichero descargado 2. Se extrae la información del fichero y se almacena en un objeto 3. Se establece conexión con la base de datos MongoDB 4. Se inserta la información del objeto en la base de datos 5. Si todo es correcto, se establece conexión con el servidor FTP y se copian los ficheros de la carpeta de entrada de pedidos a la carpeta de pedidos procesados 6. Se eliminan los ficheros locales y remotos 7. Se cierra conexión con el servidor FTP
Postcondiciones	Los ficheros de pedido se han cargado correctamente en la base de datos MongoDB. Se eliminan los pedidos nuevos de la carpeta remota de pedidos quedando lista para un nuevo ciclo de descarga.
Notas	-

ID: CU-004	
Nombre	Iniciar Sesión
Prioridad	Alta
Descripción	Permite que los usuarios de la aplicación (repartidores y administradores) inicien sesión en la aplicación y tras autenticarse, puedan acceder a sus respectivas funcionalidades.
Actores	Repartidor, Administrador
Precondiciones	El usuario debe estar registrado previamente en el sistema
Iniciado por	Repartidor, Administrador
Flujo	1. El usuario inicia la aplicación móvil o accede a la misma a través de un navegador web. 2. El usuario introduce su cuenta de usuario (email) y contraseña 3. El sistema verifica si las credenciales proporcionadas son válidas. 4. Si las credenciales son válidas, el sistema redirige a la pantalla de gestión de pedidos. Si no, muestra un mensaje de error. 5. Si el usuario tiene perfil 'ADMINISTRADOR', se muestra el botón de opciones de administración.
Postcondiciones	El usuario inicia sesión correctamente y se le permite acceder a las funcionalidades asociadas a su perfil.
Notas	Este caso de uso es incluido por casi todos los casos de uso de la aplicación, para asegurar que el usuario tenga acceso al sistema y a la funcionalidad correspondiente. El tiempo de sesión se establecerá en un valor lo suficientemente grande para que los usuarios (principalmente los repartidores) no tengan que estar continuamente introduciendo sus credenciales.

ID: CU-005	
Nombre	Comprobar pedidos pendientes
Prioridad	Alta
Descripción	Permite que los repartidores vean los pedidos entregados y pendientes de entrega de cada tienda.
Actores	Repartidor
Precondiciones	El repartidor debe haber iniciado sesión
Iniciado por	Repartidor
Flujo	1. El repartidor inicia la aplicación móvil e inicia sesión 2. El sistema muestra una lista, en donde aparecen en primer lugar las tiendas asignadas al repartidor y después el resto de tiendas. 3. El repartidor puede comprobar qué pedidos están entregados y cuales están pendientes en cada tienda que seleccione. Los pedidos pendientes aparecerán en primer lugar, seguido de los entregados.
Postcondiciones	Se muestran los pedidos entregados y pendientes de cada tienda
Notas	Este caso de uso incluye al caso de uso "Iniciar sesión" y extiende a "Modificar estado de pedido"

ID: CU-006	
Nombre	Modificar estado de pedido
Prioridad	Alta
Descripción	Permite que un repartidor cambie el estado de un pedido.
Actores	Repartidor
Precondiciones	El repartidor debe haber iniciado sesión y localizado un pedido concreto
Iniciado por	Repartidor
Flujo	1. El repartidor selecciona un pedido 2. El repartidor pulsa sobre el botón de estado de pedido para cambiar su estado de "Pendiente" a "Entregado" o viceversa. 3. El sistema actualiza el estado del pedido en la base de datos.
Postcondiciones	El estado del pedido se actualiza correctamente
Notas	Este caso de uso se extiende desde "Comprobar pedidos pendientes"

ID: CU-007	
Nombre	Obtener informe de pedidos
Prioridad	Normal
Descripción	Permite que un usuario administrador consulte y pueda generar un informe de pedidos en un periodo determinado, y de una o todas las tiendas.
Actores	Administrador
Precondiciones	El usuario debe haber iniciado sesión y haberse autenticado como administrador.
Iniciado por	Administrador
Flujo	1. El administrador accede a la aplicación e inicia sesión 2. El administrador accede a la opción de informe de pedidos. 3. El administrador elige las condiciones de selección si lo desea (intervalo de fechas y tienda) y pulsa el botón de generar el informe. 4. El sistema genera el informe y lo muestra. 5. El administrador exporta en Excel el informe si lo desea a través del botón correspondiente
Postcondiciones	Se muestra el informe de pedidos al administrador
Notas	Este caso de uso incluye al caso de uso "Iniciar sesión"

ID: CU-008	
Nombre	Generar cuadro de facturación
Prioridad	Normal
Descripción	Permite que un usuario administrador genere un cuadro de facturación mensual que posteriormente se remitirá al cliente (Eroski)
Actores	Administrador
Precondiciones	El usuario debe haber iniciado sesión y haberse autenticado como administrador.
Iniciado por	Administrador
Flujo	1. El administrador accede a la aplicación e inicia sesión 2. El administrador accede a la opción de cuadro de facturación. 3. El administrador elige el intervalo de fechas que contendrá el cuadro y pulsa el botón para generarlo. 4. El sistema genera el cuadro de facturación y lo muestra. 5. El administrador exporta en Excel el cuadro si lo desea a través del botón correspondiente
Postcondiciones	Se muestra el informe de pedidos al administrador
Notas	Este caso de uso incluye al caso de uso "Iniciar sesión". La generación del cuadro es normalmente mensual, así que cuando se seleccione "fecha desde", el sistema calculará un mes desde entonces y lo propondrá en "fecha hasta"

ID: CU-009	
Nombre	Ver/Gestionar Usuarios
Prioridad	Normal
Descripción	Permite que el administrador vea y gestione los usuarios del sistema.
Actores	Administrador
Precondiciones	El usuario debe haber iniciado sesión y haberse autenticado como administrador.
Iniciado por	Administrador
Flujo	1. El administrador accede a la aplicación e inicia sesión 2. El administrador accede a la opción de Gestión de usuarios 3. El sistema muestra la lista de usuarios del sistema. 4. El sistema permite al administrador registrar un nuevo usuario o eliminar un usuario existente.
Postcondiciones	Se muestra la información de los usuarios del sistema
Notas	Este caso de uso incluye al caso de uso "Iniciar sesión" y extiende a los casos de uso "Registrar nuevo usuario" y "Eliminar usuario"

ID: CU-010	
Nombre	Registrar nuevo usuario
Prioridad	Normal
Descripción	Permite que el administrador registre un nuevo usuario en el sistema.
Actores	Administrador
Precondiciones	El usuario debe haber iniciado sesión y haberse autenticado como administrador.
Iniciado por	Administrador
Flujo	1. El administrador accede a la opción de registro de nuevo usuario desde la opción “Ver/Gestionar usuarios” 2. El administrador introduce los datos del usuario 3. El administrador confirma el registro del nuevo usuario. 4. El sistema almacena los datos del usuario en la base de datos.
Postcondiciones	El nuevo usuario se registra correctamente en el sistema.
Notas	Este caso de uso se extiende desde “Ver/Gestionar usuarios”

ID: CU-011	
Nombre	Eliminar usuario
Prioridad	Baja
Descripción	Permite que el administrador elimine a un usuario existente del sistema.
Actores	Administrador
Precondiciones	El usuario debe haber iniciado sesión y haberse autenticado como administrador.
Iniciado por	Administrador
Flujo	1. El administrador selecciona un usuario para eliminar 2. El administrador confirma la eliminación del usuario 3. El sistema elimina los datos del usuario de la base de datos.
Postcondiciones	El usuario queda eliminado del sistema.
Notas	Este caso de uso se extiende desde “Ver/Gestionar usuarios”

3.2 Diseño de la arquitectura

3.2.1 Diagrama de Base de Datos

La base de datos que se va a utilizar en el proyecto va a ser MongoDB. Aunque MongoDB es una base de datos NoSQL, y por tanto, no se basa en un esquema de diseño relacional, se incluye el siguiente diagrama físico de base de datos para la identificación de las estructuras de datos y las relaciones entre ellas:

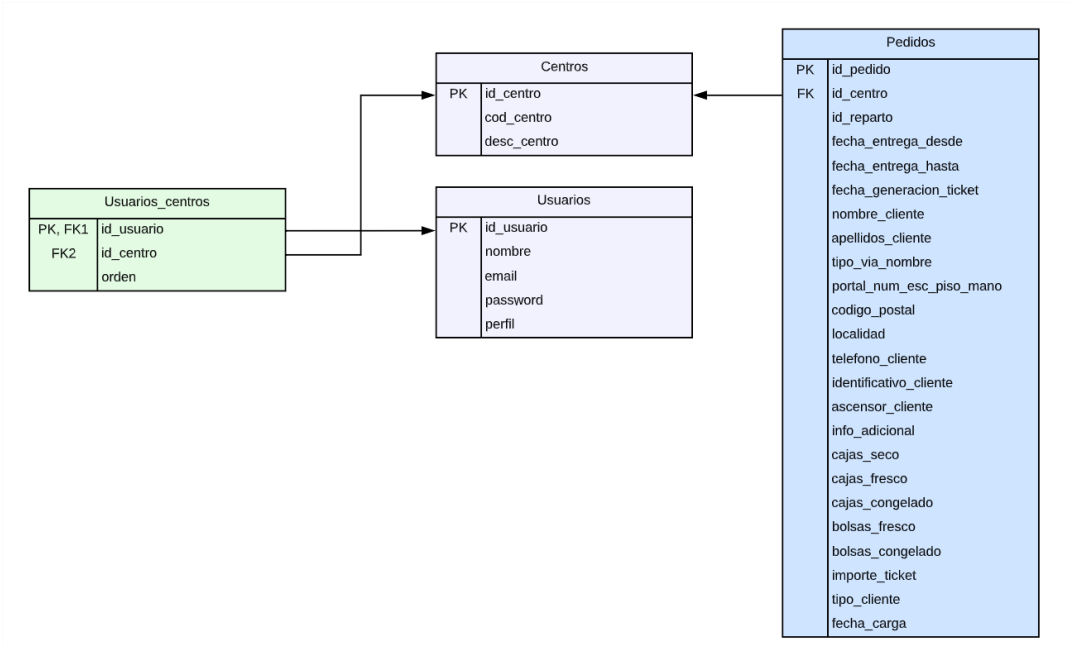


Figura 24 – Diagrama físico de base de datos

3.2.2 Diagrama de Clases

El diagrama de clases resultante, sería el siguiente:

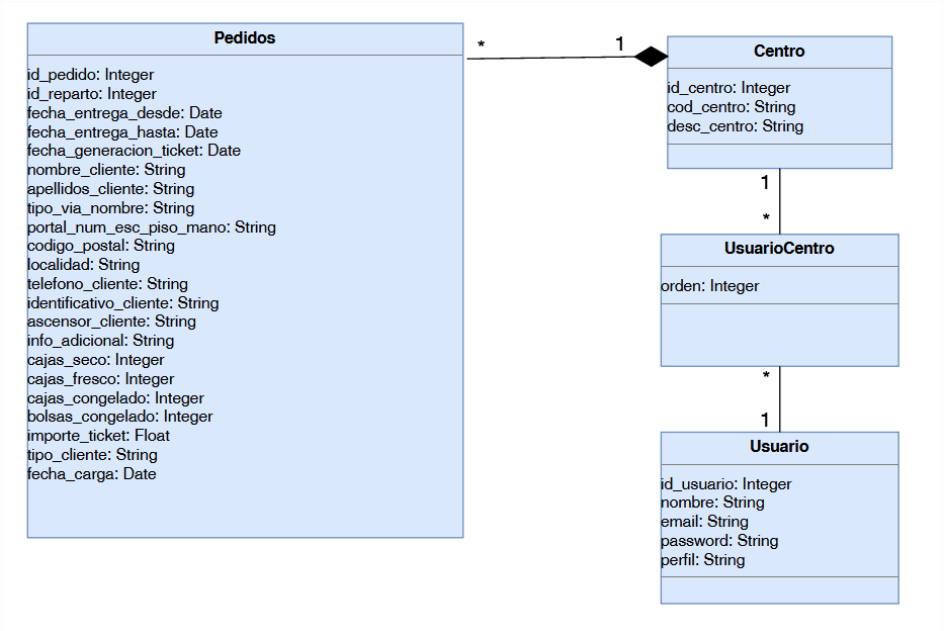


Figura 25 – Diagrama de clases

3.2.3 Diagrama de Arquitectura

La arquitectura del sistema se puede observar en el siguiente diagrama:

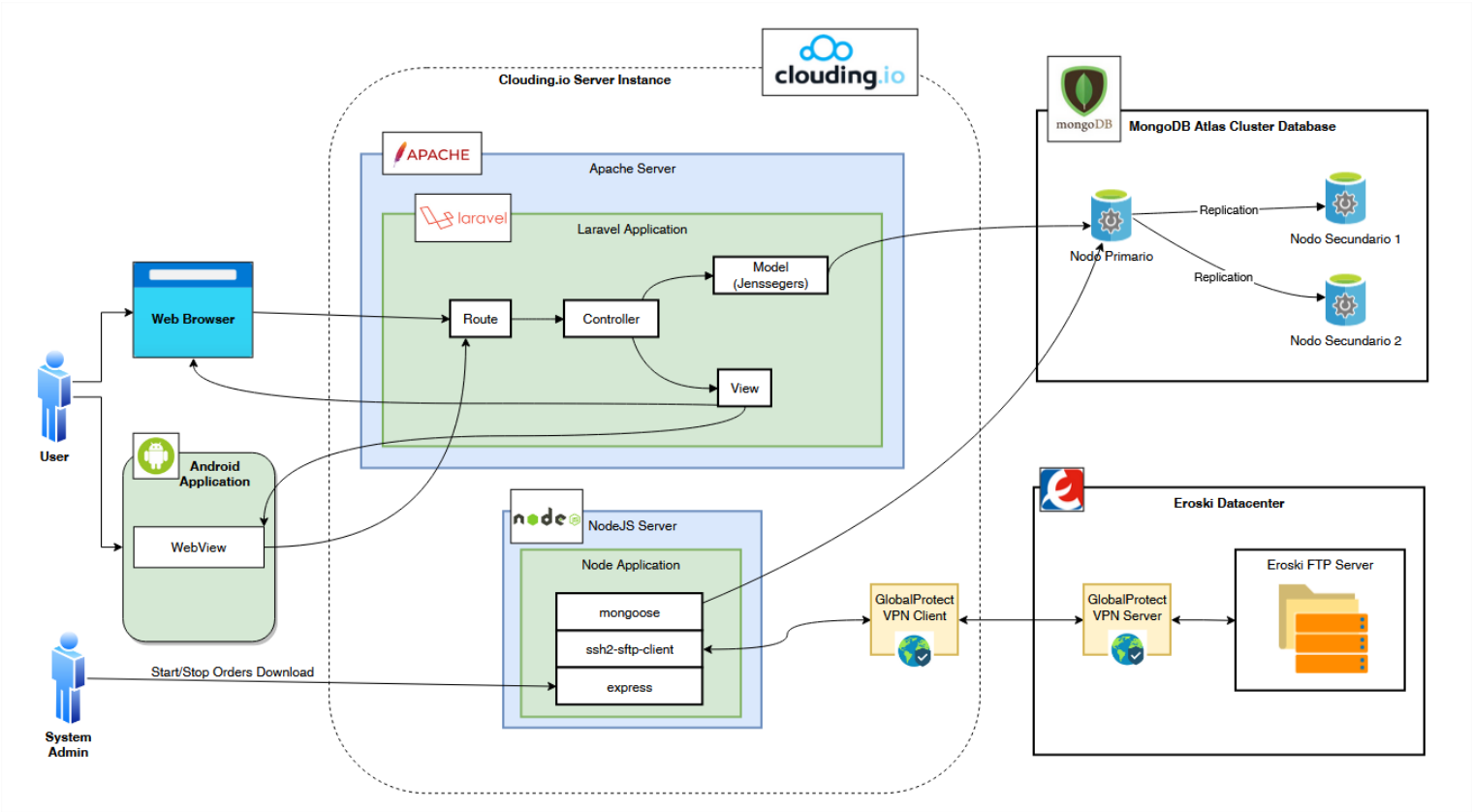


Figura 26 – Diagrama de arquitectura

4. Implementación

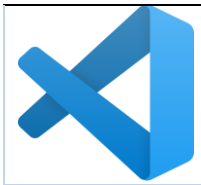
4.1 Introducción

Como se puede observar en el diagrama de arquitectura del apartado anterior (Figura 26), el sistema se compone de 3 aplicaciones diferentes (resaltadas en color verde):

- **Aplicación de descarga de pedidos**: Aplicación Node.js que está constantemente en funcionamiento revisando si entran pedidos nuevos en la carpeta FTP del servidor de Eroski.
- **Aplicación web de gestión de pedidos**: Aplicación PHP mediante framework Laravel que permite realizar la gestión de los pedidos, tareas de administración, etc.
- **Aplicación móvil de gestión de pedidos**: Aplicación Android que facilita la gestión de pedidos por repartidores.

