UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

# TESINA DE LICENCIATURA

**TÍTULO:** Innovación tecnológica en el comercio local: Desarrollo de la plataforma web para "El Paseo te lo lleva"

**AUTORES:** Melisa Ailén Onofri

**DIRECTORA:** Claudia Queiruga

**DIRECTOR:** Jorge Rosso

**ASESORA PROFESIONAL:** Soledad Rial

## Resumen

*Esta tesis de grado de Licenciatura en Sistemas aborda las problemáticas de la Economía Social y Solidaria, específicamente las relacionadas a la comercialización, surgidas en las ferias de familias productoras y de artesanos de la UNLP durante la pandemia de COVID-19. La digitalización de dicho proceso es la motivación de este proyecto de tesis. Para ello, se trabajó en conjunto con la comercializadora de la UNLP "El Paseo te lo lleva" en la co-construcción de un portal web de código fuente abierto y basado en tecnologías libres, para automatizar el proceso de comercialización online de los productos que se ofrecen en la feria y además fortalecer los lazos sociales entre productores y consumidores. El portal se construyó siguiendo los lineamientos de la Ingeniería de Software: se describen y analizan los diferentes productos elaborados durante las distintas etapas que ésta establece. Las pruebas realizadas con los diferentes perfiles de usuario arrojaron resultados satisfactorios que alientan a la puesta en producción del portal.*

## Palabras Clave

*Economía Social y Solidaria, Mercados Solidarios, Desarrollo Web, Java, Angular, Ingeniería de Software, Software Libre, Código Fuente Abierto.*

## Conclusiones

*Desde lo personal esta experiencia de tesis me permitió hacer uso de los conocimientos adquiridos a lo largo de mi carrera universitaria y ponerlos en práctica resolviendo un problema social de nuestra comunidad, como es la comercialización online de la comercializadora de la UNLP "El Paseo te lo lleva". Este desarrollo fue un desafío constante: la co-construcción con los adoptantes, la implementación de tecnologías con las que no contaba con experiencia previa, ser responsable de principio a fin del proyecto, y lograr alcanzar las expectativas de los usuarios y la organización adoptante. Se logró recrear las ferias presenciales desde Internet y mantener los lazos sociales entre productores y consumidores, que son los valores de "El Paseo te lo lleva". Además, se logró automatizar al cien por ciento el proceso de comercialización, y se ofrecieron nuevas herramientas confiables y eficientes para realizar tareas estratégicas y tomar decisiones informadas.*

## Trabajos Realizados

*- Análisis del enfoque económico de la Economía Social y Solidaria y el rol del Consejo Social de la UNLP en el fortalecimiento del subsistema económico en la región.*
*- Estudio de la comercializadora de la UNLP "El Paseo te lo lleva".*
*- Evaluación de tecnologías libres y de código fuente abierto que mejor se adecúan al desarrollo del portal web.*
*- Desarrollo de una aplicación web que resuelve la comercialización online de la comercializadora de la UNLP "El Paseo te lo lleva" siguiendo los lineamientos de las distintas etapas del desarrollo de software de la Ingeniería de Software para lograr un producto final de calidad que resuelva las necesidades del cliente y cumpla con sus expectativas.*

## Trabajos Futuros

*- Desarrollar una aplicación móvil reutilizando el backend desarrollado en este proyecto.*
*- Robustecer el sistema en términos de Seguridad Informática, a partir de la detección de vulnerabilidades y su respectiva mitigación para evitar ataques.*
*- Readaptación de recursos dedicados y arquitectura del sistema de acuerdo al crecimiento de la comercializadora.*
*- Implementar herramientas para facilitar las tareas de mantenimiento del software, como ser: tecnologías de debugging y detección de errores, herramientas de integración continua, o ambiente de prueba dedicado separado al de producción.*

**Fecha de la presentación** 11/2024


# Innovación Tecnológica en el Comercio Local: Desarrollo de la plataforma web para "El Paseo te lo lleva"

Onofri Melisa Ailén, 15293/9

8 de noviembre de 2024

**Resumen**

La innovación tecnológica de "El Paseo te lo lleva" me enfrentó al desafío de poner a prueba mis habilidades como desarrolladora de software, adquiridas a lo largo de toda la carrera universitaria. Con el desarrollo del portal web de comercialización, se logró brindar una herramienta moderna para eliminar las tareas manuales y tediosas que la comercializadora realiza diariamente y que afectan negativamente la experiencia tanto de los trabajadores involucrados como de los consumidores.

La digitalización ofrece a El Paseo nuevas herramientas, modernas y confiables, que automatizan los procesos propios de la comercialización y además permite obtener información para analizar el funcionamiento de la comercializadora a lo largo del tiempo. Esto habilita a tomar decisiones basadas en datos y analizar riesgos de manera más efectiva.

En este proyecto de tesina, se explicará y analizará paso a paso el proceso llevado a cabo para llegar al producto final de software, abordando las distintas etapas del desarrollo de software: Análisis de Requerimientos, Modelado y Diseño, Codificación, y Pruebas del software.

# Agradecimientos

Quiero expresar mi más profundo agradecimiento a todas las personas que me apoyaron y guiaron a lo largo de toda la carrera universitaria.

A los profesores, directores de tesina y asesores profesionales: gracias por compartir su conocimiento y experiencia con tanta dedicación y predisposición.

Y especialmente a mi familia, quienes me apoyaron y motivaron en todo momento, desde el primer día en que me convertí en estudiante de la Facultad de Informática.

# Índice general

# Capítulo 1

# Introducción

En este proyecto de tesina se describe y analiza el proceso de desarrollo de un portal web para la comercializadora solidaria "El Paseo te lo lleva" de la Universidad Nacional de La Plata (en adelante UNLP), como propuesta de innovación tecnológica que aporte a la digitalización de los procesos de comercialización. El código fuente abierto y el software libre son valores de esta tesina.

El portal desarrollado permite la comercialización online de productos conectando productores, artesanos y emprendedores que elaboran alimentos y otros bienes de manera artesanal, con consumidores. Se trata de diversas producciones locales realizadas con trabajo cooperativo, familiar, autogestivo y asociativo, ubicados en el Gran La Plata, Brandsen, Alejandro Korn, Punta Indio, Loma Verde, Bavio y Magdalena y acompañadas por la UNLP.

El Paseo promueve nuevos circuitos comerciales cortos para la Economía Social y Solidaria (en adelante ESS) impulsando compras de proximidad de productos elaborados localmente, y basándose en los principios de la ESS.

## 1.1. Motivación

"El Paseo de la Economía Social y Solidaria" (en adelante PESyS) es una organización social que nace en el 2011 producto de varios debates llevados adelante por la Comisión de Trabajo "Economía Social y Solidaria" del Consejo Social de la UNLP [1].

El Consejo Social es un organismo asesor de la Presidencia de la UNLP, mediante el cual se propone enriquecer el proceso de retroalimentación entre

la Universidad y la comunidad, haciendo frente a situaciones de emergencia social, analizando las principales problemáticas socioeconómicas, políticas, culturales y ambientales, y discutiendo estrategias de abordaje mediante políticas locales y nacionales. Todo este trabajo influye en la mejora de la calidad de vida del conjunto de la población local.

El PESyS surge como respuesta a una demanda concreta de un sector de la sociedad: la falta de trabajo formal [2]. Comenzó funcionando como ferias mensuales de productos en el edificio de la Presidencia de la UNLP con la participación de más de 200 productores, emprendedores y artesanos, quienes tienen sus predios productivos en diferentes localidades del Gran La Plata y Brandsen, Alejandro Korn, Punta Indio, Loma Verde, Bavio y Magdalena. En este sentido, el PESyS se constituyó como un espacio de comercialización y visibilización de la ESS en la región mencionada.