4.2 Herramientas utilizadas

A lo largo del desarrollo de las diferentes aplicaciones, se han utilizado diversas herramientas, editores y tecnologías para llevar a cabo las diferentes tareas y componentes del proyecto. A continuación, se describen brevemente estas herramientas:

	**Visual Studio Code (VSCode)**: Se ha utilizado Visual Studio Code como editor de código para el desarrollo tanto de la aplicación Laravel como la nodeJS, debido a su versatilidad, soporte para múltiples lenguajes de programación, y su amplia gama de extensiones y complementos disponibles.
	**GitHub Desktop:** Para el control de versiones, se ha elegido GitHub Desktop por su integración con GitHub, así como por su interfaz gráfica intuitiva que facilita el seguimiento y la colaboración en el código fuente a lo largo del proyecto. Se han creado tres repositorios independientes, uno para cada aplicación.
	**Android Studio:** Se ha seleccionado Android Studio como plataforma de desarrollo para la aplicación móvil debido a que es el entorno de desarrollo oficial para aplicaciones Android, proporcionando herramientas específicas y optimizadas para la creación, depuración y distribución de aplicaciones en esta plataforma.
	**Node.js**: Para el desarrollo del proceso de descarga de pedidos, se eligió Node.js debido a su eficiencia, escalabilidad y capacidad para manejar tareas asíncronas, así como su eficiencia en el manejo de solicitudes de entrada/salida, lo que lo convierte en una buena opción para tareas que implican la transferencia de datos a través de la red. Se han utilizado varios paquetes para ampliar las funcionalidades y facilitar el desarrollo: • Express: Es un marco de aplicación web minimalista y flexible para Node.js que proporciona un conjunto sólido de características para crear aplicaciones web y API. En este proyecto, se utilizó Express para crear una API simple que permite gestionar la descarga de pedidos y su posterior carga en la base de datos MongoDB.

	• Mongoose: ofrece una solución para modelar los datos de MongoDB en aplicaciones basadas en Node.js. Mongoose proporciona una capa de abstracción para interactuar con la base de datos MongoDB, facilitando la realización de operaciones CRUD y garantizando la consistencia y validez de los datos. • ssh2-sftp-client: Este paquete se utilizó para establecer una conexión SFTP segura con el servidor de Eroski y así descargar los archivos de pedidos de manera segura. ssh2-sftp-client ofrece una API y promesas nativas para simplificar la implementación de las operaciones SFTP.
	**MongoDB Atlas:** Para hospedar y gestionar la base de datos MongoDB, se ha utilizado MongoDB Atlas, un servicio de base de datos en la nube que ofrece alta disponibilidad y posibilidad de escalabilidad bajo demanda.
	**MongoDB Compass**: Para la gestión de la base de datos MongoDB, se ha utilizado MongoDB Compass, una herramienta gráfica se integra perfectamente con MongoDB Atlas y que permite explorar, analizar y manipular los datos de MongoDB de manera sencilla, facilitando la administración de la base de datos.
	**XAMPP:** Se ha empleado XAMPP como servidor para la aplicación Laravel debido a su facilidad de instalación y configuración, así como por incluir todo lo necesario para ejecutar aplicaciones PHP, como el servidor web Apache y el gestor de bases de datos MariaDB, que en este proyecto no se ha utilizado por usar MongoDB para la persistencia de los datos.
	**Laravel**: Se ha seleccionado Laravel como framework de desarrollo web para la aplicación debido a su robustez, su sencillez de uso, su arquitectura MVC y su amplia comunidad de desarrolladores que ofrecen un gran soporte y recursos en línea.
	**Globalprotect:** El uso de este cliente VPN es un requisito impuesto por Eroski para asegurar la conexión segura y el acceso a su red corporativa para la conexión al FTP y la descarga de los ficheros de pedidos.

4.2 Análisis del estado actual del proyecto

Hasta la fecha actual (24 de mayo de 2023), el proyecto ha avanzado considerablemente, logrando cumplir la mayoría de los objetivos funcionales inicialmente propuestos, indicados en el apartado 1.2 del presente documento.

Entre los objetivos alcanzados se encuentran los siguientes:

- Implementación de la descarga y tratamiento automatizado de los pedidos, eliminando así los errores derivados del procesado manual.

- Desarrollo de una aplicación móvil que permite a los repartidores modificar el estado de los pedidos en tiempo real.

- Creación de un sistema de gestión que proporciona una visión actualizada del estado de los pedidos (entregados, pendientes, etc.)

- Establecimiento de un sistema que permite a cada repartidor conocer en tiempo real los pedidos que tiene que recoger y entregar.

- Integración de un sistema que permite obtener consultas y estadísticas de los pedidos realizados durante un periodo específico.

- Creación de los entornos de pruebas y producción y despliegue de las aplicaciones en dichos entornos.

Sin embargo, durante el desarrollo del proyecto se han encontrado algunas desviaciones respecto a la planificación inicial. Concretamente, no se ha podido alcanzar dos de los objetivos funcionales inicialmente propuestos, relativos a la a la creación de un sistema de alertas y a la elaboración de informes de control del desempeño de los repartidores.

Las desviaciones en la planificación se han producido debido a que determinadas funciones requirieron más tiempo del planificado inicialmente para su desarrollo. Esto impidió llegar a tiempo a la fecha límite para desarrollar dichas características, pero inicialmente ya se puso de manifiesto que estos desarrollos eran los que menos prioridad tenían en el proyecto.

Para corregir estas desviaciones y alcanzar los objetivos pendientes, se propone desarrollar las mismas en la fase de mantenimiento del proyecto como desarrollos evolutivos que requerirán una nueva planificación. Para ello se mantendrá la comunicación entre las partes interesadas para asegurar que se entienden y se cumplen los requisitos de las nuevas funcionalidades, y para gestionar eficazmente estos desarrollos y cuales quiera otros que aparezcan en el futuro.

5. Pruebas

Para garantizar el correcto funcionamiento y la calidad de la aplicación, se ha seguido una estrategia de pruebas abarca pruebas unitarias y pruebas de integración. No se han realizado pruebas de carga ya que el nivel tan bajo de número de usuarios, en torno a unos 10, no supone un problema de rendimiento y no se prevé que vaya a crecer a medio plazo. Además, el hecho de funcionar en entornos cloud, hace que se puedan asignar recursos en un momento dado si es necesario. En cuanto a la interfaz de usuario de la aplicación móvil, se consultó con los usuarios repartidores formas de mejorar en cuanto a la disposición de los campos y los botones de modificación de estado.

Debido a que prácticamente toda la lógica de negocio reside en la aplicación PHP desarrollada en Laravel, las pruebas unitarias se han desarrollado en dicha aplicación utilizando el framework PHPUnit, que se incluye por defecto con Laravel. Estas pruebas están diseñadas para validar el correcto funcionamiento de cada uno de los componentes individuales de la aplicación, comprobando que cada método y función realiza su trabajo como se espera.

Posteriormente se han realizado pruebas de integración. Estas pruebas garantizan que los componentes del sistema (la aplicación Node.js, la aplicación Laravel, la aplicación Android y la base de datos MongoDB) interactúan correctamente entre sí. Este proceso de prueba abarca desde la descarga y procesamiento de archivos desde el servidor FTP utilizando el cliente VPN, hasta la visualización de los datos de los pedidos en la aplicación Android.

La aplicación móvil, que es esencialmente un Webview que muestra la interfaz de usuario proporcionada por Laravel, se ha probado en un dispositivo móvil real. Decidimos realizar pruebas en un dispositivo en lugar de un emulador debido a la velocidad de respuesta más realista y la mejor representación del rendimiento de la aplicación en condiciones reales. Estas pruebas se centraron en la experiencia del usuario, verificando que la aplicación fuera fácil de usar y que cumpliera con las necesidades y expectativas de los usuarios en cuanto a la funcionalidad.

5.1 Pruebas Unitarias

Las pruebas unitarias desarrolladas en la aplicación PHP han sido las siguientes:

5.1.1 Prueba testIndex de la Clase CentroControllerTest

Esta prueba unitaria prueba el método *index* de la clase *CentroController* (Controlador de centros o tiendas) que devuelve una lista de las tiendas existentes. La prueba unitaria comprueba que se devuelve una vista con el nombre ‘centros’ y que devuelve una lista de centros no nula (> 0) con datos. La prueba asume que ya existen datos en la base de datos antes de que se ejecute la prueba.

```php
use App\Http\Controllers\CentroController;
use App\Models\Centro;
use Illuminate\Http\Request;
use Tests\TestCase;

//Clase de prueba unitaria del controlador CentroController
class CentroControllerTest extends TestCase
{

    public function testIndex()
    {
        $controller = new CentroController();

        // Llamamos al método index del controlador de centros
        $response = $controller->index();

        // Realizamos las comprobaciones del test:
        $this->assertEquals('centros', $response->getName());   //Comprobamos que se devuelve una vista con nombre 'centros'
        $this->assertArrayHasKey('centros', $response->getData());       //Comprobamos que la vista tiene una variable centros disponible
        $datos = $response->getData()['centros'];
        $this->assertGreaterThan(0, $datos->count()); //Comprobamos que se devuelven datos de centros
    }
}
```

Figura 27 – Test unitario testIndex de CentroControllerTest

5.1.2 Prueba testOrdersFecha de la Clase OrderControllerTest

El método *ordersFecha* de la clase controlador *OrderController* devuelve una lista de pedidos entre dos fechas concretas que se le pasan en la petición (request). El test comprueba que para una fecha concreta, se devuelve una vista de nombre pedidos a la que se le pasa un array de nombre 'orders', que la fecha que se le pasa a la vista es la correcta, y que todos los pedidos que figuran en el array son de la fecha correcta.

```php
public function testOrdersFecha()
{
    $request = new Request();
    $request->setMethod('POST');
    $request->request->add(['fecha' => '2023-05-23']);

    $controller = new OrderController();

    // Acción
    $response = $controller->ordersFecha($request);

    // Comprobaciones
    $this->assertEquals('pedidos', $response->getName());   //Comprobamos que devuelve una vista de nombre 'pedidos'
    $this->assertArrayHasKey('orders', $response->getData());  //Comprobamos que se devuelve un array de orders
    $this->assertArrayHasKey('fecha', $response->getData());  //Comprobamos que se devuelve la fecha que se consulta

    $datos = $response->getData()['orders'];
    $this->assertGreaterThan(0, $datos->count());   //Comprobamos que se devuelven pedidos

    foreach($datos as $order){
        //Comprobamos que para cada pedido obtenido, la fecha sea la correcta
        $this->assertEquals($response->getData()['fecha'], $order->fecha_entrega_desde->toDateTime()->setTimezone(new \DateTimeZone('Europe/Madrid'))->format('Y-m-d'));

    }

}
```

Figura 28 – Test unitario testOrdersFecha de OrderControllerTest

5.1.3 Prueba testIndex de la Clase OrderControllerTest

El método *testIndex* de la clase controlador *OrderController* realiza una función similar al de testOrdersFecha, pero devuelve los pedidos del dia actual. En el test se comprueban los mismos aspectos que en el test de *testOrdersFecha,* es decir, la correctitud de la vista y del array de pedidos, y que las fechas son las correctas.

```php
public function testIndex()
{
    $request = new Request();
    $request->setMethod('POST');
    $request->request->add(['fec' => '2023-05-23']);

    $controller = new OrderController();

    // Crear un usuario ficticio.
    $user = User::factory()->create();

    // Simular que el usuario está autenticado.
    $this->actingAs($user);

    // Acción
    $response = $controller->index($request);

    // Comprobaciones
    $this->assertEquals('pedidos', $response->getName());   //Comprobamos que devuelve una vista de nombre 'pedidos'
    $this->assertArrayHasKey('orders', $response->getData());  //Comprobamos que se devuelve un array de orders
    $this->assertArrayHasKey('fecha', $response->getData());  //Comprobamos que se devuelve la fecha que se consulta

    $datos = $response->getData()['orders'];
    $this->assertGreaterThan(0, $datos->count());   //Comprobamos que se devuelven pedidos

    foreach($datos as $order){
        //$this->assertEquals($response->getData()['fecha'], $order->fecha_entrega_desde );
        $this->assertEquals($response->getData()['fecha'], $order->fecha_entrega_desde->toDateTime()->setTimezone(new \DateTimeZone('Europe/Madrid'))->format('Y-m-d'));

    }
}
```

Figura 29 – Test unitario testIndex de OrderControllerTest

5.1.4 Prueba testActualizaEstado de la Clase OrderControllerTest

Esta prueba unitaria comprueba el método que cambia de estado de un pedido. Para ello realiza lo siguiente: Primero crea un pedido ficiticio con estado = 0 (pendiente). Posteriormente autentifica a un usuario también ficticio creado al efecto. Posteriormente envía una solicitud de tipo POST al método *actualizaEstado* del controlador *OrderController* pasando los datos del pedido ficticio. Una vez hecho esto se comprueba que el pedido ha pasado correctamente del estado “Pendiente” al estado “Entregado”. Posteriormente se vuelve a hacer lo mismo pero a la inversa, es decir, se pasa el pedido de Entregado a Pendiente y se vuelve a comprobar que la operación ha sido correcta. Tambien se comprueban en ambos casos que se rellenan de forma correcta los campos del usuario que ha modificado el pedido y la fecha de modificación.

```php
public function testActualizarEstado()
{

    //Eliminamos todos los documentos que tengan un campo fecha = '01/01/2000' (El campo fecha no se utiliza en la aplicacion. Las fechas tienen otros nombr
    $order=Order::where('fecha','01/01/2000')->delete();

    // Crear un usuario ficticio.
    $user = User::factory()->create();
    $user->email='jlrubiol@gmail.com';
    $this->actingAs($user);

    // Crear un pedido ficticio.
    $order = new Order([
        'centro'=>'4041',
        'fecha' =>'01/01/2000',
        'estado' => 0 // Estado inicial.
    ]);

    $order->save(); //Guarda los datos para el usuario

    // Simular que el usuario está autenticado.
    $this->actingAs($user);
    $fecha_actu=Carbon::now();

    // Enviar una solicitud POST a la función incluyendo los datos necesarios que pase de pendiente a entregado.
    $request = new Request();
    $request->setMethod('POST');
    $request->request->add(['id'=> $order->_id, 'centro' => $order->centro, 'fecha='> $fecha_actu,'estado' =>$order->estado]);

    $controller = new OrderController();
    $response=$controller->actualizarEstado($request);

    // Comprobar que el estado del pedido se ha actualizado correctamente de pendiente a entregado.
    $order->refresh(); //Refrescamos los datos del pedido
    $this->assertEquals(1, $order->estado);
    //Comprobamos que se ha actualizado correctamente el campo de usuario modificador
    $this->assertEquals($user->email, $order->usuario);
    //Comrprobamos que se ha actualizado la fecha de modificación.
    $this->assertEquals($fecha_actu->format('Y-m-d'), $order->updated_at->toDateTime()->setTimezone(new \DateTimeZone('Europe/Madrid'))->format('Y-m-d'));


    // Enviar una solicitud POST a la función incluyendo los datos necesarios que pase de entregado a pendiente.
    $request = new Request();
    $request->setMethod('POST');
    $request->request->add(['id'=> $order->_id, 'centro' => $order->centro, 'fecha='> $fecha_actu,'estado' =>$order->estado]);

    $controller = new OrderController();
    $response=$controller->actualizarEstado($request);

        // Comprobar que el estado del pedido se ha actualizado correctamente de pendiente a entregado.
        $order->refresh(); //Refrescamos los datos del pedido
        $this->assertEquals(0, $order->estado);
        //Comprobamos que se ha actualizado correctamente el campo de usuario modificador
        $this->assertEquals($user->email, $order->usuario);
        //Comrprobamos que se ha actualizado la fecha de modificación.
        $this->assertEquals($fecha_actu->format('Y-m-d'), $order->updated_at->toDateTime()->setTimezone(new \DateTimeZone('Europe/Madrid'))->format('Y-m-d'));

        //Eliminamos el pedido de prueba para sucesivos tests
        $order=Order::destroy($order->id);
    }


}
```

Figura 30 – Test unitario testActualizaEstado de OrderControllerTest

5.1.5 Prueba testIndexconAutenticacion de la Clase LoginControllerTest

Esta prueba unitaria comprueba el método *index* de la clase *LoginController*, que comprueba que si el usuario está autentificado y tiene una sesión activa, le redirige a la página de pedidos (/orders). En caso de que no esté autenticado, se redirige al usuario a la página de login. Para ello, se crea un usuario ficticio y se simula su autenticación mediante el método *actingAs.* Posteriormente se realiza la llamada a la ruta /login (que está enlazada con el método *index* del *LoginController*) y se comprueba que se ha realizado la redirección esperada a la ruta /orders

```php
public function testIndexConAutenticacion()
{

    // Creamos un usuario usando la factoría de usuarios
    $user = User::factory()->create();
    $user->email='jlrubiol@gmail.com';

    // Simulamos que el usuario está autenticado
    $this->actingAs($user);

    // Visita la ruta asociada al método 'index' de la clase
    $response = $this->get('/login');

    // Comprobamos  que se redirige a la ruta esperada, en este caso /orders
    $response->assertRedirect('/orders');
}
```