En el 2020, con el contexto de aislamiento social a partir de la pandemia de COVID-19, se complicó naturalmente el desarrollo económico de estos productores locales, ya que dependían directamente de las ventas presenciales. Desde el Consejo Social se trabajó para acompañar a los trabajadores en la generación de nuevos canales de venta acordes a la nueva realidad, adaptándose al contexto y dando así inicio a la comercializadora "El Paseo te lo lleva", la cual se enfoca en las necesidades y desarrollo del sector productivo local.

De esta forma, la comercializadora tuvo un primer acercamiento a la digitalización, ya que comenzó a organizar la toma de pedidos mediante mensajes de texto con los consumidores a través de la aplicación WhatsApp, con un perfil corporativo. Antes de implementar esta tecnología, los pedidos se coordinaban con los consumidores a través de formularios de Google, en donde se informaba las características de los productos disponibles para la venta y se recibía la cantidad pedida por los consumidores. Estos formularios eran difundidos a la comunidad a través de las redes sociales. Sin embargo, este primer acercamiento a la digitalización trajo aparejado nuevos inconvenientes, los cuales aceleraron la necesidad de contar con un portal de comercialización.

Entonces, en pocas palabras, "El Paseo te lo lleva" es un espacio de convivencia e intercambio de bienes y saberes, donde los productores de alimentos —frescos y saludables— y de artesanías ofrecen sus productos sin intermediarios, satisfaciendo las necesidades del consumidor a un precio justo. Es una unión de productores, emprendedores y artesanos autogestionados y democráticamente organizados que desarrollan actividades de producción y comercialización como forma de vida. No es solo una feria para comercializar

sus productos, es más bien un lugar de fortalecimiento y construcción de un nuevo paradigma, que implica discusiones, talleres, capacitaciones y demás [3].

Además, trabajan sobre la base de ciertos valores compartidos que orientan sus pensamientos, actitudes y comportamientos. Estos valores, descritos a continuación, dan vida a la comercializadora y la hacen un lugar especial de trabajo y pertenencia:

- Impulsar la economía local, social y solidaria, abrazando el concepto de dignidad, cooperación y complementación, sosteniendo que todos los trabajos y capacidades son valiosos.
- Establecer un precio justo, reconociendo en este a las necesidades del consumidor, el valor de los insumos y el trabajo digno del productor, y teniendo en cuenta el correcto equilibrio entre el esfuerzo por producir algo y el dinero que el consumidor puede pagar por ello.
- Visualizar y mejorar la calidad de vida de los pequeños productores locales.
- Impulsar el consumo responsable y consciente, siendo respetuosos con el ambiente. Se busca evaluar y disminuir cualquier impacto negativo tanto en la elaboración de los productos, su conservación, comercialización y disposición final de residuos.
- Brindar seguridad a los consumidores con productos de calidad.
- Promover una venta directa del productor al consumidor e intercambio de prácticas y saberes, eliminando intermediarios y fortaleciendo la confianza entre los mismos. Se da lugar a la interacción, acercamiento y reconocimiento entre unos y otros, generando un espacio de socialización que representa a la feria y define la participación en la misma. Esto permite construir un vínculo principalmente entre consumidores y productores, permitiendo acceder a la información de producción y consumo, fortaleciendo así los lazos sociales y el consumo consciente.
- Se sostiene y fortalece a sectores desfavorecidos de la economía frente a aquellos agentes que tienden a la concentración de la distribución, de la comercialización o de ambas instancias del sistema productivo.

Actualmente, la comercializadora cuenta con dos canales de comercialización simultáneos. Por un lado, el canal presencial encabezado por la feria que se realiza semanalmente en la entrada del edificio de la Presidencia de la UNLP, la cual reúne consumidores y productores para concretar las ventas de los productos. Por otro lado, el canal virtual vía WhatsApp, el cual permite recibir pedidos de los consumidores y coordinar las entregas a domicilio o en los distintos puntos de entrega disponibles. El trabajo realizado en esta tesina se enfoca en impulsar y mejorar el canal de comercialización virtual a partir del desarrollo de un sistema de software que resuelva la comercialización online de productos y el manejo administrativo de los pedidos, productos y productores que forman parte de la feria.

A partir de esta forma de trabajo es que El Paseo se enfrenta a diferentes desafíos que dificultan y obstaculizan el día a día de los trabajadores, los cuales se mencionarán a continuación.

En primer lugar, la modalidad actual de toma de pedidos de forma virtual causa diversas consecuencias negativas tanto para los trabajadores locales como para los consumidores, ya que la gestión de pedidos que llegan por esta vía sigue realizándose de forma manual, lo que aumenta el riesgo de errores en los cálculos de montos y en el armado de los mismos, afectando directamente la experiencia de los involucrados.

En segundo lugar, la ausencia de una digitalización de los datos de los consumidores complica la identificación de patrones de compra, la segmentación por zonas geográficas, y la administración eficiente de los envíos de los pedidos. Esta limitación dificulta la toma de decisiones, ya que entre otras cosas impide reconocer a los consumidores más frecuentes, identificar en qué otras áreas geográficas se podría fomentar la participación de consumidores, o simplemente poder administrar y priorizar los pedidos de acuerdo a las zonas en dónde deben entregarse. Esta carencia de datos principalmente limita la capacidad de la comercializadora para administrar eficientemente los recursos humanos.

Por otro lado, los recursos humanos que forman parte de la mediación de la UNLP con la comercializadora a partir del Consejo Social, quienes se dedican a administrar la mercadería disponible para comercializar, se ven totalmente perjudicados, ya que su trabajo se sobrecarga fácilmente a partir de la falta de tareas automatizadas. Entre las tareas más repetitivas y tediosas se encuentran las que involucran una actualización de información que puede llegar a variar todos los días, por ejemplo: el stock de los productos, su precio o descripción, las fechas de comienzo, fin y entrega de las rondas

de ventas, o las direcciones de los puntos de venta habilitados para retirar la mercadería comprada por los consumidores. Siguiendo la misma línea, al tener que confirmar con los distintos productores que esta información sea la correcta a la hora de concretar una venta, el tiempo de la misma se extiende y corre riesgo su conclusión, afectando la experiencia del consumidor y generando una gran presión y sobrecarga de trabajo para los trabajadores administrativos involucrados.

Además, en muchas ocasiones a la comercializadora se le dificulta adaptarse a los cambios sociales y a las nuevas necesidades de los consumidores. En determinadas épocas del año es esencial que la comercializadora pueda ofrecer sus famosos "combos", por ejemplo, los combos navideños, escolares o de frutas de verano. En estos casos, teniendo en cuenta que la mayoría de las tareas que se deben llevar a cabo para ofrecer un nuevo producto son manuales, los recursos humanos deben trabajar más para ofrecer estas oportunidades de compra a los consumidores en tiempo y forma y evitar así perder oportunidades de venta.

A lo mencionado anteriormente se suma que una vez realizado un pedido por parte de un consumidor, naturalmente puede surgir su deseo de hacer un cambio en la mercadería solicitada o directamente arrepentirse de la compra. Este tipo de solicitudes se vuelven difíciles de resolver por la falta de una base de datos centralizada en donde la actualización de información no sea una complicación.

La situación se va haciendo más compleja a medida que los consumidores crecen en cantidad, pero al mismo tiempo, el hecho de que la comercializadora cada vez pise más fuerte en el comercio local es un objetivo que beneficia a gran parte de la comunidad y las familias locales, por lo que resulta imperativo cambiar su realidad para poder satisfacer y llegar cada vez a más consumidores, cuidando también la calidad de trabajo del personal.