Figura 31 – Test unitario testIndexConAutenticacion de LoginController

5.1.6 Prueba testIndexSinAutenticacion de la Clase LoginControllerTest

Esta prueba unitaria verifica el otro caso de *testIndex*, es decir, el caso en el que usuario no esta autenticado en la aplicación y debe ser redirigido a la página de login. Para ello, de la misma manera que con el método anterior, se crea un usuario ficticio, pero en este caso no se autentica. Posteriormente se redirige a la ruta /login y se comprueba que la redirecciona a la vista de login.

```php
public function testIndexSinAutenticacion()
{
    // Creamos un usuario usando la factoría de usuarios
    $user = User::factory()->create();

    // Visita la ruta asociada al método 'index' de la clase
    $response = $this->get('/login');

    // Comprobamos que la respuesta es correcta
    $response->assertStatus(200);

    // Comprobamos  que se redirige a la vista de login
    $response->assertViewIs('login');

}
```

Figura 32 – Test unitario testIndexSinAutenticacion de LoginController

5.1.7 Prueba testLoginOK de la Clase LoginControllerTest

Esta prueba unitaria comprueba el funcionamiento del método *validarLogin* de la clase *LoginController*, para el caso de que se produzca una validación correcta de un usuario.
Para ello, primero se crea un usuario ficticio con una contraseña determinada (en nuestro caso 'pruebasuoc'). Posteriormente se crea la petición incluyendo los datos de usuario y contraseña, se genera una sesión ficticia y se llama al método *validarLogin* de la clase *LoginController.* Tras esto se comprueba que el usuario se ha autenticado correctamente.

```php
public function testLoginOK()
{
    // Creamos un usuario usando la factoría de usuarios con la contraseña pruebasuoc
    $user = User::factory()->create(['password' => bcrypt($password = 'pruebasuoc')]);

    // Enviar una solicitud POST a la función incluyendo los datos necesarios que pase de pendiente a entregado.
    $request = new Request();
    $request->setMethod('POST');
    $request->request->add([
        'email' => $user->email,
        'password' => 'pruebasuoc',
    ]);

    // Simulamos una sesión
    $request->setLaravelSession($this->app['session.store']);

    //Realizamos la llamada al método del controlador
    $controller = new LoginController();
    $response=$controller->validarLogin($request);

    // Comprobamos que el usuario autenticado es el que ha de ser
    $this->assertAuthenticatedAs($user);

    //Borramos los usuarios creados
    User::where('email', 'like', '%example%')->delete();
}
```

Figura 33 – Test unitario testLoginOk de LoginControllerTest

5.1.8 Prueba testLoginKO de la Clase LoginControllerTest

En este caso, este test unitario comprueba el caso contrario del método *testLoginOK*, es decir, cuando el usuario que se valida no está registrado en el sistema. Para ello realiza los mismos pasos que en el método anterior, es decir, creación de usuario ficticio con una contraseña, creación de la petición al método *validarLogin* y finalmente comprobación de que el usuario no se ha autenticado.

```php
public function testLoginKO()
{
    // Creamos un usuario usando la factoría de usuarios con la contraseña pruebasuoc
    $user = User::factory()->create(['password' => bcrypt($password = 'pruebasuoc')]);

    // Enviar una solicitud POST a la función incluyendo los datos necesarios que pase de pendiente a entregado.
    $request = new Request();
    $request->setMethod('POST');
    $request->request->add([
        'email' => $user->email,
        'password' => 'passwordErronea',
    ]);

    // Simulamos una sesión
    $request->setLaravelSession($this->app['session.store']);

    //Realizamos la llamada al método del controlador
    $controller = new LoginController();
    $response=$controller->validarLogin($request);

    // Comprobamos que el usuario no fue autenticado
    $this->assertGuest();

    //Limpiamos los usuarios creados
    User::where('email', 'like', '%example%')->delete();
}
```

Figura 34 – Test unitario testLoginKO de LoginControllerTest

5.1.9 Comprobación de test unitarios

Mediante el comando .\vendor\bin\phpunit ejecutamos todos los tests para comprobar que todos pasan correctamente. Podemos ver el resultado en la siguiente imagen:

```
C:\xampp\htdocs\blaniza>.\vendor\bin\phpunit --debug
PHPUnit 9.5.27 by Sebastian Bergmann and contributors.

Test 'Tests\Unit\CentroControllerTest::testIndex' started
Test 'Tests\Unit\CentroControllerTest::testIndex' ended
Test 'Tests\Unit\LoginControllerTest::testIndexConAutenticacion' started
Test 'Tests\Unit\LoginControllerTest::testIndexConAutenticacion' ended
Test 'Tests\Unit\LoginControllerTest::testIndexSinAutenticacion' started
Test 'Tests\Unit\LoginControllerTest::testIndexSinAutenticacion' ended
Test 'Tests\Unit\LoginControllerTest::testLoginOK' started
Test 'Tests\Unit\LoginControllerTest::testLoginOK' ended
Test 'Tests\Unit\LoginControllerTest::testLoginKO' started
Test 'Tests\Unit\LoginControllerTest::testLoginKO' ended
Test 'Tests\Unit\OrderControllerTest::testOrdersFecha' started
Test 'Tests\Unit\OrderControllerTest::testOrdersFecha' ended
Test 'Tests\Unit\OrderControllerTest::testIndex' started
Test 'Tests\Unit\OrderControllerTest::testIndex' ended
Test 'Tests\Unit\OrderControllerTest::testActualizarEstado' started
Test 'Tests\Unit\OrderControllerTest::testActualizarEstado' ended
Test 'Tests\Feature\ExampleTest::test_the_application_returns_a_successful_response' started
Test 'Tests\Feature\ExampleTest::test_the_application_returns_a_successful_response' ended


Time: 00:06.078, Memory: 30.00 MB

OK (9 tests, 234 assertions)

C:\xampp\htdocs\blaniza>
```

Figura 35 - Salida del comando phpunit

5.2 Pruebas de integración

Las pruebas de integración se han desarrollado para verificar la correcta interacción entre las diferentes aplicaciones (Node.js, Laravel y Android) y la base de datos MongoDB.

Conexión a FTP y descarga de archivos: Se ha verificado que la aplicación Node.js puede conectarse al servidor FTP y descargar los archivos de pedidos correctamente. También se han probado escenarios de error, como la caída del servidor FTP y la presencia de archivos en formatos no esperados y se ha comprobado que la aplicación Node.js dispone de los mecanismos para detectar dichos errores y mostrarlos al usuario, además de reflejarlos en los logs de la aplicación. En un futuro se espera que puedan realizarse avisos y alertas que informen de una caída del servidor FTP de Eroski. Actualmente hay que comprobar periódicamente que los ficheros llegan de forma correcta, lo que casi siempre es el caso.

Procesamiento de archivos y carga en MongoDB: Se han desarrollado pruebas para asegurar que la aplicación Node.js puede leer los archivos descargados, extraer los datos de los pedidos y cargarlos correctamente en la base de datos MongoDB. Se han incluido escenarios de error, como la presencia de archivos corruptos o datos de pedidos que no cumplen con el formato esperado. En ese caso, la aplicación se comporta rechazando dichos pedidos y mostrando el error por consola y en los logs. También se ha comprobado que todos los datos de los ficheros se traspasan de forma correcta a los documentos MongoDB.

Consulta de datos desde Laravel: Se han realizado pruebas para verificar que la aplicación Laravel puede consultar los datos de los pedidos de MongoDB y mostrarlos correctamente. Esto incluye la visualización de datos individuales de pedidos y la generación de resúmenes y estadísticas.

Interacción entre Laravel y la aplicación Android: Se ha probado que la aplicación Android puede mostrar correctamente la interfaz de usuario proporcionada por Laravel y que los usuarios pueden interactuar con ella como se espera. Esto incluye cambiar el estado de un pedido y la disposición de los campos de la forma más cómoda para los repartidores.

Escenarios de prueba de extremo a extremo: Además de las pruebas individuales, se han realizado pruebas de extremo a extremo que cubren todo el proceso, desde la descarga de un archivo de pedido del FTP hasta su visualización en la aplicación Android. Estas pruebas ayudan a garantizar que todas las partes del sistema funcionan correctamente en conjunto.

6. Conclusiones

A lo largo del desarrollo de este proyecto, se han utilizado diversas tecnologías para el desarrollo de los diferentes componentes del sistema. La mayoría de ellas han requerido un esfuerzo de aprendizaje e investigación ya que eran desconocidas de inicio, por lo que se han adquirido conocimientos y experiencias que creo que pueden ser útiles en proyectos futuros. El uso de Node.js me ha permitido conocer un uso para la tecnología diferente del clásico de creación de APIs, como es el de un proceso servidor que está monitorizando continuamente el estado de una carpeta al estilo de un Daemon de un sistema operativo. Trabajar con MongoDB me ha permitido el conocer cómo funciona un sistema NoSQL y como puede ser utilizado de manera eficiente en ciertos casos de uso. Además, este proyecto ha permitido la oportunidad de aplicar y expandir el conocimiento obtenido durante la carrera sobre tecnologías como Laravel en un contexto de mundo real. También considero positivo la experiencia de desplegar las aplicaciones en un servidor IASS en la nube, lo que me ha permitido conocer en la práctica aspectos específicos de la tecnología como el dimensionamiento de los recursos, el pago por uso, etc.

Además de las habilidades técnicas adquiridas, este proyecto ha destacado la importancia del diseño centrado en el usuario (DCU). Mediante los métodos de indagación, como el shadowing y las entrevistas y a través de la creación de los perfiles de usuario y la implementación de los mismos en el desarrollo de la aplicación mediante los escenarios de uso, se ha podido observar de primera mano cómo un enfoque basado en las necesidades y experiencias de los usuarios puede conducir a una aplicación más útil y efectiva.

Por otro lado, se ha hecho evidente la necesidad de una buena gestión de proyectos para poder alcanzar las fechas de plazo requeridas y de la realización de pruebas unitarias y de integración para garantizar la calidad del software desarrollado. El seguimiento de la planificación del proyecto ha sido un desafío. Aunque se estableció un cronograma inicial, hubo desviaciones debido a factores imprevistos. La mayoría de los objetivos planteados inicialmente fueron logrados con éxito. Sin embargo, no se lograron completamente los objetivos relacionados con la creación de un sistema de alertas y el seguimiento del desempeño de los repartidores. Dichos objetivos no se lograron debido a la complejidad inherente a estas funcionalidades que requerían un esfuerzo adicional de aprendizaje que no era compatible con los plazos de entrega. Estas funcionalidades tienen previsto su desarrollo en el futuro.

En términos de metodología, se ha elegido trabajar con una metodología ágil basada en Kanban que ha demostrado ser efectiva, permitiendo la adaptación a los cambios y la entrega continua e incremental de funcionalidades nuevas en el sistema.

En el futuro, sería útil completar las funcionalidades ya comentadas que han quedado pendientes como el sistema de alertas y el seguimiento del desempeño de los repartidores. Además, sería interesante explorar la posibilidad de implementar más características como la geolocalización de los repartidores o la optimización de la planificación de las entregas en base a las rutas. También podría ser útil el desarrollo de una aplicación móvil más robusta que pueda funcionar independientemente de la aplicación web, ofreciendo más flexibilidad y funcionalidad a los repartidores.

7. Glosario

Término	Definición
FTP	FTP (File Transfer Protocol) es un protocolo de red que se utiliza para la transferencia de archivos entre un cliente y un servidor en una red de ordenadores
SFTP	SFTP (SSH File Transfer Protocol) es un protocolo de red que proporciona funcionalidades para la transferencia de archivos y la manipulación de archivos y directorios remotos. SFTP proporciona la seguridad y la integridad de datos utilizando SSH (Secure Shell).
VPN	VPN (Virtual Private Network) es una tecnología que crea una red segura y encriptada sobre una red menos segura, como internet. Permite que los dispositivos envíen y reciban datos a través de redes compartidas como si estuvieran directamente conectados a una red privada.
REST	REST (Representational State Transfer) es un estilo de arquitectura de software para sistemas distribuidos. Es utilizado en el desarrollo de servicios web debido a su simplicidad y la facilidad para integrarse con sistemas existentes.
NoSQL	NoSQL es un tipo de sistemas de gestión de bases de datos que no utiliza el modelo relacional tradicional. NoSQL es útil cuando se trabaja con grandes conjuntos de datos distribuidos, ya que permite una mayor flexibilidad y escalabilidad que las bases de datos SQL tradicionales.
HTML5	HTML5 es la quinta versión del lenguaje de marcado HTML. HTML5 introduce nuevas etiquetas y atributos que reflejan el uso moderno de la web, como la incorporación de multimedia y aplicaciones gráficas.
PHP	PHP (Hypertext Preprocessor) es un lenguaje de programación de código abierto que se utiliza principalmente en el desarrollo web. Es especialmente adecuado para el desarrollo de sitios web dinámicos y aplicaciones web.
Node.js	Node.js es un entorno de ejecución de JavaScript basado en el motor de JavaScript V8 de Chrome. Permite ejecutar JavaScript en el servidor, lo que lo hace muy útil para el desarrollo de aplicaciones web en tiempo real.
MongoDB	MongoDB es una base de datos NoSQL orientada a documentos. Almacena los datos en documentos tipo JSON con un esquema flexible, lo que permite la integración de datos de diferentes tipos y estructuras.
CRUD	CRUD (Create, Read, Update, Delete) e un acrónimo que representa las cuatro operaciones básicas que se pueden realizar en cualquier base de datos: Crear, Leer, Actualizar y Borrar.
API	API (Application Programming Interface) es un conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre sí. Sirve como interfaz entre diferentes programas de software para facilitar su interacción.
MongoDB Atlas	Es la versión en la nube de MongoDB, proporcionada como un servicio. Permite desplegar, operar y escalar bases de datos MongoDB en la nube
Webview	Es un componente utilizado en el desarrollo de aplicaciones móviles para mostrar contenido web.
UML	UML (Unified Modeling Language) es un lenguaje de modelado de sistemas que se utiliza en el desarrollo de software. Proporciona un conjunto estándar de diagramas y notaciones para describir la arquitectura y los procesos de un sistema de software. Los diagramas UML pueden representar componentes de software, la interacción entre diferentes componentes, flujos de trabajo, entre otros.

8. Bibliografía

[1] Simple Gantt Chart
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html
Vertex42.com – Visitado el 13/03/2023

[2] Diseño Centrado en el Usuario para dispositivos móviles
https://xwiki.recursos.uoc.edu/wiki/matm1202es/view/Main/
Jordi Almirall López – XWIKI Universitat Oberta de Catalunya – Visitado el 25/03/2023

[3] Shadowing
https://idital.com/diccionario-seo/shadowing/
idital.com (Idiwiki) Visitado el 04/04/2023

[4] "Personas": El Diseño de Interacción según Alan Cooper
https://blog.ida.cl/diseno/personas-el-diseno-de-interaccion-segun-alan-cooper/
Rodrigo Vera – {ida Blog – Visitado el 04/04/2023

[5] De personas a Personas
https://aprendizajeenredeafit.wordpress.com/2016/04/15/de-personas-a-personas/
Victor García - Visitado el 04/04/2023

[6] Tipos de relaciones en diagramas de casos de uso. UML.
https://www.seas.es/blog/informatica/tipos-de-relaciones-en-diagramas-de-casos-de-uso-uml/
José Maria Megino Barquinero – Blog SEAS – Visitado el 10/04/2023

[7] Node.js: The Complete Guide to Build RESTful APIs (2018)
https://www.udemy.com/course/nodejs-master-class/
Mosh Hamedani - Udemy.com - Visitado recurrentemente desde el 07/01/2023

[8] How to connect to SFTP in Node.js
https://sftptogo.com/blog/node-sftp/
Moty Michaely – SFTP To Go blog - Visitado el 08/01/2023

[9] Connect to a MongoDB Database Using Node.js
https://www.mongodb.com/developer/languages/javascript/node-connect-mongodb/
MongoDB Developer Documentation - Visitado el 7 marzo 23

[10] Laravel Documentation
https://laravel.com/docs/10.x/readme
laravel.com developer documentation – Visitado recurrentemente desde el 08/01/2023

[11] Responsive Table V1
https://colorlib.com/wp/template/responsive-table-v1/
colorlib.com – Visitado el 07/04/2023

[12] How do you make an accordion in a table for a responsive site
https://www.justinmind.com/community/topic/how-do-you-make-an-accordion-in-a-table-for-a-responsive-site
justinmind.com – Visitado el 08/04/2023

[13] MongoDB and Laravel Integration - Connect MongoDB to Laravel in minutes with Atlas.
https://www.mongodb.com/compatibility/mongodb-laravel-intergration
MongoDB.com - Visitado el 01/05/2023

[14] How to Export HTML Table to Excel Using JavaScript
https://linuxhint.com/export-html-table-to-excel-using-javascript/
Farah Batool – linuxhint.com – Visitado el 13/05/2023

[15] Create a WebView in Android Studio: ProgressBar + Swipe to Refresh + Custom Error Page on Network Failure
https://codeflarelimited.com/blog/create-a-webview-in-android-studio-progressbar-swipe-to-refresh-custom-error-page-on-network-failure/
codeflare.com - Visitado el 18/04/2023

[16] How to display progress bar while loading a url to webView in Android?
https://www.tutorialspoint.com/how-to-display-progress-bar-while-loading-a-url-to-webview-in-android
Azhar – Tutorialspoint.com – Visitado el 18/04/2023

[17] Parte 6: Test unitarios (unit test) en nuestro blog con LARAVEL 8
https://cosasdedevs.com/posts/test-unitarios-unit-test-en-nuestro-blog-con-laravel-8/
Alber – Blog cosasdedevs.com – Visitado el 22/05/2023

9. Anexos

Anexo I - Manual Aplicación de descarga de pedidos

El presente manual proporciona instrucciones para utilizar la aplicación de descarga de pedidos de los servidores de Eroski. Esta aplicación, desarrollada en Node.js, descarga y procesa los pedidos pendientes de los servidores de Eroski, y los almacena en una base de datos MongoDB desplegada en un servidor en la nube.

1.- Conexión VPN

La aplicación necesita una conexión VPN para comunicarse con el servidor de Eroski de manera segura. Es importante asegurarse de que la conexión VPN está correctamente configurada y funcionando antes de iniciar la aplicación. Actualmente, Eroski recomienda el uso del cliente VPN GlobalPortect. Los detalles específicos sobre cómo configurar la conexión VPN los proporciona el departamento técnico de Eroski.

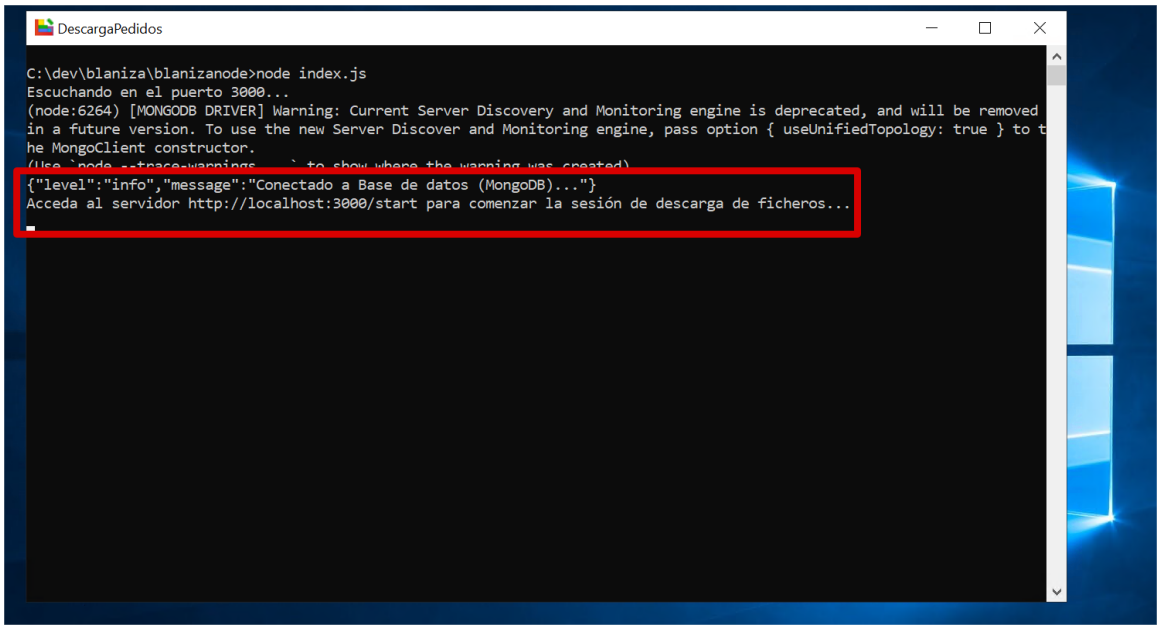
2.- Instalación y ejecución de la aplicación

Los pasos para poder ejecutar la aplicación son los siguientes:

1. Instalación de Node.js y NPM: Es necesario tener instalado Node.js y el administrador de paquetes NPM. Si no están instalados, se pueden descargar desde el sitio web oficial de Node.js (https://nodejs.org/es/download). Con la instalación de Node.js viene ya incluido el gestor de paquetes NPM, por lo que no es necesario instalarlo por separado.
2. Descomprimir los ficheros de la aplicación blanizanode en una carpeta
3. Mediante una ventana de intérprete de comandos, acceder a la carpeta de la aplicación
4. Como ya se han incluido en el paquete todas las dependencias no es necesario ejecutar npm install para descargarlas.
5. Ejecutar la aplicación escribiendo en la ventana de intérprete de comandos: node index.js
6. También es posible ejecutar la aplicación haciendo doble click en el fichero DescargaPedidos.bat incluido en los archivos de la aplicación.

3.- Funcionamiento de la aplicación

Al ejecutar la aplicación, esta iniciará el servidor express con el que funciona Node. Dicho servidor permanece a la escucha en el puerto 3000. Veremos los siguientes mensajes en la ventana de intérprete de comandos:



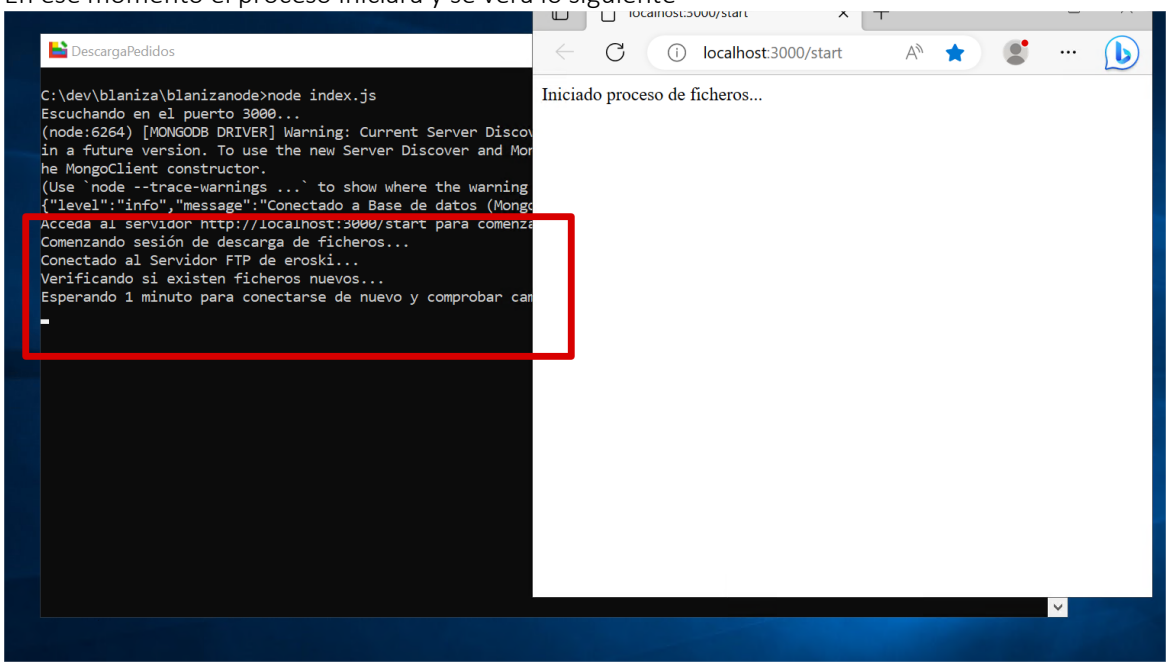
La aplicación en ese momento está lista para iniciar la sesión de descarga de pedidos.

4.- Inicio de la sesión de descarga de archivos

Para comenzar la sesión de descarga de archivos, use la ruta **/start**. Esta acción establecerá un intervalo (por defecto cada 60 segundos) durante el cual la aplicación comprobará si existen nuevos archivos en el servidor de Eroski, los descargará, procesará y almacenará los datos de los pedidos en la base de datos.

Por ejemplo, se puede comenzar la sesión de descarga de archivos visitando la dirección http://localhost:3000/start en su navegador web, donde "localhost" es la dirección de su servidor y "3000" es el puerto en el que se está ejecutando la aplicación.

En ese momento el proceso iniciará y se verá lo siguiente



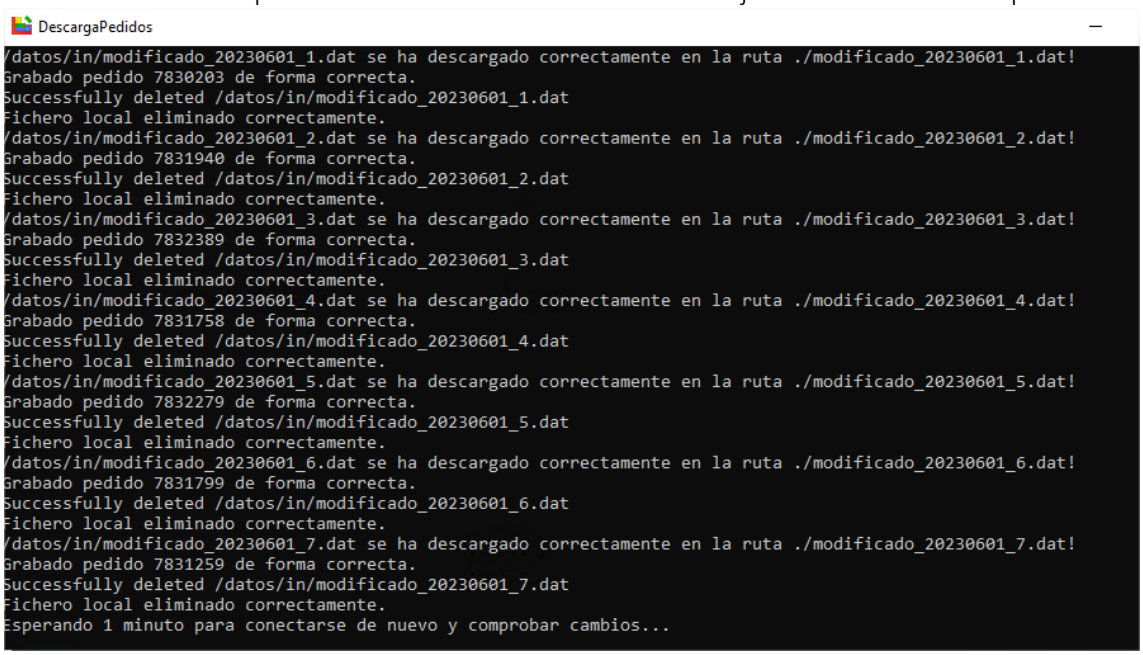
La aplicación utiliza dos carpetas en el servidor FTP de Eroski: /datos/in y /datos/out.

- **Carpeta /in:** Esta es la carpeta donde la aplicación busca nuevos archivos de pedido para procesar. Cuando hay un nuevo pedido, Eroski coloca un archivo correspondiente en esta

carpeta. La aplicación descarga estos archivos, los procesa y luego los elimina de la carpeta /in.

- Carpeta /out: Después de que la aplicación ha procesado un archivo de pedido y lo ha cargado en la base de datos, copia el archivo a la carpeta /out en el servidor FTP. Esto sirve como un registro de los archivos que han sido procesados. Si hay un problema con la aplicación, estos archivos pueden ser útiles para depuración y recuperación de datos.

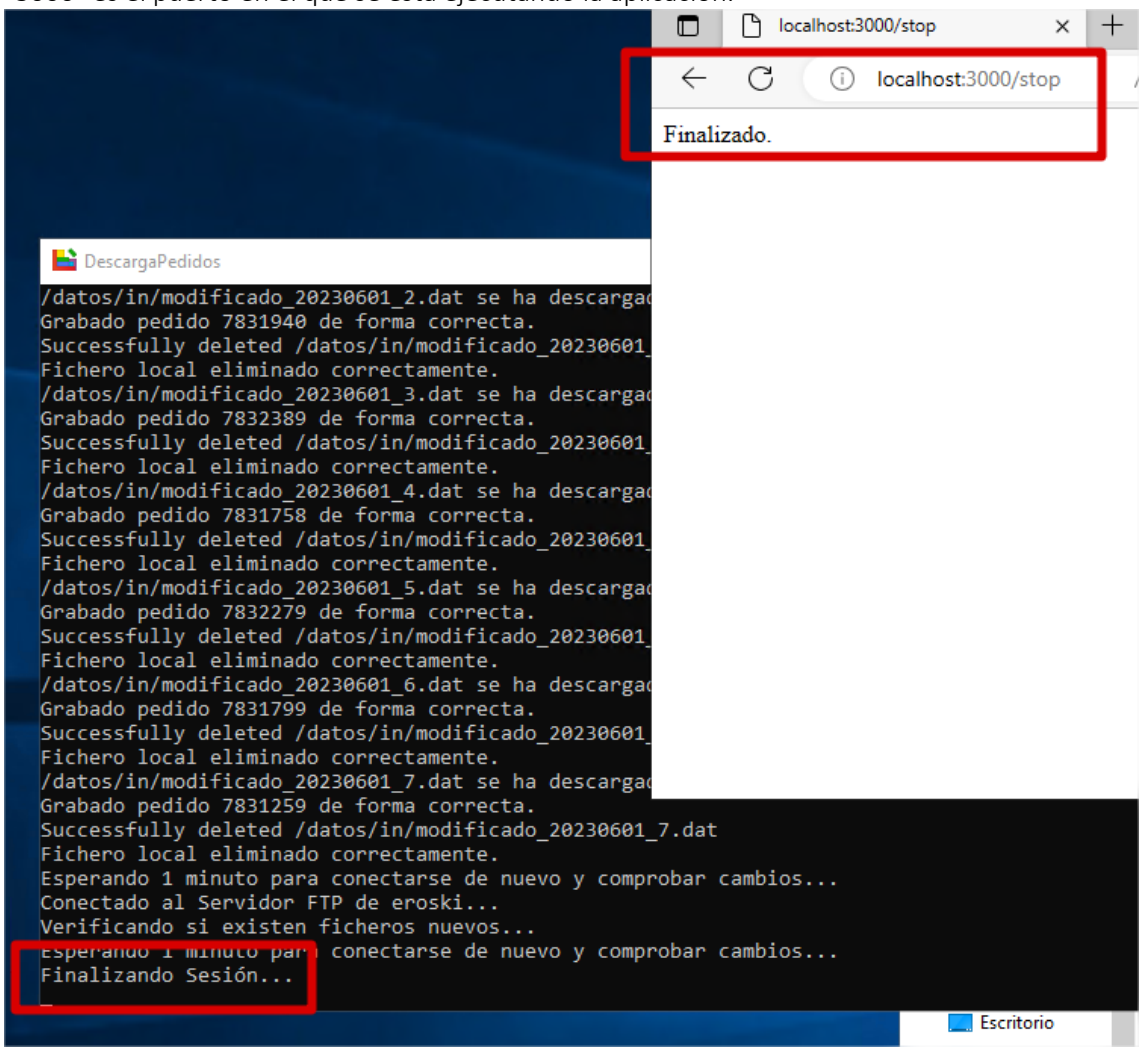
Es importante asegurarse de que estas carpetas existen en el servidor FTP y de que la aplicación tiene los permisos adecuados para leer, escribir y eliminar archivos en estas carpetas. Si hay un problema con las carpetas /in y /out, es posible que la aplicación no funcione correctamente.
Durante la ejecución de la sesión, en cada ciclo de 60 segundos, la aplicación intentará establecer una conexión con el servidor de Eroski, descargar cualquier archivo nuevo que se encuentre en la carpeta remota de entrada del servidor FTP, leer y cargar el contenido del archivo en la base de datos MongoDB, y luego mover el archivo a la carpeta /out del servidor remoto. La aplicación eliminará cualquier archivo que haya descargado localmente después de procesarlo y moverlo a la carpeta de salida del servidor remoto.
Para cada uno de los procesos descritos se mostrará un mensaje en la ventana de la aplicación:



Si ocurre algún error durante el procesamiento de un archivo, la aplicación registrará el error y pasará al siguiente archivo. Si la aplicación no puede conectar con el servidor de Eroski o si hay un problema grave, la aplicación detendrá la sesión.

5.- Finalización de la sesión de descarga de archivos

Para finalizar la sesión de descarga de archivos, use la ruta /stop. Esta acción detendrá el intervalo de comprobación de archivos nuevos y la descarga y procesado de estos.
Por ejemplo, puede finalizar la sesión de descarga de archivos visitando la dirección http://localhost:3000/stop en su navegador web, donde "localhost" es la dirección de su servidor y "3000" es el puerto en el que se está ejecutando la aplicación.