En conclusión, es evidente que las herramientas digitales que se usan actualmente terminan siendo precarias e ineficientes. La falta de digitalización presenta no solo obstáculos operativos, sino que también limita la capacidad para optimizar los procesos internos y ofrecer experiencias más satisfactorias a los consumidores y trabajadores.

En el primer semestre de la materia "Java y aplicaciones avanzadas sobre internet" del ciclo lectivo del año 2023 de la carrera Licenciatura en Sistemas, se nos permitió a los estudiantes acercarnos a la comercializadora y conocer su problemática. Al finalizar la asignatura, resultó muy motivante entender que con los conocimientos y las herramientas adquiridas tanto en la materia

como a lo largo de toda la carrera universitaria es posible involucrarse en una problemática social y resolver grandes desafíos que tienen un gran impacto en la comunidad de la que somos parte.

Como estudiantes informáticos entendemos que muchos de los problemas actuales que afronta la comercializadora podrían solucionarse a partir de un proceso de digitalización que tenga como objetivo producir un sistema de software dedicado a la comercialización online de productos. Con un portal de comercialización online, la comercializadora resolvería gran parte de las complicaciones que hoy debe sobrellevar. El portal no solo resolvería la oferta de los productos a los consumidores, sino también toda la parte administrativa que hoy está basada en tareas manuales que pasarían a ser automáticas. Además, el portal cumpliría un rol fundamental en la construcción de lazos entre productores y consumidores, ya que será un lugar en donde se podrá publicar noticias actualizadas de loas familias productoras, sus historias, información sobre prácticas saludables y recetas, etc., manteniendo e impulsando la involucración social que tanto identifica a la comercializadora.

Con la realización de esta tesina se busca darle continuidad al trabajo comenzado en la materia, llevando adelante el desarrollo, la digitalización y puesta en producción de un portal de comercialización web para "El Paseo te lo lleva".

Cabe aclarar que no es la primera vez que los alumnos informáticos se involucran en un proyecto de innovación tecnológica para una comercializadora universitaria local de la ESS. En el año 2022, los licenciados José Francisco Blanco e Iván Gabriel Gorosito presentaron en su proyecto de tesina el desarrollo de un portal de comercialización online para la comercializadora universitaria La Justa, obteniendo resultados muy satisfactorios para la misma. La experiencia que tuvieron los compañeros, la cual ampliaremos más adelante en este informe, nos ayudó a comprender y convencernos de que lograr un impacto social a partir de las herramientas aprendidas a lo largo de la carrera universitaria es posible.

Así es que al involucrarnos en esta problemática social, continuamos lo comenzado por estos alumnos, y buscamos también seguir fomentando a que los profesionales del área de informática intervengan en estas problemáticas locales. De esta forma se busca favorecer el desarrollo socioeconómico regional y entender que las tecnologías digitales —estudiadas a lo largo de toda la carrera universitaria— hoy constituyen un instrumento fundamental y un punto de inflexión en el desarrollo de procesos socioeconómicos de inclusión social en la comunidad de la que somos parte.

## 1.2. Objetivo

Como se describió previamente, "El Paseo te lo lleva" es un espacio concebido a partir de la organización de familias productoras en articulación con la UNLP, constituyéndose en una evidencia de que una economía con rostro humano es posible y que la misma favorece la vida digna de los productores, garantizando intercambios valiosos con la comunidad. Desde este proyecto de tesina, se logró colaborar y trabajar junto con la comercializadora, y de esta manera nos alineamos con esta visión.

El objetivo principal de esta tesina es desarrollar e implementar un portal web que logre un impacto positivo en la comercializadora, en las familias productoras, en la experiencia de los consumidores y en la comunidad local en general. Para ello se busca proporcionar una herramienta moderna que logre resolver los principales inconvenientes anteriormente mencionados, con el propósito de:

- Colaborar en una organización eficaz de la comercializadora, facilitando el trabajo diario y la logística al automatizar las tareas que son actualmente manuales y requieren de mucho esfuerzo del personal humano.
- Optimizar la gestión de ventas para mejorar directamente la experiencia tanto de los consumidores como de los productores, emprendedores y artesanos que ofrecen sus productos.
- Optimizar la gestión de los recursos humanos disponibles, al disminuir el tiempo dedicado en la realización de tareas repetitivas que hoy deben realizarse diariamente, y que pasarán a ser automatizadas. Además, se le dará la posibilidad a los productores de actualizar la información de sus productos ofrecidos, lo que alivianará el trabajo de los administradores del portal.
- Adaptarse rápidamente a las necesidades de los consumidores al tener la posibilidad de actualizar información de productos, familias productoras, rondas de ventas y combos con un solo clic.
- Mejorar notablemente la experiencia de los consumidores al tener un portal para consultar la disponibilidad de productos con información actualizada. Además, podrán hacer cambios en sus pedidos de mercadería de una forma clara y sin mayores complicaciones.

- Contribuir al mejor posicionamiento en el mercado mediante la implementación de un canal de venta digital que permita llegar a nuevos consumidores, informando de forma clara y dinámica las áreas geográficas en donde la comercializadora tiene cobertura para realizar envíos de mercadería a domicilio.

- Proporcionar nuevas herramientas de análisis, como la administración de stock, reportes, métricas, entre otras, que potencien la toma de decisiones basada en datos y la identificación de riesgos de forma temprana.

La implementación de estas mejoras culminaría en una significativa optimización de los tiempos y la carga laboral de los recursos humanos, quienes actualmente se ven afectados por la falta de herramientas modernas que permitan la automatización de sus tareas diarias.

Una vez que se finalizó el desarrollo del portal web, se llevaron a cabo distintas pruebas funcionales para garantizar que el producto final efectivamente resuelve las problemáticas actuales de la comercializadora. Estas pruebas incluyeron usuarios finales reales asignados a los distintos roles disponibles en el sistema: productores, consumidores y administradores.

Por otro lado, el desarrollo del portal está basado en tecnologías libres y abiertas, y se publicó en un repositorio de software libre, favoreciendo la colaboración con otras iniciativas de la Economía Social y Solidaria, y buscando también incentivar la participación de futuros tesistas en realizar un trabajo similar dedicado a resolver los inconvenientes de otra comercializadora apoyada por la UNLP, tales como “Manos de Tierra”, “La Veredita” o “Pueblo a Pueblo”.

## 1.3. Resultados esperados

Se propone desarrollar un portal web de comercio electrónico online que se incorpore a la dinámica de trabajo de la comercializadora universitaria El Paseo, revolucionando e innovando su forma de trabajo actual, permitiendo a los trabajadores locales publicar sus productos e interactuar con los consumidores para llevar a cabo las ventas. Además, los perfiles administrativos tendrán la capacidad de actualizar la información que estará disponible en el sistema siempre que lo requieran, fomentando siempre que los consumidores puedan acceder a información actualizada.

El portal podrá ser accedido por distintos tipos de usuarios: consumidores, productores y administradores. Cada usuario podrá realizar distintas operaciones en el sistema, de acuerdo a sus privilegios y permisos.

Comprenderá las siguientes tecnologías, las cuales se analizarán en detalle más adelante en este informe:

**Frontend:** Desarrollado en JavaScript usando la extensión de TypeScript para mejorar la calidad, mantenibilidad y escalabilidad del código al proporcionar un sistema de tipado estático. Además, se usará el framework Angular para construir la aplicación web de forma robusta y de una sola página.

**Backend:** Desarrollado en Java usando el framework Jersey para la creación de servicios web RESTful documentados con Swagger. Como tecnología de Mapeo Objeto-Relacional se utilizará Java Persistence API implementado con Hibernate. Además, se usará el framework HK2 para la inyección de dependencias.

**Base de datos:** Se usará una base de datos relacional MySQL para una estructura de datos organizada.

Tanto el frontend como el backend se disponibilizarán en un repositorio público de software.