6.- Descripción de posibles errores

1. No se puede conectar al servidor de Eroski: Este error puede ocurrir si la dirección del servidor FTP, el puerto, el nombre de usuario o las contraseñas son incorrectos. También puede ocurrir si la VPN o el servidor de Eroski está caído o no está accesible.

2. Error al descargar el archivo: Este error puede ocurrir si hay un problema con la conexión de red durante la descarga del archivo, si el archivo no existe en el servidor de Eroski, o si hay un problema con los permisos del archivo en el servidor.

3. Error al leer el archivo local: Este error puede ocurrir si hay un problema con el sistema de archivos local, si el archivo ha sido eliminado o modificado después de la descarga, o si hay un problema con los permisos del archivo.

4. Error al cargar el archivo en la base de datos: Este error puede ocurrir si hay un problema con la conexión a la base de datos.

5. Error al mover el archivo a la carpeta /out: Este error puede ocurrir si hay un problema con la conexión de red o la VPN durante la transferencia del archivo, si la carpeta /out no existe, o si hay un problema con los permisos de la carpeta.

6. Error al eliminar el archivo local: Este error puede ocurrir si hay un problema con el sistema de archivos local o si hay un problema con los permisos del archivo.

7. El centro no existe: Este error puede ocurrir si el código de centro (tienda) proporcionado en el archivo no existe en la base de datos.

8. El pedido ya existe: Este error puede ocurrir si el nombre del archivo ya existe en la base de datos, lo que indica que el pedido ya ha sido procesado.

9. Error de validación del pedido: Este error puede ocurrir si los datos en el archivo no cumplen con las reglas de validación de pedidos (pedidos con valores vacios, etc.).

Anexo II - Manual Aplicación web gestión de pedidos

El presente manual de usuario proporciona instrucciones para utilizar la aplicación de gestión de pedidos. Esta aplicación, desarrollada en lenguaje PHP mediante framework Laravel, explota la información almacenada en la base de datos MongoDB cargada previamente por la aplicación de descarga automática de pedidos. La aplicación proporciona funcionalidades de gestión de pedidos (Consulta de pedidos pendientes, modificación de estados, etc.), obtención de informes y estadísticas, así como cuadros de facturación mensual. También permite la gestión de los usuarios de la aplicación, registrando las nuevas altas, así como las bajas de los usuarios que dejan la empresa.

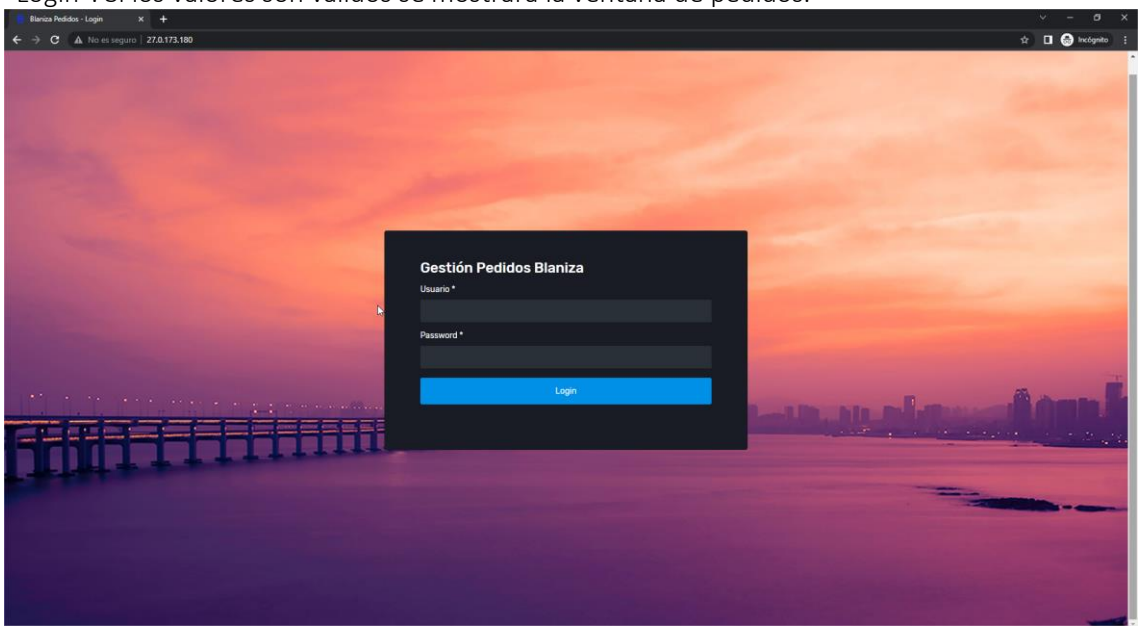
1.- Instalación de la aplicación

Los pasos para poder instalar la aplicación en un servidor son los siguientes:

1. Es necesario tener instalado PHP y Composer (La aplicación se ha instalado con la versión 8 de PHP que se puede obtener del sitio oficial http://php.net o también descargando una versión de XAMPP que ya lo incluya. Composer se puede descargar de https://getcomposer.org/)
2. Extraer los ficheros con el código fuente en una carpeta
3. Ejecutar el comando “php artisan serve” para ejecutar el propio servidor de desarrollo que incorpora laravel.
4. Otra opción es copiarlo en la carpeta htdocs de un servidor como XAMPP y servirlo desde allí.

2.- Inicio de sesión

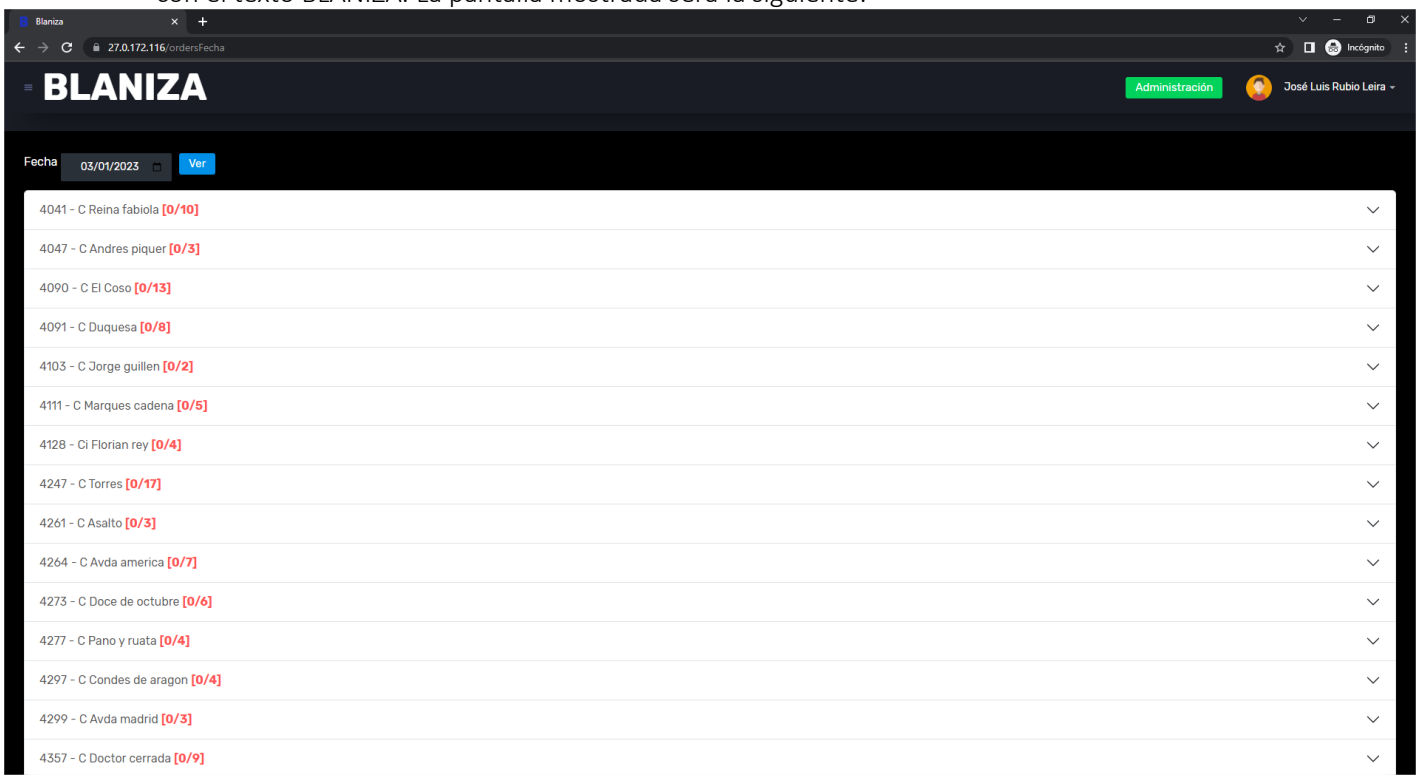
Al entrar por primera vez en la aplicación (introduciendo la URL del servidor donde esté alojada en la barra de direcciones del navegador), se nos pedirá iniciar sesión en la aplicación. Para ello se introducirá el nombre de usuario y la contraseña en los campos proporcionados y se pulsará el botón “Login”. Si los valores son válidos se mostrará la ventana de pedidos.



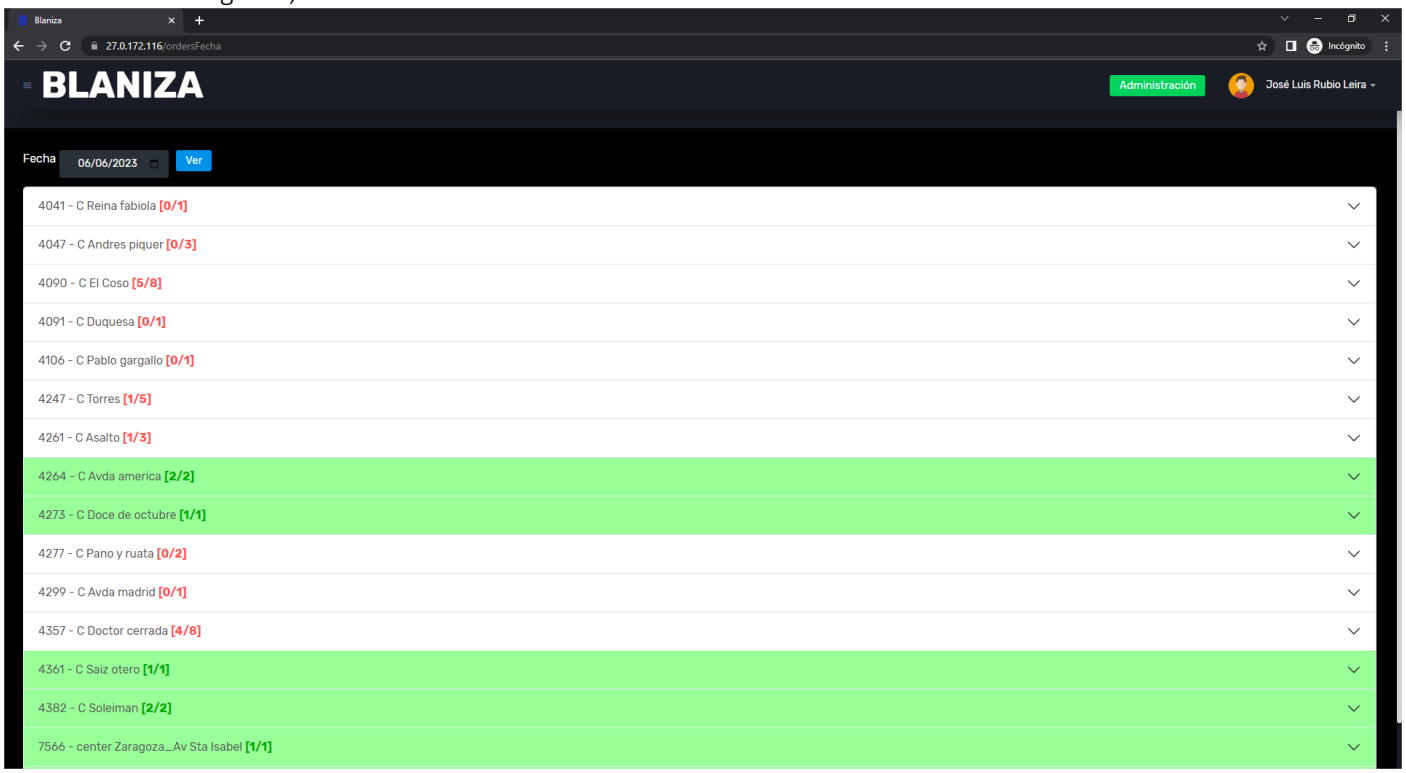
3.- Gestión de los pedidos

Tras iniciar sesión se mostrará en un objeto de tipo acordeón el listado de todas las tiendas que tienen pedidos cargados en el día actual. A la derecha del nombre de cada tienda, en rojo, tendremos un indicador de pedidos entregados sobre el total de pedidos para dicha tienda. Se podrá acceder

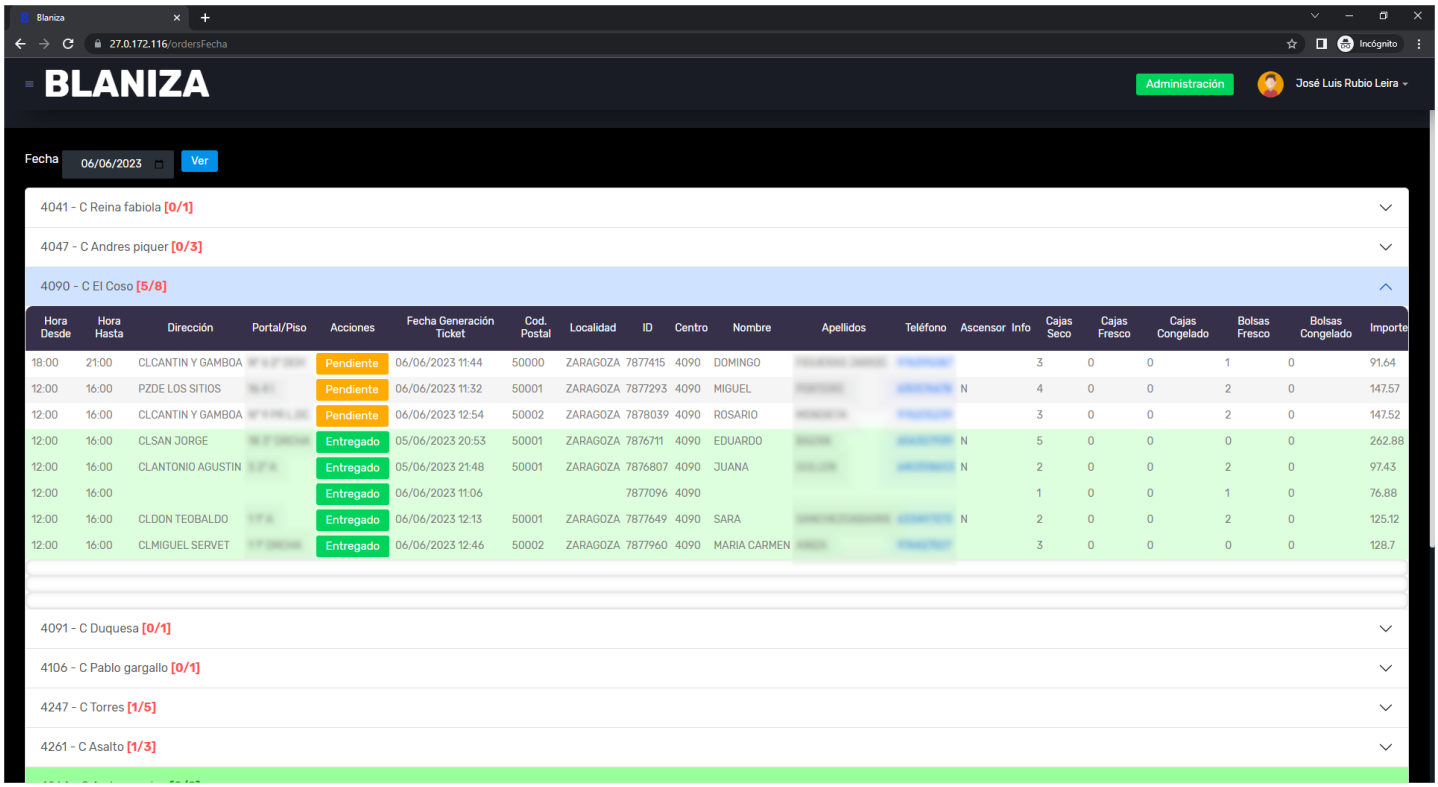
directamente a esta pantalla o recargarla en cualquier momento pulsando el logotipo de la aplicación con el texto BLANIZA. La pantalla mostrada será la siguiente:



A medida que se vayan entregando pedidos, cuando todos los pedidos pendientes de una tienda estén entregados, se nos mostrará el nombre de dicha tienda en color verde.

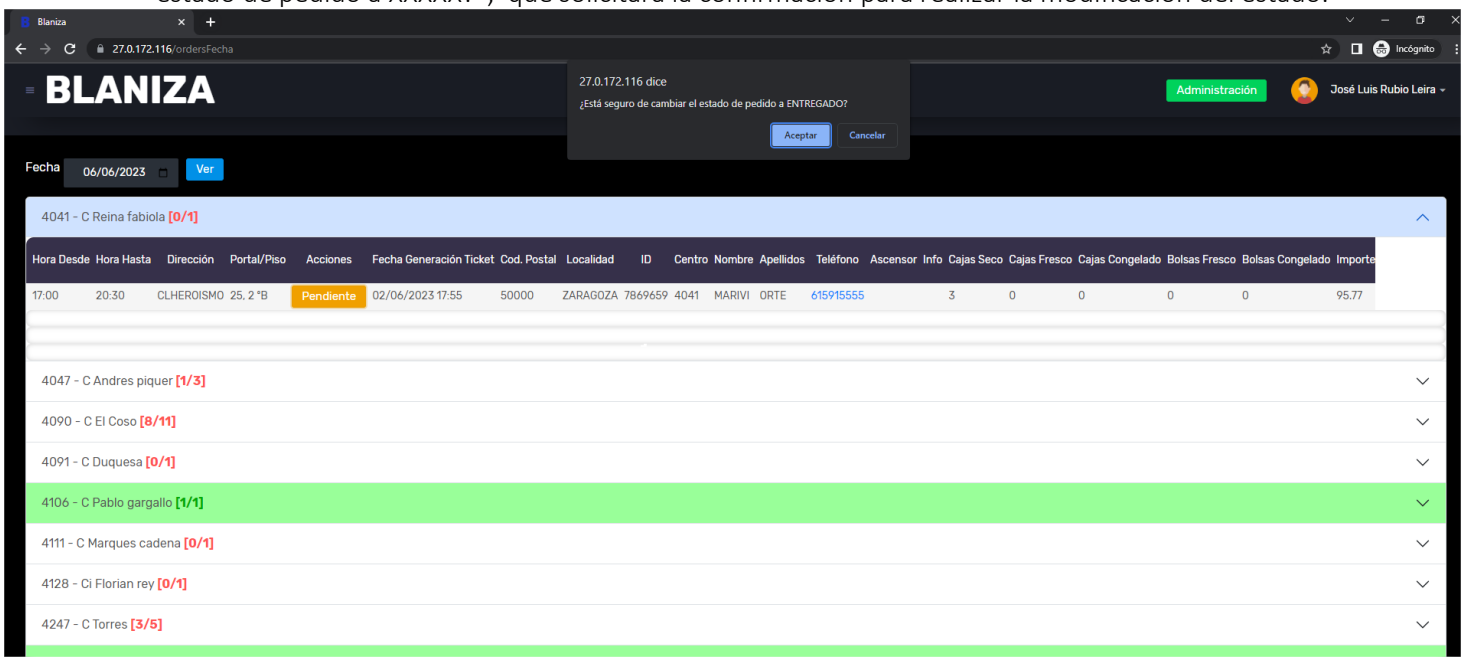


Al pulsar en el registro de una tienda, se desplegará el acordeón y se verán los pedidos que se han recibido en ese día para esa tienda. Siempre aparecerán en la parte superior los pedidos pendientes de entrega, quedando en la parte inferior los pedidos que hayan pasado a estado "*Entregado*".



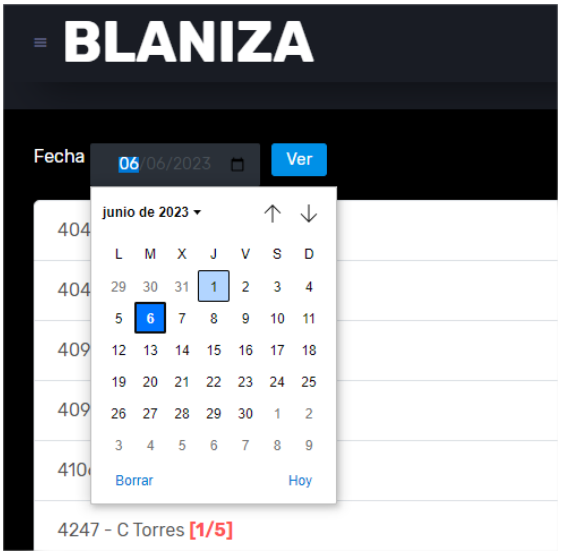
3.1.- Modificación del estado de un pedido

Para modificar el estado de un pedido de “Pendiente” a “Entregado” o viceversa, se pulsará en el btón correspondiente dentro del pedido. Al hacerlo se mostrará un mensaje: “¿Está seguro de cambiar el estado de pedido a XXXXX?”, que solicitará la confirmación para realizar la modificación del estado:

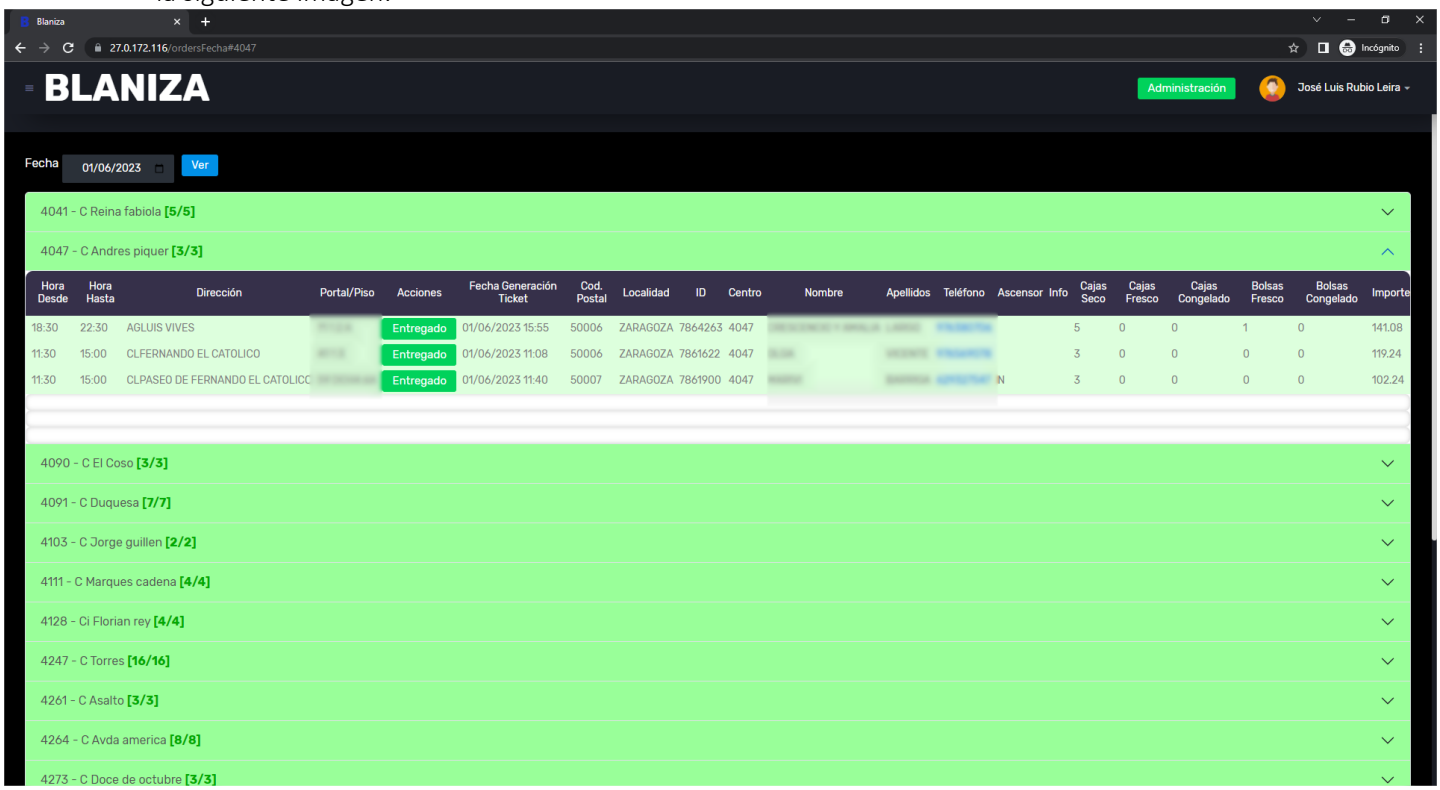


3.2.- Consultar pedidos de otras fechas

Para consultar los pedidos de otras fechas, tanto pasadas como futuras, es posible utilizar el selectro de fechas que se encuentra en la parte superior de la pantalla. Para ello, se seleccionará la fecha deseada y se pulsará el botón “*Ver*”,



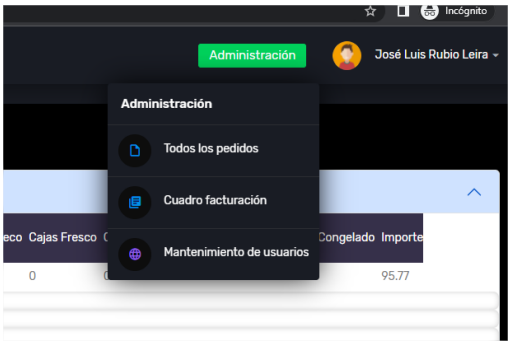
Una vez hecho esto, aparecerán los pedidos correspondientes a la fecha indicada. Normalmente todos los pedidos de fechas pasadas aparecerán en verde por estar todos entregados como se puede ver en la siguiente imagen:



Si se desea volver a ver los pedidos del día actual se podrá hacer simplemente haciendo click en el logotipo de la aplicación.

4.- Opciones de administración

En el caso de que se inicie sesión en la aplicación con un usuario del perfil '*GESTIÓN*', se mostrará en la parte izquierda de la cabecera de la pantalla un botón verde con el texto 'Administración'. Este botón da acceso a las opciones específicas del perfil '*GESTIÓN*'.

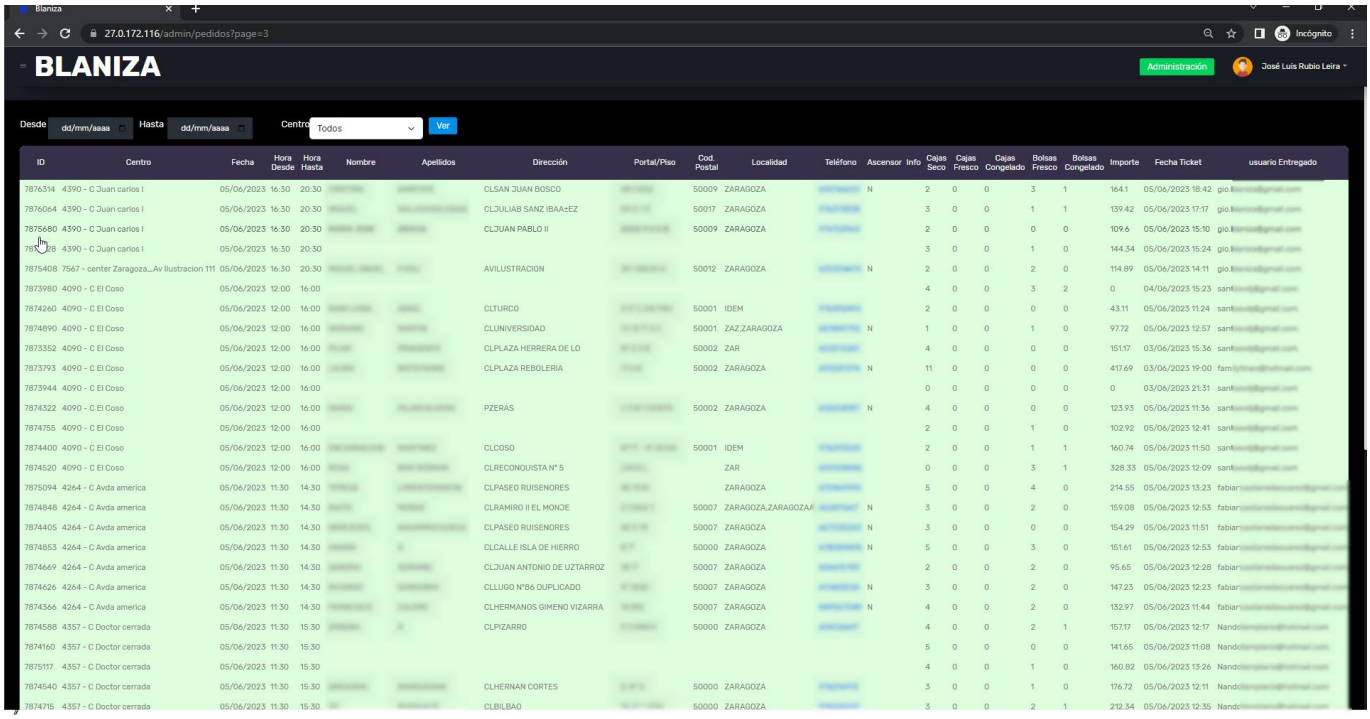


Estas opciones son las siguientes:

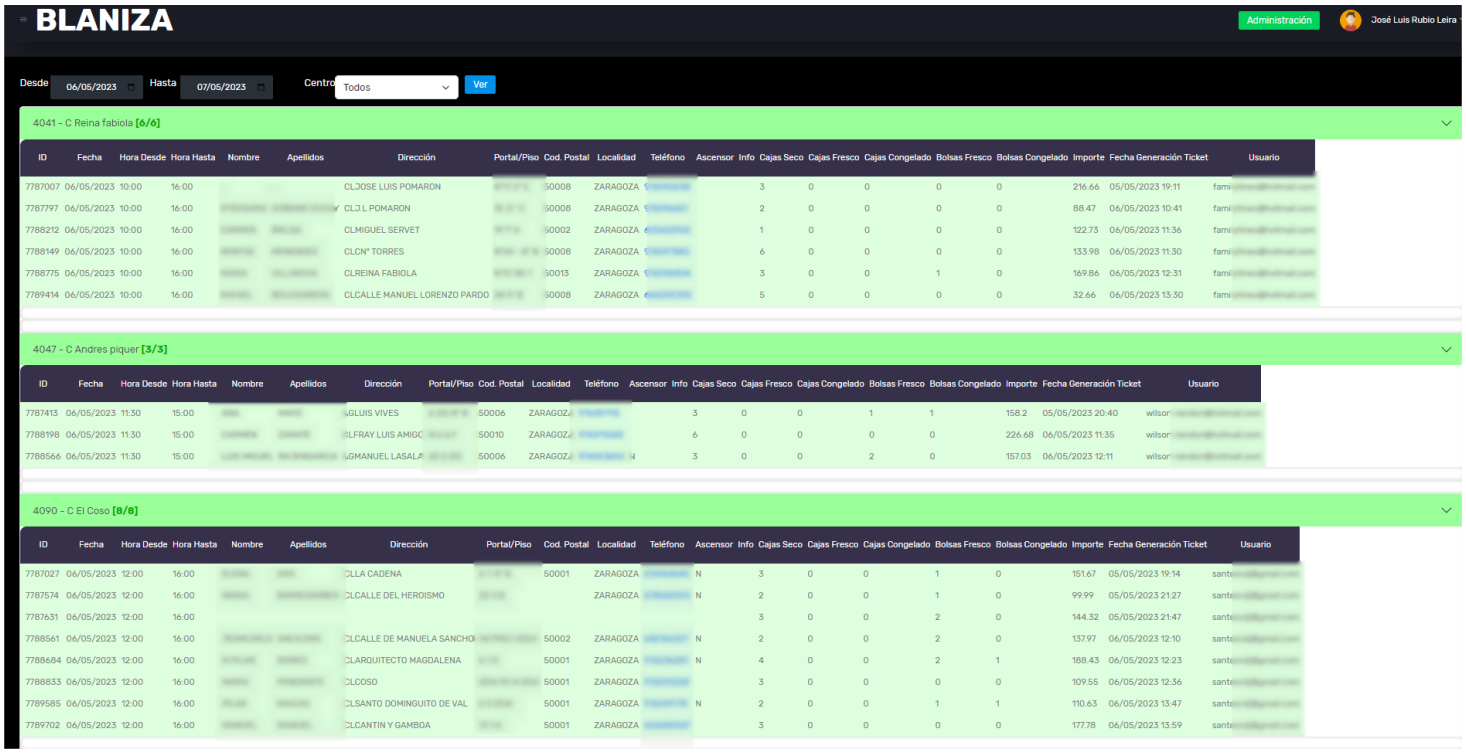
- **Todos los pedidos:** Informe de datos de pedidos entre pudiendo establecer filtros por rango de fechas y por tienda.
- **Cuadro de facturación**: Esta opción permitirá generar un cuadro con el desglose de los pedidos de cada tienda en un periodo determinado, que por defecto será mensual. Esta será la información que se entregue con la facturación que se le haga a Eroski.
- **Mantenimiento de usuarios**: Desde esta opción se podrán dar de alta y de baja usuarios en el sistema.