En conclusión, al finalizar la tesina, el portal web desarrollado y puesto en producción ofrecerá:

**A los administradores:** Eliminar, crear, actualizar y listar todas las entidades que viven en el sistema. Podrán administrar las ventas, usuarios, generar reportes útiles (por ejemplo, sobre las estadísticas de venta, stock de productos, etc.), y visualizar métricas de negocio. Además, podrán actualizar toda la información disponible en el sistema automáticamente, con el objetivo de evitar la difusión de información desactualizada a los consumidores.

**A los productores:** Publicar sus productos en venta y realizar mantenimiento de la información de los mismos, actualizando stock, precio, descripción, imagen, etc. para lograr ofrecer información al día a los consumidores, y además alivianar el trabajo de los usuarios administrativos.

**A los consumidores:** Realizar órdenes de compra optando entre la modalidad "entrega a domicilio" o "retiro por punto". Además, accederán a su historial de compras realizadas, podrán hacer modificaciones si lo desean, y administrar su información personal.

# Capítulo 2

# Trabajo Relacionado

## 2.1. Comercializadora Universitaria La Justa

Como mencionamos anteriormente, a partir de la pandemia de COVID-19 que se decretó el 20 de marzo de 2020 y de las medidadas de Aislamiento Social Preventivo y Obligatorio (ASPO)[1], las ferias universitarias presenciales debieron cerrar sus puertas inmediatamente. Debido a esto, los trabajadores locales perdieron rápidamente la posibilidad de comercializar sus productos, afectando directamente la estabilidad económica de familias enteras.

En respuesta a esta problemática social, el Equipo de Dirección de Economía Popular, Social y Solidaria de la Prosecretaría de Políticas Sociales junto con la Prosecretaría de Agricultura Familiar de la Facultad de Ciencias Veterinarias de la UNLP, comenzaron a trabajar en conjunto con el objetivo de encontrar una salida. En este contexto es que se dio inicio a la comercializadora de Intermediación Solidaria La Justa[2].

Una comercializadora de Intermediación Solidaria se encarga de mediar sin objeto de lucro entre la producción y el consumo local. Además, se dedica a la distribución de los productos, acercándolos a los consumidores bajo un precio justo, y establece criterios cooperativos y de producción sostenible, tanto social como ambiental. Los criterios cooperativos no solo abarcan a las tareas de producción, sino a toda la organización interna.

En términos generales, La Justa es una comercializadora de la ESS que

[1] https://www.boletinoficial.gob.ar/detalleAviso/primera/227042/20200320

[2] Véase https://www.facebook.com/La-Justa-comercializadora-105707981222282 y https://www.instagram.com/lajusta.comercializadora

desde la UNLP y en red con organizaciones sociales, comunitarias, políticas y culturales, promociona una compra de proximidad de alimentos y otros bienes elaborados artesanalmente por productores de la economía popular y la agricultura familiar, intermediando solidariamente entre el consumo y la producción local.

La comercializadora involucra unas 200 familias productoras, organizadas en torno a 15 organizaciones. El equipo de trabajo lo conforman unas 40 personas que se organizan en distintas áreas: acompañamiento técnico a productores, comunicación, logística y coordinación general. La comercializadora tiene una red de consumidores en continuo crecimiento: durante 2020 se lograron vender 86 toneladas de alimentos y en 2021, 61 toneladas.

Antes de adoptar el portal de comercialización, resultado de la tesina de grado de Lic. en Informática de Francisco Blanco e Iván Gorosito[4], su esquema de trabajo se iniciaba con la difusión y promoción vía redes sociales de la oferta de alimentos disponibles por parte de las familias productoras; luego se recepcionaban y confirmaban los pedidos y por último se distribuían y entregaban en los 13 nodos de retiro de la red (localizados en distintos puntos estratégicos de la ciudad de La Plata y de Berisso).

Cabe destacar que, al igual que la comercializadora El Paseo, en La Justa también tuvieron un primer acercamiento a la digitalización a partir del uso de Formularios de Google para resolver ciertas tareas. En un principio se utilizaron para la comercialización digital: como explican en el Proyecto de Tesina "Portal de comercialización online para la Economía Social y Solidaria Local"[3], *"Se diseñó un formulario en el que volcaron los principales productos, con sus características, precio y fotos, y selección de retiro en un nodo (en un principio 4, pero hacia el segundo semestre de 2020 ya eran 10), que resuelve provisoriamente la construcción de la oferta (mostrar los productos disponibles de las familias productoras) y la demanda (selección de productos y nodos por parte de los consumidores)"*[4].

A pesar de que este uso de la tecnología permitió resolver rápidamente una parte de las tareas necesarias para concretar la toma de pedidos, la realidad es que a su vez trajo aparejado múltiples inconvenientes para el trabajo interno en la comercializadora, ya que, entre otras cosas:

- No permitía informar ni actualizar automáticamente información de stock de los productos disponibles. Esto hacía que los días de apertura

[3] Véase https://sedici.unlp.edu.ar/handle/10915/143765

del formulario para tomar pedidos gran parte del equipo de La Justa debía organizarse en turnos y dedicarse a controlar que los pedidos no superen los stocks disponibles, y "quitar" la opción de compra de un producto cuando se agotaba la disponibilidad. En conclusión, se volvía una tarea totalmente manual y, por lo tanto, propensa a errores humanos.

- El formulario no era cien por ciento confiable, ya que podía ocurrir que algunos pedidos no se vean impactados correctamente en la base de datos provista por el formulario de Google. Para detectar estos casos a tiempo, otro grupo de trabajo de la comercializadora debía dedicar varios días a la semana a tomar la planilla de datos resultante que se obtiene del formulario y confirmar uno por uno los pedidos por WhatsApp con cada cliente. Nuevamente, una tarea que debía ser automática se convertía en algo manual y exhaustivo.

Es evidente que las cuestiones mencionadas consumían una gran cantidad de tiempo de trabajo del equipo de La Justa que se quitaba al acompañamiento técnico y organizativo de los grupos de productores y otras tareas relacionadas con la mejora de la calidad del trabajo y de vida de los mismos. En ese momento, antes de contar con un portal web de venta online, la comercializadora se enfrentaba a distintos desafíos en relación con mejorar los procesos de logística y distribución, la comunicación con consumidores y lograr fortalecer y establecer nuevas alianzas con comercializadoras y productores, que permitan ampliar el volumen y los canales de venta.

En el año 2020, en la Facultad de Informática de la UNLP, la propuesta de tesina de grado de José Francisco Blanco e Iván Gabriel Gorosito consistió en investigar los desafíos a los que se enfrentaba La Justa ante la falta de digitalización y desarrollar y poner en producción un sistema de software que abordara y solucionara estos desafíos, con el objetivo de dar soporte y crecimiento a las experiencias colectivas de comercialización local, fomentando así el concepto de ESS.

Como conclusión, el desarrollo y puesta en producción del portal web significó un punto de inflexión para el funcionamiento de la comercializadora La Justa. Actualmente, el portal web[4] no es solo una herramienta que resuelve tareas de organización y logística de pedidos, sino que es un medio para seguir fomentando el fortalecimiento de los lazos entre los mercados solidarios

[4]Página oficial: `https://www.lajustaunlp.com.ar/shop/home`

y la comunidad. Logra llegar más allá de la comercialización, abarcando otras dimensiones que son parte del proyecto, como la promoción de otras formas de producción, intercambio y consumo, o la difusión de prácticas alimentarias y saberes saludables. Así es que las herramientas que ofrece el portal no se limitan a la venta de productos, sino que también acortan distancia entre productores y consumidores: se cuentan las historias detrás de quienes producen los alimentos y artesanías, y se difunde contenido de consumo consciente y responsable como recetas o videos sobre la producción con el objetivo de concientizar acerca de la agricultura familiar, la agroecología y la soberanía alimentaria.