4.1.- Informe de pedidos

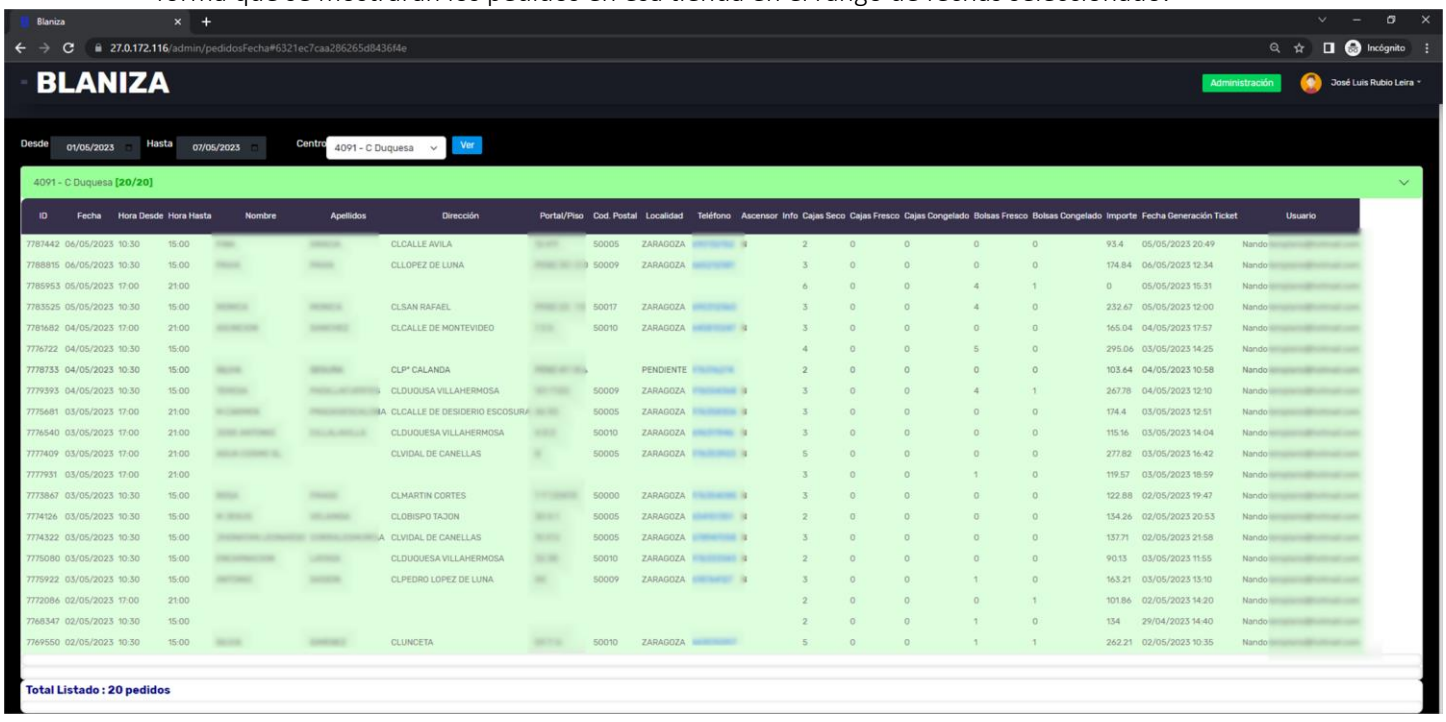
Mediante la opción "Todos los pedidos" del menú de administración, accederemos a la siguiente consulta, donde se nos mostrará de forma paginada una consulta con todos los pedidos que existan en el sistema:



Mediante las opciones de filtrado de la cabecera se podrá limitar la consulta entre un rango de fechas, en cuyo caso, aparecerán los datos de los pedidos en ese rango de fechas, agrupados por tienda.

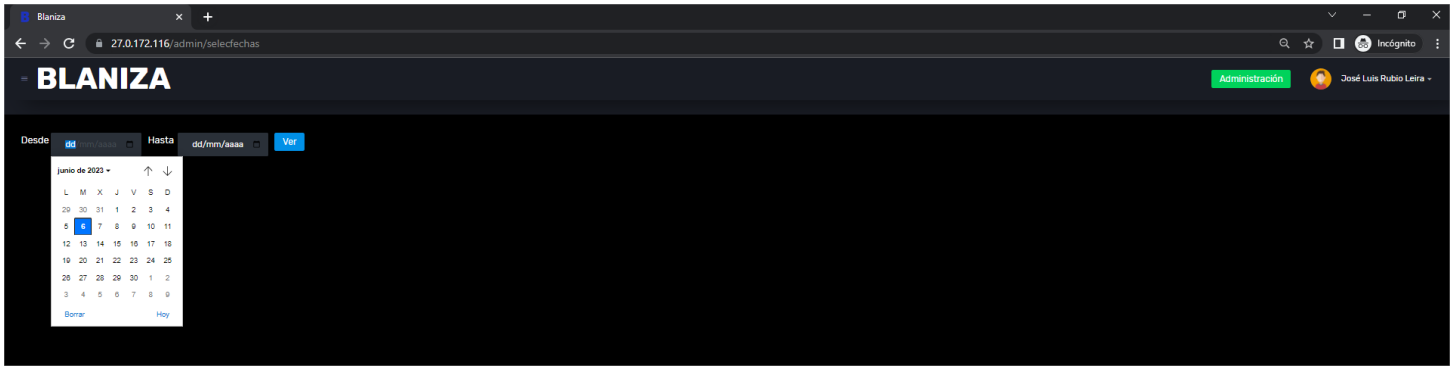


También se podrá acotar el ámbito del informe para obtener los datos de una tienda determinada, de forma que se mostrarán los pedidos en esa tienda en el rango de fechas seleccionado:

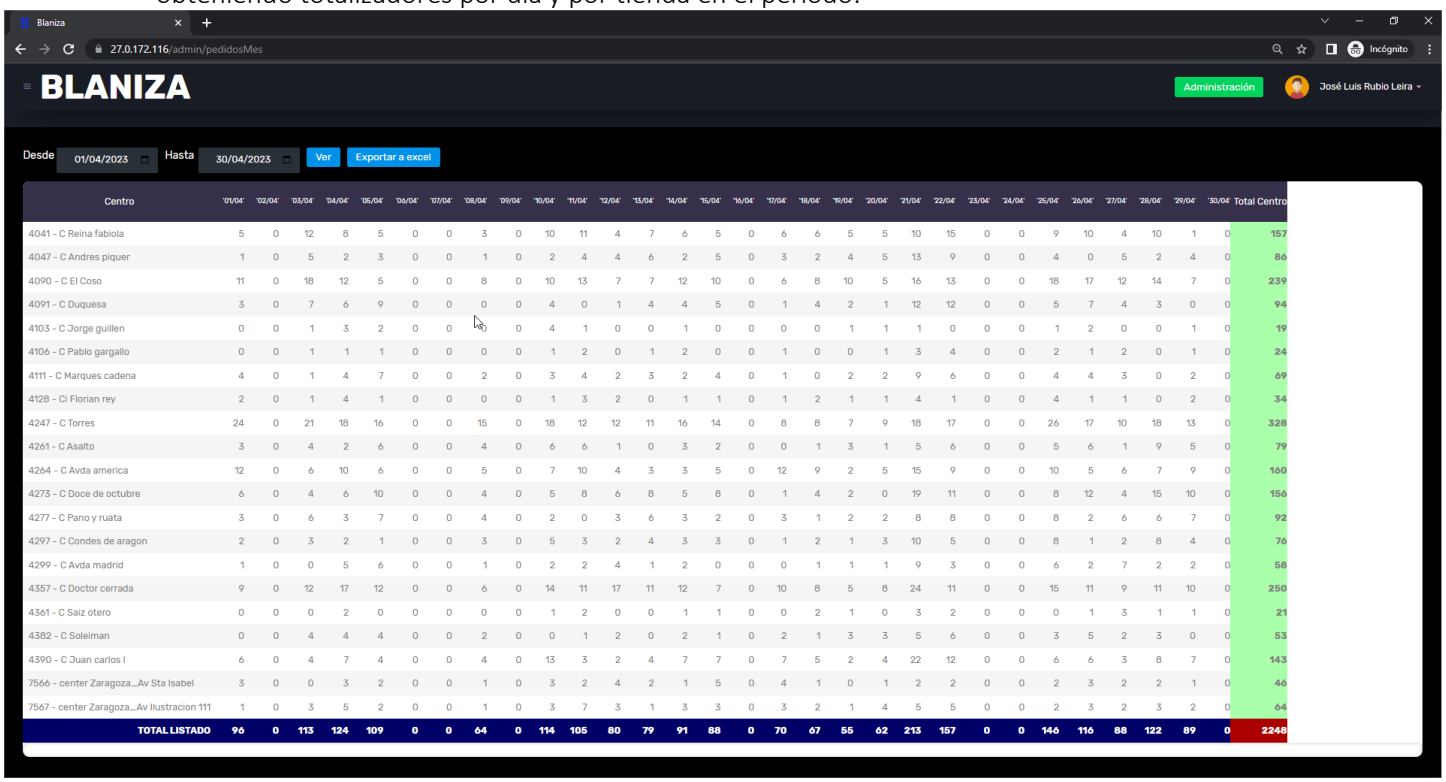


4.2.- Cuadro de facturación

Esta opción permitirá generar un cuadro con el desglose de los pedidos de cada tienda en un periodo determinado, que por defecto será mensual. Esta será la información que se entregue con la facturación que se le haga a Eroski. Cuando se acceda a la opción, se mostrarán los selectores de fecha para acotar la información del cuadro al rango que interese. Al seleccionar la fecha inicial, automáticamente se pondrá la fecha final un més más tarde de la fecha inicial, pudiendo este valor ser cambiado manualmente.



Una vez seleccionado el rango de fechas deseado, al pulsar el botón '*Ver*' obtendremos el cuado de facturación correspondiente donde veremos por cada día y tienda el número de pedidos entregados, obteniendo totalizadores por día y por tienda en el periodo:

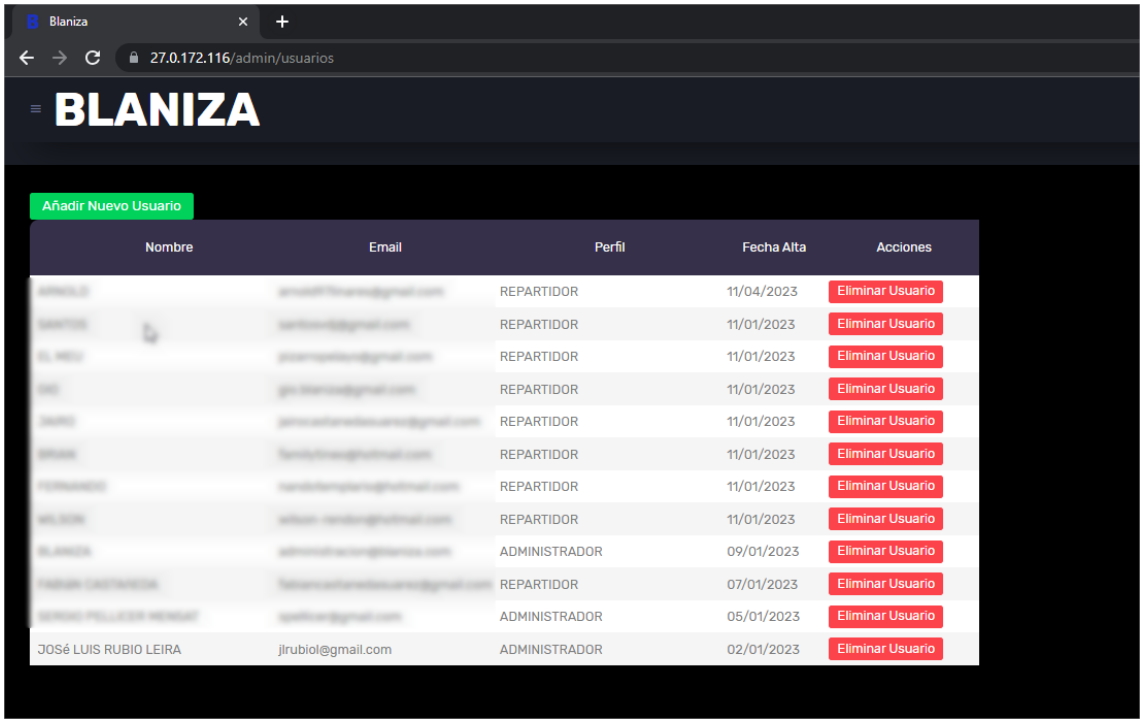


Centro	'01/04'	'02/04'	'03/04'	'04/04'	'05/04'	'06/04'	'07/04'	'08/04'	'09/04'	'10/04'	'11/04'	'12/04'	'13/04'	'14/04'	'15/04'	'16/04'	'17/04'	'18/04'	'19/04'	'20/04'	'21/04'	'22/04'	'23/04'	'24/04'	'25/04'	'26/04'	'27/04'	'28/04'	'29/04'	'30/04'	Total Centro
4041 - C Reina fabiola	5	0	12	8	5	0	0	3	0	10	11	4	7	6	5	0	6	6	5	5	10	15	0	0	9	10	4	10	1	0	157
4047 - C Andres piquer	1	0	5	2	3	0	0	1	0	2	4	4	6	2	5	0	3	2	4	5	13	9	0	0	4	0	5	2	4	0	86
4090 - C El Coso	11	0	18	12	5	0	0	8	0	10	13	7	7	12	10	0	6	8	10	5	16	13	0	0	18	17	12	14	7	0	239
4091 - C Duquesa	3	0	7	6	9	0	0	0	0	4	0	1	4	4	5	0	1	4	2	1	12	12	0	0	5	7	4	3	0	0	94
4103 - C Jorge guillen	0	0	1	3	2	0	0	0	0	4	1	0	0	1	0	0	0	0	1	1	1	0	0	0	1	2	0	0	1	0	19
4106 - C Pablo gargallo	0	0	1	1	1	0	0	0	0	1	2	0	1	2	0	0	1	0	0	1	3	4	0	0	2	1	2	0	1	0	24
4111 - C Marques cadena	4	0	1	4	7	0	0	2	0	3	4	2	3	2	4	0	1	0	2	2	9	6	0	0	4	4	3	0	2	0	69
4128 - Ci Florian rey	2	0	1	4	1	0	0	0	0	1	3	2	0	1	1	0	1	2	1	1	4	1	0	0	4	1	1	0	2	0	34
4247 - C Torres	24	0	21	18	16	0	0	15	0	18	12	12	11	16	14	0	8	8	7	9	18	17	0	0	26	17	10	18	13	0	328
4261 - C Asalto	3	0	4	2	6	0	0	4	0	6	6	1	0	3	2	0	0	1	3	1	5	6	0	0	5	6	1	9	5	0	79
4264 - C Avda america	12	0	6	10	6	0	0	5	0	7	10	4	3	3	5	0	12	9	2	5	15	9	0	0	10	5	6	7	9	0	160
4273 - C Doce de octubre	6	0	4	6	10	0	0	4	0	5	8	6	8	5	8	0	1	4	2	0	19	11	0	0	8	12	4	15	10	0	156
4277 - C Pano y ruata	3	0	6	3	7	0	0	4	0	2	0	3	6	3	2	0	3	1	2	2	8	8	0	0	8	2	6	6	7	0	92
4297 - C Condes de aragon	2	0	3	2	1	0	0	3	0	5	3	2	4	3	3	0	1	2	1	3	10	5	0	0	8	1	2	8	4	0	76
4299 - C Avda madrid	1	0	0	5	6	0	0	1	0	2	2	4	1	2	0	0	0	1	1	1	9	3	0	0	6	2	7	2	2	0	58
4357 - C Doctor cerrada	9	0	12	17	12	0	0	6	0	14	11	17	11	12	7	0	10	8	5	8	24	11	0	0	15	11	9	11	10	0	250
4361 - C Saiz otero	0	0	0	2	0	0	0	0	0	1	2	0	0	1	1	0	0	2	1	0	3	2	0	0	0	1	3	1	1	0	21
4382 - C Soleiman	0	0	4	4	4	0	0	2	0	0	1	2	0	2	1	0	2	1	3	3	5	6	0	0	3	5	2	3	0	0	53
4390 - C Juan carlos I	6	0	4	7	4	0	0	4	0	13	3	2	4	7	7	0	7	5	2	4	22	12	0	0	6	6	3	8	7	0	143
7566 - center Zaragoza...Av Sta Isabel	3	0	0	3	2	0	0	1	0	3	2	4	2	1	5	0	4	1	0	1	2	2	0	0	2	3	2	2	1	0	46
7567 - center Zaragoza...Av Ilustracion 111	1	0	3	5	2	0	0	1	0	3	7	3	1	3	3	0	3	2	1	4	5	5	0	0	2	3	2	3	2	0	64
TOTAL LISTADO	**96**	**0**	**113**	**124**	**109**	**0**	**0**	**64**	**0**	**114**	**105**	**80**	**79**	**91**	**88**	**0**	**70**	**67**	**55**	**62**	**213**	**157**	**0**	**0**	**146**	**116**	**88**	**122**	**89**	**0**	**2248**