## 2.2. Experiencia previa con un portal web para El Paseo

El Paseo tuvo la oportunidad de participar en un proyecto de desarrollo de software para crear su portal de digitalización en línea, gracias a una persona cercana a la comercializadora que estaba buscando un cliente para su proyecto de tesis para la universidad de Seattle en Estados Unidos.

Aunque hasta ese momento no habían considerado la posibilidad de digitalizar sus procesos debido a otras prioridades, decidieron aceptar la propuesta con la esperanza de automatizar tareas que se volvían cada vez más complejas y tediosas de realizar manualmente.

El proceso de desarrollo de software involucró reuniones y comunicaciones con los desarrolladores del sistema. Si bien el sistema web se desarrolló y se puso en producción, lamentablemente, después de unos meses de uso, se tuvo que dar de baja por diversos motivos.

Al consultarles sobre este antecedente al personal de El Paseo, nos comentaron que el problema principal fue que el portal desarrollado no logró resolver completamente ni llevar adelante los procesos necesarios para que la comercializadora funcione correctamente. El personal administrativo tuvo que continuar con tareas manuales simultáneamente con el uso del portal, ya que el sistema no abordaba todas las necesidades. Esta combinación de tareas automatizadas y manuales generó más complicaciones de las que El Paseo enfrentaba previamente. Por ejemplo, aunque el portal recopilaba información sobre los productos pedidos por los consumidores, no registraba la cantidad de cada producto solicitado. Por lo tanto, cada vez que se recibía un

pedido, era necesario contactar telefónicamente al consumidor para confirmar la cantidad solicitada, realizando el chequeo y actualización del stock de los productos de forma manual. Esto resultó muy inconveniente para los administradores y afectó negativamente la experiencia tanto de los consumidores como del personal de El Paseo.

Desde la perspectiva de un desarrollador de software, es evidente que el análisis de requisitos falló en el proceso de desarrollo. Los desarrolladores no lograron comprender las necesidades del cliente, lo que llevó a una interpretación errónea de lo que debía abordar el sistema. Como resultado, las funcionalidades del portal no resolvieron los problemas de la comercializadora y generaron nuevos inconvenientes.

Además, los desarrolladores utilizaron fondos personales para publicar el portal web, lo que hizo que, una vez finalizado el proyecto de tesis, decidieran no continuar asumiendo esos gastos, que además eran en dólares. Estas complicaciones llevaron a que la comercializadora diera de baja el portal web y retomara al cien por ciento las tareas manuales.

En nuestro proyecto de tesina, tomamos en cuenta esta experiencia previa de El Paseo con el objetivo de identificar qué fue lo que salió mal, aprender de los errores y trabajar para que no vuelvan a ocurrir.

Al aplicar y poner en práctica las herramientas de desarrollo de software aprendidas a lo largo de la carrera universitaria se buscó respetar todas las instancias del proceso de construcción de software. Entender el problema y las expectativas de los adoptantes del sistema, en este caso El Paseo, para evitar el error de concluir con un producto que no resuelve sus necesidades, fue una prioridad en este proyecto de tesina de grado. Además, en nuestro proceso contamos con distintas instancias de retroalimentación con los usuarios, en las cuáles se presentaron los avances del portal web y recibimos opiniones, para rápidamente darnos cuenta de si lo desarrollado era de utilidad, o si era necesario modificar el trabajo realizado para adaptarlo a la problemática que se debía resolver.

# Capítulo 3

# Economía Social y Solidaria

La ESS es un enfoque económico que pone énfasis en la solidaridad, la participación democrática, la sostenibilidad y el bienestar colectivo, priorizando el trabajo sobre el capital y al ser humano sobre el dinero. Se basa en la idea de que la economía debe estar al servicio de las personas y las comunidades, no solo de la maximización del beneficio[5].

Abarca una variedad de iniciativas como cooperativas, asociaciones, empresas sociales y otras formas de organización económica que buscan equidad, inclusión y sostenibilidad. Asimismo, sus prácticas fomentan la diversidad cultural, la armonía, la valoración de la naturaleza, la dignificación del trabajo, la justicia social, la ayuda mutua y en consecuencia el sentido de comunidad.

Se entiende como la economía de los trabajadores, de los que viven o quieren vivir de sus saberes y trabajo, la economía de sus familias, comunidades, asociaciones, redes y organizaciones, de los que tienen recursos materiales acumulados limitados, que dependen fundamentalmente de la continua realización de su fuerza de trabajo para sobrevivir y sostener proyectos de vida digna. Donde el crecimiento es condición, pero no fin, ya que la ESS se juzgará por su estructura y calidad social y ecológica, y no por su valor monetario [5].

## 3.1. Un poco de historia

Históricamente, el concepto de ESS surge como una alternativa al modelo planteado durante décadas de capitalismo. Esta corriente culmina con una

identificación de economía hegemónica basada en un sistema de mercados, sosteniendo la idea de que el mismo naturalmente se maneja a partir de la oferta y la demanda de mercancías, sujeto a sus propias leyes sin regulaciones.

Esta desigualdad intrínseca en el sistema se ve afectada a partir de las experiencias de "éxito" o "fracaso" que se impusieron históricamente en el juego del libre mercado. Además, en toda sociedad naturalmente hay tendencias a la discriminación y la desvalorización de los que no pueden trabajar, o se dedican a trabajos mal juzgados "inferiores". No todos los sectores sociales demandan los mismos trabajos, por lo que su reconocimiento por unos puede ir acompañado por la indiferencia o desprecio por otros.

El capitalismo propone un sistema basado en la meritocracia[1], pero sin contemplar las desigualdades, discriminación y estigmatización que pueden sufrir los individuos. Quedan totalmente excluidos y desprotegidos quienes no lograron insertarse exitosamente en el mercado, generalmente a partir de no contar con las mismas oportunidades de desarrollo debido a su condición social. Es así que muchos autores concluyen que el libre mercado genera mucha desigualdad, y a partir de la misma se llega a generar desigualdad extrema y marginalidad, fragmentando a la sociedad.

Para José Luis Coraggio, la clase trabajadora queda fragmentada en cuatro segmentos socioeconómicos con ingresos, educaciones y condiciones de vida en general diferenciadas[5]:

- Los que trabajan en relaciones de dependencia, que tienen un trabajo asalariado formal, con una plataforma lograda de derechos sociales que el Estado debe garantizar y una perspectiva de ascenso social.
- Los que trabajan en relación de dependencia pero de manera intermitente o precaria, con derechos incompletos.
- Los que trabajan de manera autónoma, autogestionada (individual, familiar, comunitaria o libremente asociada) con derechos parciales no plenamente establecidos.
- Los pobres e indigentes, los que quedan excluidos, desempleados o que nunca tuvieron un trabajo, cuyos derechos se reducen a la asistencia pública.

[1]Definido por la Real Academia Española como un sistema de gobierno en que los puestos de responsabilidad se adjudican en función de los méritos personales

Teniendo en cuenta lo mencionado anteriormente, se puede concluir que quienes participan de estas últimas categorías tienen baja posibilidad de poder ascender en la escala social por sus propios medios.

En las comunidades organizadas y con cosciencia social, surge un proyecto político que busca generar una alternativa a lo impuesto por el capitalismo y el libre mercado, construyendo una economía social y solidaria por y para la sociedad, también llamada la "otra" economía. Para el desarrollo de la ESS resulta esencial la construcción de mercados solidarios autogestionados, en los cuales la solidaridad es un eje central, y en donde sus participantes puedan impulsar una economía alternativa a la economía hegemónica que permita la mejora continua de la calidad de vida de las personas que conforman estos circuitos, a partir de intercambios que se rigen por prácticas y valores solidarios [6].

Hay que tener en cuenta que este nuevo paradigma no busca excluir los intereses individuales ni las relaciones mercantiles, sino que las busca alinear al principio ético de que todos puedan convivir sin exclusiones ni desigualdades extremas, pudiendo vivir y trabajar dignamente. Teniendo esto en cuenta, el desarrollo de una ESS olvida el objetivo de maximizar las ganancias y se concentra en asegurar la reproducción y el desarrollo de la vida digna de los participantes de una comunidad. La solidaridad se basa en el reconocimiento de los otros seres humanos y la responsabilidad en los intercambios con la naturaleza, reconociendo las necesidades de todos los involucrados y contribuyendo a la resolución de las mismas.

En consecuencia, la ESS plantea relaciones sociales de producción e intercambio basadas en la no explotación del trabajo ajeno, el intercambio justo, la reciprocidad, la competencia cooperativa y el reconocimiento del otro como un par, no necesariamente renunciando a los intereses personales legítimos [5]. Además, su forma de gestión es autónoma y democrática.

## 3.2. La Universidad y la ESS

El concepto de ESS es divulgado a la sociedad a partir del accionar de distintos actores que se involucran en las problemáticas sociales. La UNLP no es la excepción, ya que a partir de sus decisiones y proyectos logra integrarse con la comunidad de la que es parte, conociendo sus problemas y trabajando activamente para resolverlos. Desde hace más de una década que la UNLP acompaña este esquema de comercialización, buscando principalmente am-

pliar canales de venta de los productores locales, y así mejorar su estructura de ingresos y, por lo tanto, su calidad de vida.

En la reforma estatutaria de 2008, la UNLP deja en claro su postura, compromiso y responsabilidad en el preámbulo del estatuto [2]:

*"(la Extensión), debatida y consensuada con el conjunto de la comunidad, perseguirá contribuir a la búsqueda de respuestas a problemas sociales, fundamentalmente de aquellos sectores más vulnerables por no tener sus derechos esenciales garantizados. La Extensión Universitaria será el principal medio de la Universidad Nacional de La Plata para lograr su función social, contribuyendo al tratamiento de los problemas que afectan al bienestar de la comunidad, la reconstrucción del tejido social, el desarrollo económico sustentable y el fortalecimiento de la identidad cultural"*

En el Capítulo III, en su artículo 17, establece con claridad *"la Universidad reconoce como una de sus funciones primordiales la extensión universitaria, entendida como un proceso educativo no formal de doble vía, planificada de acuerdo a intereses y necesidades de la sociedad, cuyos propósitos deben contribuir a la solución de las más diversas problemáticas sociales, la toma de decisiones y la formación de opinión, con el objeto de generar conocimiento a través de un proceso de integración con el medio y contribuir al desarrollo social"*.

Siguiendo esta línea, el 28 de septiembre de 2010 se creó el Consejo Social con el objetivo de planificar acciones para el desarrollo productivo y la recuperación de los derechos esenciales del conjunto de la población, sin mediar propósito de lucro. Se convocó a organismos públicos, actores gubernamentales, sindicatos y movimientos sociales, sumando también a representantes de facultades y colegios universitarios. Desde el comienzo se pensó como un espacio de articulación concreta entre la Universidad y la Comunidad para dar respuestas a las principales problemáticas sociales de la región conformada por los municipios de La Plata, Berisso, Ensenada, Magdalena, Brandsen y Punta Indio.

Los fines establecidos para el Consejo Social son: reunir a todos los actores de la región para analizar las principales problemáticas socioeconómicas, políticas, culturales y ambientales y discutir conjuntamente posibles estrategias de abordaje mediante políticas locales y nacionales. Uno de sus primeros temas de trabajo fue la ESS, a partir de que se constituyera en 2011 la Comi-

[2]Estatuto de la UNLP. Disponible en https://unlp.edu.ar/gobierno/estatuto_unlp-4287

sión de Economía Popular, Social y Solidaria (en adelante EPSyS) como un espacio de trabajo para el desarrollo de proyectos y actividades destinados a fortalecer este subsistema económico en la región.

En términos generales, se puede plantear que la EPSyS incluye aquellas actividades cuyos integrantes se organizan para resolver sus necesidades de manera autogestiva y trabajando por un proyecto transformador, equitativo, justo y sostenible para toda la sociedad [7]. Cooperativas, fábricas recuperadas, unidades productivas y de servicios surgidos de políticas públicas, feriantes, artesanos, agricultores familiares, están por lo general involucradas en la EPSyS.

En el partido de La Plata hay una gran presencia de organizaciones de la agricultura familiar, familias productoras, artesanos y emprendedores que se fueron agrupando en diferentes organizaciones sociales para hacer frente a las problemáticas sociales que afectan su desarrollo económico y subsistencia. Así surge la iniciativa de la UNLP denominada "El Paseo de la Economía Social y Solidaria" como una experiencia de comercialización y visibilización de la ESS, dando inicio a los reconocidos "mercados solidarios".

## 3.3. Conociendo a El Paseo

Como describimos anteriormente, El Paseo surge por la iniciativa de productores, emprendedores y artesanos democráticamente organizados, apoyados y acompañados por la UNLP. Tiene una estructura organizativa horizontal entre quienes la integran, vertical entre sus proveedores de materias primas y los consumidores, y en diagonal con la UNLP [1].

Su objetivo es entender y abordar las problemáticas sociales que afectan el desarrollo económico y la permanencia en el mercado de los trabajadores locales.

Antes de la pandemia de COVID-19, El Paseo rotaba semanalmente entre diversos espacios de la ciudad de La Plata, donde llevaba a cabo su comercialización a partir de ferias itinerantes, siendo el principal sitio el ingreso al edificio de la Presidencia de la UNLP.

A partir del decreto de Aislamiento Social y Preventivo[3] dispuesto en marzo de 2020, se suspendieron las ventas presenciales y la atención de los distintos predios. Esto afectó directamente a los trabajadores locales, quienes

[3] https://www.boletinoficial.gob.ar/detalleAviso/primera/227042/20200320

dependían de las ventas que realizaban de forma presencial. Ante esta problemática, el Consejo Social de la UNLP trabajó para adecuarse al contexto y acompañar a los afectados, proponiendo un nuevo canal de venta a través de medios virtuales.

Esta medida, que significó el primer acercamiento digital de El Paseo, implicó que se reorganizara la metodología de comercialización, ahora basada en nodos para la venta y distribución, buscando recrear la feria desde Internet y sosteniendo las redes de intercambio construidas junto a la comunidad local. Además, se amplió el personal a cargo de la logística, ya que naturalmente surgieron nuevas tareas a resolver: toma de pedidos a través de formularios online de Google (con su apertura, cierre y edición), publicidad, atención a las redes sociales, control de pedidos y stock, armado de pedidos y entregas, seguimiento de los mismos, pago a los productores, atención a reclamos y consultas, atención al medio de comunicación de WhatsApp, etc. A su vez, se amplió el número de repartidores de pedidos, lo que permitió cubrir además de la ciudad de La Plata demandas de pobladores de los municipios aledaños de Berisso y Ensenada. Toda esta reorganización permitió que los pedidos se incrementaran considerablemente, llegando a ser entre 200 y 300 por semana.

Con el paso del tiempo, El Paseo fue ampliando la oferta de productos e incluyendo "combos" ajustados a distintas necesidades. Así es que en la actualidad, los pedidos realizados por los consumidores pueden incluir: bolsones de verdura y fruta, productos agroecológicos de otras provincias, lácteos, miel, conservas y mermeladas, productos panificados, pastas, productos veganos y sin TACC, productos de limpieza, de indumentaria, entre otros.