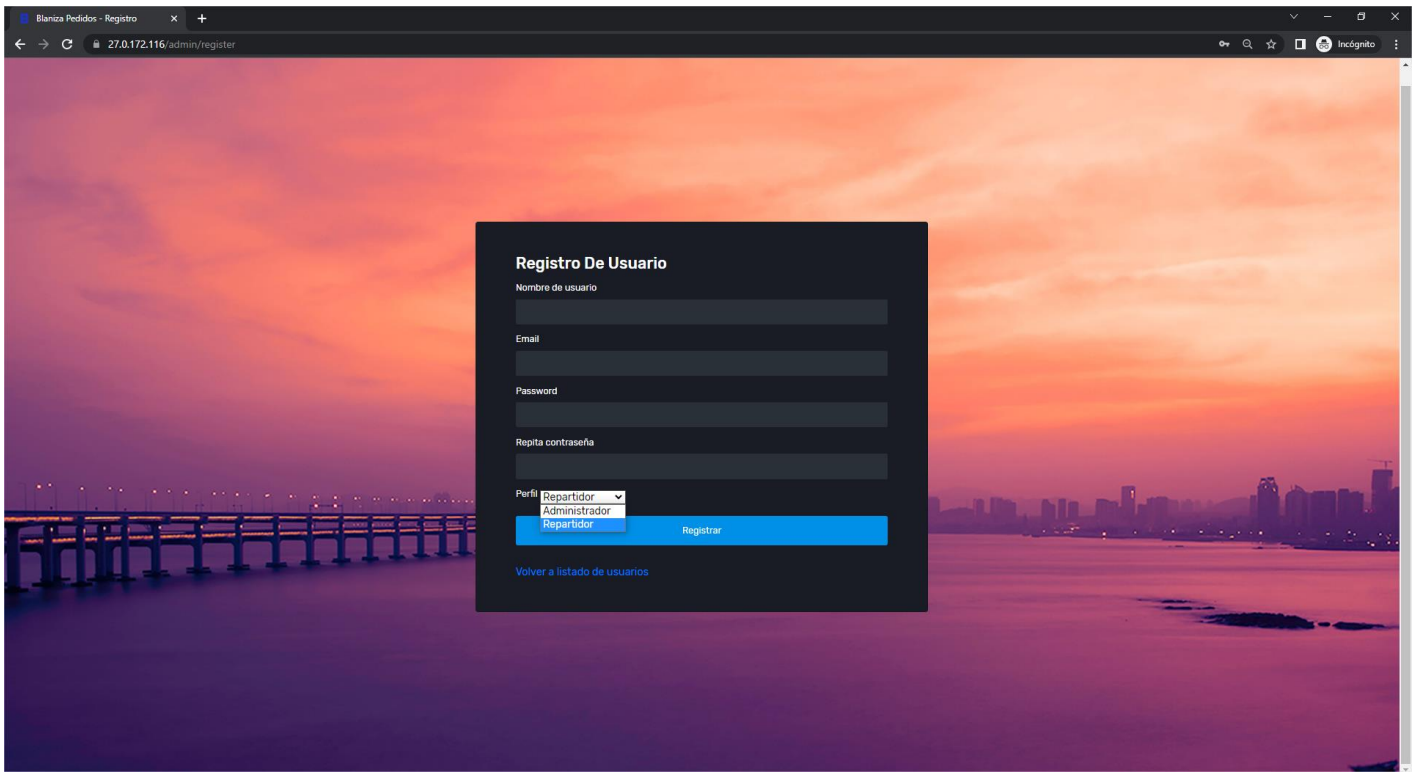
Se podrá exportar los datos del cuadro de facturación a una hoja Excel pulsando el botón de la cabecera '*Exportar a Excel*'. Al hacerlo se descargará un fichero con el nombre cuadro.xlsx con los datos.

4.2.- Mantenimiento de usuarios

Accediendo a esta opción veremos un listado con los datos de los usuarios disponibles en el sistema. En este listado aparecerá el nombre del usuario, el email que hará las veces de código de usuario, el perfil que será REPARTIDOR o ADMINISTRADOR (perfil gestión) y la fecha de alta de dicho usuario
Se podrá eliminar un usuario determinado, pulsando el botón 'Eliminar Usuario' y confirmando la acción posteriormente.



Para registrar un nuevo usuario, pulsaremos el botón verde de la cabecera que dice '*Añadir Nuevo Usuario*'. Al hacerlo, se mostrará la ventana de registro de nuevos usuarios:



Para dar de alta el nuevo usuario, se rellenarán los datos incluyendo el perfil del usuario en la aplicación, y posteriormente se pulsará el botón '*Registrar*'. Si los datos son correctos, se mostrará un mensaje de éxito del registro. Si existe algún problema se mostrará igualmente con un mensaje. Para volver al mantenimiento de pedidos se pulsará sobre el enlace que dice '*Volver a listado de usuarios*'.

Anexo III - Manual Aplicación móvil Android

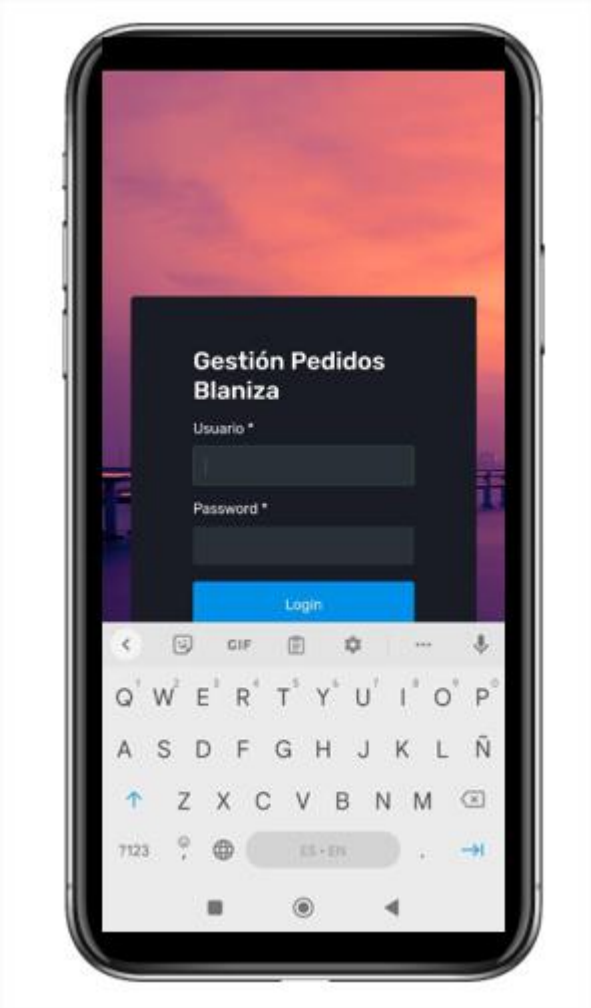
1.- Instalación de la aplicación

Para instalar la aplicación, habrá que copiar en un dispositivo Android el fichero Blaniza.apk y ejecutarlo. El dispositivo deberá tener activada la opción de instalación de aplicaciones desde orígenes desconocidos. Transconfirmar que se desea instalar la aplicación, aparecerá un icono en el escritorio o en el cajón de aplicaciones dependiendo del dispositivo.

Para ejecutar la aplicación simplemente haremos click en el icono rosa con una B blanca con el título "*Blaniza Pedidos*".

2.- Inicio de sesión

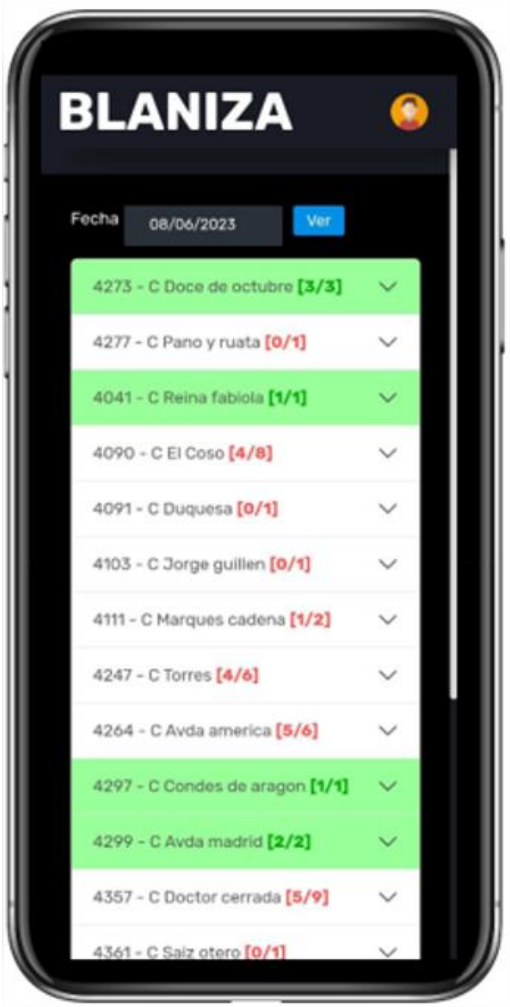
Tras ejecutar la aplicación se nos pedirá que iniciemos sesión si no lo hemos hecho antes. Para ello veremos la siguiente pantalla en la que se dispone de dos cajas de texto para introducir el usuario y la contraseña del usuario:



Tras rellenar los datos y pulsar el botón '*Login*', si los datos son correctos accederemos a la ventana principal de gestión de pedidos. Si existe algún error, se mostrará un mensaje indicándolo. Para que el uso de la aplicación sea lo más eficiente posible, El tiempo de duración de sesión está establecido en un valor lo suficientemente amplio para que solo sea necesario el iniciar sesión una vez cada día.

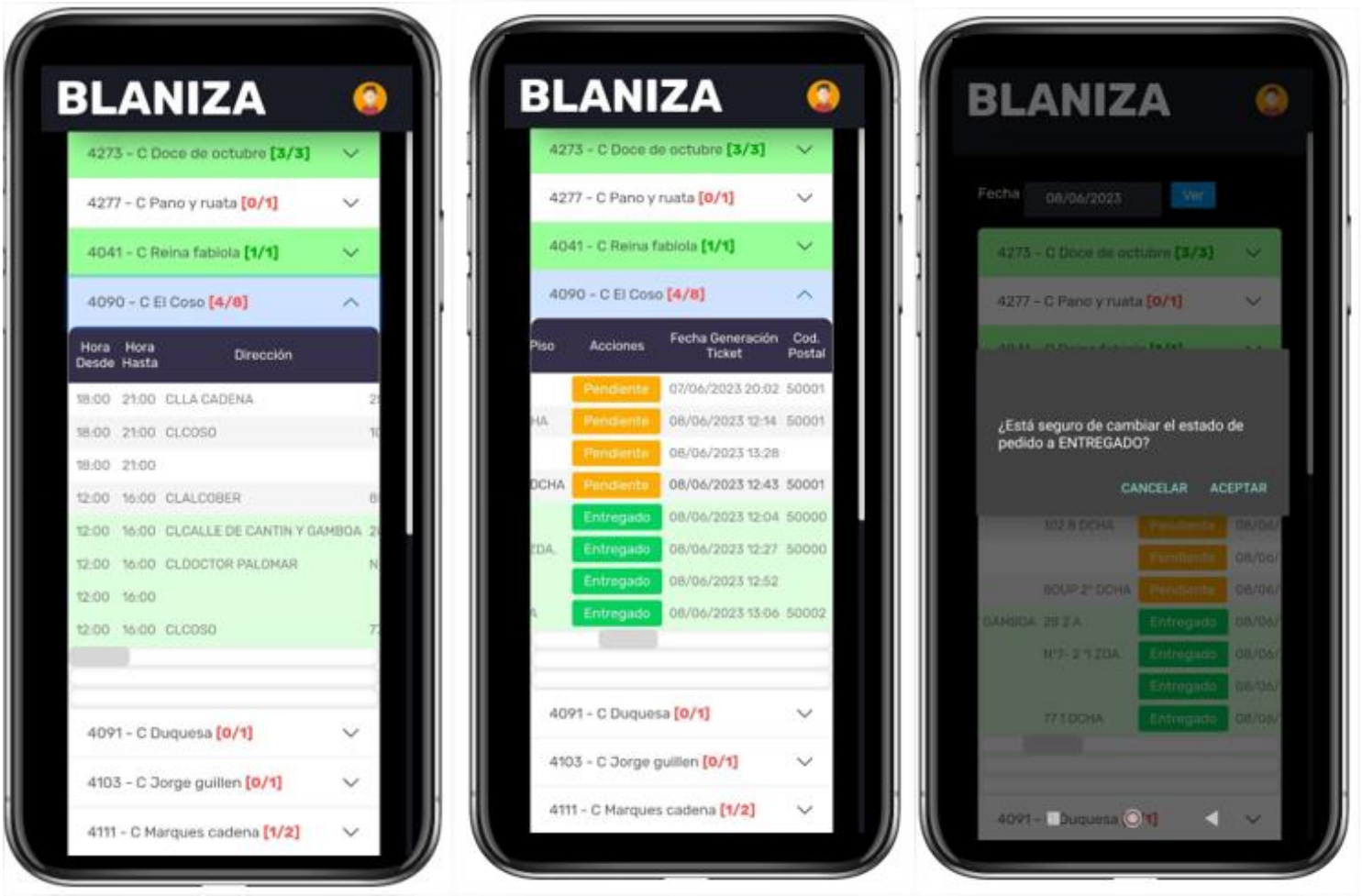
3.- Gestión de pedidos

Tras iniciar sesión con éxito, accederemos a la pantalla de gestión de pedidos. En dicha pantalla veremos un listado de las tiendas en las que se nos indicará mediante dos valores entre corchetes en texto rojo, el número de pedidos que están pendientes sobre el total de pedidos de la tienda. En el caso de que todos los pedidos de una tienda hayan sido entregados, el nombre de la tienda se mostrará en color verde como se puede observar en la siguiente imagen:



Cada usuario puede tener asignadas unas tiendas concretas, pero en un momento dado, puede necesitar conocer los datos de cualquiera de las otras tiendas por tener que ayudar por motivos de demanda. El orden de aparición de las tiendas en el listado será el siguiente: Primero aparecerán las tiendas que el usuario tiene asignadas, y posteriormente el resto de tiendas ordenadas por código de tienda. De esta manera, cada repartidor siempre tendrá en la parte superior del listado las tiendas en las que está asignado.

Para desplegar los pedidos del día de una de las tiendas, se pulsará el nombre de la tienda correspondiente. Esto desplegará en un objeto de tipo acordeón, los pedidos correspondientes a la tienda seleccionada. Los pedidos entregados figurarán en verde al final de la lista de pedidos, y en la parte superior se podrán ver los pedidos que están pendiente de entrega.
Los primeros datos que se visualizarán del pedido serán las direcciones y las horas de entrega de los pedidos. No obstante, desplazando a la derecha la lista de pedidos tenemos todo el resto de datos, incluyendo el estado del pedido, los datos y el teléfono del cliente, el importe del pedido, etc.



Como se puede ver en las capturas, pulsando en el botón de estado del pedido se podrá cambiar dicho estado de Pendiente a Entregado cuando un repartidor haya realizado una entrega. Al hacerlo, el pedido se pondrá en color verde y pasará a la parte inferior de la lista con el resto de pedidos Entregados. También se actualizarán los contadores de pedidos entregados sobre el total de pedidos. El objetivo es que todos los pedidos acaben estando en color verde.

También será posible el ver los pedidos de una fecha pasada o futura. Es interesante el que se puedan ver pedidos de fechas futuras, ya que un cliente puede pedir que le entreguen un pedido en una fecha futura, de forma que por ejemplo se pueden ver los pedidos pendientes para por ejemplo, el día siguiente. Para ello, se utilizará el selector de fechas ubicado en la parte superior de la aplicación y se seleccionará la fecha cuyos pedidos se quieran consultar.

Una vez seleccionada la fecha, y tras pulsar el botón ‘Ver’, se mostrarán los pedidos asociados a la fecha indicada.

4.- Opciones de administración

Aunque un usuario con perfil de gestión puede utilizar la aplicación móvil y acceder a las opciones de administración, esto no es lo recomendado ya que dichas opciones han sido diseñadas para ser utilizadas en un entorno de escritorio. Para acceder a dichas opciones se podrá acceder pulsando el icono del usuario situado en la parte superior derecha de la ventana.

Para obtener indicaciones acerca de dichas opciones, se recomienda consultar el manual de usuario de la aplicación web de gestión de pedidos.