Las instituciones participantes de la feria son actualmente 23 organizaciones de familias productoras, emprendedores y artesanos, entre las que se destacan las pertenecientes a la agricultura familiar, donde como mínimo cada organización está integrada por aproximadamente 20 familias. Entre ellas podemos mencionar la Cooperativa Nueva Esperanza, la Cooperativa Moto Méndez, el Grupo Lapacho), y el grupo Bienestar en tus manos, que pertenece a la organización Unión de los Trabajadores de la Tierra (UTT). Cabe destacar que los integrantes de estas instituciones son familias, que mantienen la cultura del trabajo en la zona como forma de vida y a lo largo de toda su historia familiar. Por otro lado, las organizaciones del rubro de elaborados, textil, reciclados y artesanías son de centros, talleres comunitarios o comedores, como El Refugio, Las Mirabal, Barrios del Plata, Taller Puerta Verde, Cooperativa de Trabajo Ayni Consumo consciente, el Movi-

miento Patria Grande de la CTA (Central de Trabajadores de la Argentina), Organización feriantes de Parque Alberti y la agrupación artesanos de Plaza Italia, donde sus integrantes son entre 10 y 20 emprendedores. Por lo expuesto hasta aquí podemos concluir que los participantes directos de la feria El Paseo son alrededor de 110 productores, artesanos y emprendedores provenientes de distintos puntos de la ciudad de La Plata y alrededores. Además, las decisiones tomadas por El Paseo afectan directamente a las familias de los trabajadores mencionados anteriormente[8].

Por otro lado, los consumidores de los productos ofrecidos no son solo ciudadanos de La Plata y alrededores, sino que también gran cantidad de bares y comercios aledaños que compran y consumen productos de El Paseo.

Como estrategia para seguir fomentando lazos con la comunidad se puede mencionar que El Paseo permanentemente trabaja para generar y promover actividades culturales, talleres y capacitaciones constantes. Sus integrantes pretenden ser proveedores de bienes de consumo, bienes culturales y promover los valores propios del modelo de ESS. Además, El Paseo busca ser un lugar donde se puedan establecer vínculos horizontales sin privilegios relativos a la posición jerárquica, generar sentidos y enriquecerse como personas.

A lo largo de estos años de experiencia, El Paseo ha ido desarrollándose y desempeñándose en distintos ámbitos, mencionados a continuación [1]:

**Cultural:** Generó a lo largo de los años de experiencia nuevos valores de equidad, justicia y reciprocidad, que han reforzado la comunidad.

**Social:** Las organizaciones que la integran han fortalecido su identidad y a su vez aportaron un mayor compromiso con el espacio colectivo.

**Económico:** El dinero que ingresa a la comercializadora recircula entre las organizaciones que la componen, generando intercambios de bienes y servicios entre los pares y con el territorio que la sostiene, evitando su captación y centralización por fuera del mismo.

**Político:** Trabajando en su visión y misión, reforzando el vínculo con los consumidores y con el resto de los actores del territorio.

**Ambiental:** Capacitándose para adquirir herramientas que apuesten a una producción más saludable, ética y solidaria, apostando al consumo responsable y consciente.

Actualmente, El Paseo es un punto de encuentro entre productores y consumidores a partir de formas de organización de la actividad económica anclada en los principios y valores de la ESS, buscando diferentes estrategias para llegar a diferentes sectores de la comunidad con sus productos, con mejor calidad y precio justo.

## 3.4. La necesidad de digitalización de El Paseo

Es evidente que a pesar de los esfuerzos para impulsar el desarrollo digital de la comercializadora, actualmente siguen existiendo ciertos inconvenientes que complican el fortalecimiento y el crecimiento de la misma.

Como se mencionó previamente en este informe, existen problemas operativos a los que se enfrentan los trabajadores diariamente. Los trabajadores administrativos deben llevar a cabo tareas repetitivas, lo que da lugar a errores humanos que impactan directamente en la experiencia de los involucrados. Además, la rotación de los trabajadores administrativos dificulta y ralentiza la curva de aprendizaje necesaria para llevar adelante los procesos, ya que las tareas que deben realizar no son intuitivas por lo que siempre un nuevo integrante necesita de una supervisación de otro integrante experimentado.

Por otro lado, dado que la comercializadora promueve la inclusión social ofreciendo productos para todas las personas, cada vez son más los consumidores que buscan ser parte de este proyecto colaborativo. Esto es algo superpositivo, pero también trae aparejado varios inconvenientes de organización y logística. El hecho de que el número de consumidores crezca sobrecarga el trabajo de los recursos humanos a la hora de atender sus consultas o resolver reclamos. Las herramientas con las que cuentan para este tipo de tareas son muy escasas y precarias, lo que hace que cada vez que haya que atender un caso se deba hacer una "investigación" para determinar cuál es el pedido que realizó el consumidor y el inconveniente en cuestión. Esto dificulta directamente la comunicación fluida con los consumidores, lo que afecta la capacidad de proporcionar información oportuna y resolver consultas de manera rápida. Se podría concluir que si no se logra una digitalización de las tareas que se realizan para llevar adelante la comercialización, va a llegar un punto en donde los recursos humanos estén tan sobrecargados que se vuelva casi imposible satisfacer todas las demandas de los consumidores.

La mayor problemática se genera por la falta de una base de datos que reúna la información de los productores, rondas, puntos de retiro, productos y consumidores. Esto hace que a la hora de sacar conclusiones o tomar decisiones sea difícil reunir la información necesaria, ya que todo queda volcado en herramientas no digitales. Asimismo, a la hora de divulgar la información sobre el ingreso de un producto o productor nuevo, o de hacer alguna promoción, no hay una herramienta eficiente para hacerlo, ya que no queda más que divulgarlo a través de las redes sociales disponibles. Esto puede generar que gran parte de los consumidores no accedan a esta información, afectando la capacidad de comunicar sus valores, y perdiendo una gran cantidad de oportunidades de venta.

Con respecto a la administración de los pedidos, al no tener una asociación directa de consumidores con localidades genera que a la hora de organizar los pedidos a entregar por zonas, esto se vuelva una ardua tarea, y que muchas veces no se logre una asignación eficiente de pedidos a los repartidores, haciendo que la coordinación de la distribución se complique. También se suman los errores ocasionados por la falta de herramientas digitales que dificultan el seguimiento preciso y mantenimiento de inventario, lo que puede llevar a problemas de exceso o escasez de productos.

La sumatoria de estas experiencias hace que en muchos casos los consumidores vivan experiencias no muy satisfactorias, lo cual no ocurriría si se contara con herramientas adecuadas y modernas para atenderlos en tiempo y forma.

En un mundo cada vez más digitalizado, no adaptarse a las herramientas modernas puede generar que la comercializadora quede rezagada, perdiendo el lugar que hoy en día logró ocupar en el mercado local. En este contexto es que los integrantes de El Paseo plantearon la necesidad de desarrollar un portal web para la comercialización online de la ESS, que permita resolver las problemáticas mencionadas anteriormente, y así mejorar no solo las experiencias de los consumidores, sino también la calidad del trabajo de los trabajadores de la comercializadora.

# Capítulo 4

# Ingeniería de Software: desarrollo del portal de comercialización online para El Paseo

Según Ian Sommerville, considerado uno de los padres de la ingeniería de software, la ingeniería de software *"es una disciplina de la ingeniería que comprende todos los aspectos de la producción del software, desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de este después de que se utiliza"* [9]. Además, indica que *"un proceso de software es un conjunto de actividades y resultados que producen un producto de software"* [9], y como actividades fundamentales menciona a la especificación, el desarrollo, la validación y la evolución del software.

La ingeniería de software es de suma importancia, ya que naturalmente los clientes que necesitan el software para resolver sus inconvenientes lo reclaman lo más pronto posible. Es por esto que existe la tentación entre los desarrolladores de comenzar a codificar sin antes hacer un buen análisis de los requerimientos del cliente. Las distintas experiencias como desarrolladores de software permiten concluir que por lo general estos casos terminan con que el sistema desarrollado no cumple con las necesidades o no alcanza las expectativas. Lo más probable es que el tiempo invertido para que el sistema funcione haya sido mayor al que se hubiera invertido aplicando ingeniería de software, y respetando sus etapas conscientemente. Esto concluye con los desarrolladores exhaustos y frustrados, y con los clientes difícilmente

satisfechos.

La definición y evolución de las distintas etapas necesarias para desarrollar el portal de comercialización de El Paseo comenzó en la cursada de la asignatura "Java y Aplicaciones Avanzadas sobre Internet"[1] de la carrera Licenciatura en Sistemas de la UNLP.

Al comienzo de la cursada, se nos presentó a los alumnos las dificultades que experimentaba la comercializadora que dificultaban tanto su trabajo diario y como su evolución. Además, se nos brindaron las herramientas tecnológicas y explicaciones necesarias para sentar las bases de un portal web que resuelva la venta online y ciertos problemas administrativos, acotando el alcance del problema para ajustarse a los tiempos de cursada de la asignatura.

Luego, a lo largo de la cursada, desde la cátedra se planteó una dinámica de trabajo que consistía en ir desarrollando el sistema en distintas etapas organizadas con sus entregas (evaluaciones) periódicas, las cuales se explicarán más adelante en este informe. Entre ellas podemos identificar el análisis de requerimientos, diseño del software, implementación y codificación, y documentación.

A partir de esta dinámica de trabajo, se nos permitió a los alumnos desde un primer momento contar con reuniones frecuentes y contacto directo con el equipo de El Paseo, con el objetivo de obtener la información necesaria para evolucionar en el desarrollo del portal web.

Al comienzo de la cursada hubo una primera reunión donde miembros de la comercializadora presentaron la problemática y describieron su forma actual de trabajo. Luego, nos mantuvimos en contacto para seguir adentrándonos en el tema y realizar consultas siempre que sea necesario. Una vez que finalizó la cursada de la asignatura me decidí a continuar con el desarrollo del portal como proyecto de tesina y para ello se retomaron las reuniones con el personal de El Paseo. Estas reuniones se dedicaron por un lado a profundizar en la situación actual de El Paseo, y a realizar demostraciones de los avances del portal (entregables), buscando resolver consultas y confirmando que el desarrollo realizado cumplía funcionalmente y se alineaba con las expectativas del cliente.

[1] Plan de estudios: https://www.info.unlp.edu.ar/wp-content/uploads/2019/05/Java-y-Aplicaciones-Avanzadas-en-Internet.pdf

## 4.1. Entender el problema: Análisis de Requerimientos

La Ingeniería de Requerimientos (en adelante IR) es una pieza fundamental y sirve como una base sólida en el proceso de desarrollo de software. Marca el punto de partida para actividades como la planeación, la definición de recursos necesarios y la elaboración de cronogramas como principal mecanismo de control durante la etapa de desarrollo. Además, es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto.

Los requerimientos se pueden definir como una descripción detallada de las necesidades de los clientes o usuarios del sistema, y es contra lo que se va a estar verificando si se están cumpliendo las metas trazadas.

En el libro "Fundamentos de Ingeniería de Software" [10] los autores ofrecen como definición de requerimiento la siguiente: *"Los requerimientos especifican qué es lo que un sistema de software debe hacer (sus funciones) y sus propiedades esenciales y deseables. Un requerimiento expresa el propósito del sistema sin considerar cómo se va a implantar. En otras palabras, los requerimientos identifican qué hace el sistema, mientras que el diseño establece el cómo lo hace"*. Además, se especifica para qué sirven los mismos: *"Primero, permiten que los desarrolladores expliquen cómo han entendido lo que el cliente espera del sistema. Segundo, indican a los desarrolladores qué funcionalidad y qué características debe tener el sistema resultante. Tercero, indican qué demostraciones se deben llevar a cabo para convencer al cliente de que el sistema que se le entrega es de hecho lo que había ordenado."* [10]

Por otra parte, el glosario de la IEEE[2] define a un requerimiento como *"una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo"* y *"una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal"*. Además, plantea que los requerimientos de software pueden dividirse en 2 categorías: requerimientos funcionales y requerimientos no funcionales.

**Funcionales:** Definen las funciones que el sistema será capaz de realizar y describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Es importante que se describa el ¿Qué? y no

[2]Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, Institute of Electrical and Electronics Engineers). Véase `https://www.ieee.org/`

el ¿Cómo? se deben hacer esas transformaciones. Estos requerimientos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema

**No funcionales:** Tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc. Es decir, estos requerimientos ponen límites y restricciones al sistema.

A partir de una mala identificación y administración de los requerimientos se puede arribar a que el proyecto de software fracase por no realizar una adecuada definición y especificación de los mismos. Dentro de la mala administración se pueden encontrar factores como la falta de participación del usuario o cliente, requerimientos incompletos y la mala adaptación al cambio.

La IR se enfoca entonces en la definición de lo que se desea producir: se busca generar especificaciones correctas que describan claramente y sin ambigüedades las necesidades de los usuarios, minimizando los problemas relacionados con la mala gestión de requerimientos. En otras palabras, es el proceso de recopilar, analizar y verificar las necesidades de los usuarios [11].

Siguiendo este lineamiento, en esta etapa analizamos nuestro producto de software a desarrollar para El Paseo, teniendo en cuenta las necesidades y expectativas del cliente, y distinguimos los distintos requerimientos funcionales y no funcionales. A modo de ejemplificación, podemos mencionar a grandes rasgos los siguientes:

**Funcionales:**

Registro de usuarios: Los usuarios consumidores deben poder registrarse en el portal proporcionando su información personal como nombre, correo electrónico, contraseña, entre otros datos obligatorios. A partir del registro exitoso, estos usuarios pueden ingresar al sistema a realizar sus compras.

Carrito de compras: Los usuarios consumidores deben poder agregar productos al carrito de compras, ver los productos presentes en el mismo, modificarlos y realizar la compra.

Registrar nuevo producto: Los usuarios administradores del sistema deben poder dar de alta nuevos productos registrando sus características, y estos productos deben quedar disponibles para que luego los usuarios consumidores puedan comprarlos.

**No funcionales:**

Seguridad: El portal debe garantizar la seguridad de la información del usuario, utilizando cifrado SSL para la transmisión de datos y almacenando las contraseñas de forma segura utilizando, por ejemplo, técnicas de hash.

Usabilidad: El portal debe ser fácil de usar y de navegar para los usuarios, con una interfaz intuitiva y amigable que facilite la búsqueda y compra de productos.

Cabe destacar que para lograr una correcta especificación de requerimientos es necesario que los mismos se puedan probar y que sean concisos, completos, consistentes, y no presentar ambigüedades.

La IR dispone de dos medios de comunicación para recoger y transmitir requerimientos:

**Comunicaciones escritas:** Registran la información de forma permanente, son más fáciles de compartir con grupos y personal remoto, se pueden pensar y revisar de forma completa y son más propensas a malinterpretaciones.

**Comunicaciones verbales:** Permiten recibir feedback de inmediato, son dinámicas, se adaptan con facilidad a nuevos desarrollos, permiten generar ideas nuevas, y alcanzar comprensión y claridad comunes con menos esfuerzo.

En el contexto de la IR, las historias de usuario se usan como una herramienta ágil de comunicación que combina las fortalezas de ambos medios: escrito y verbal. Son en sí requerimientos, ya que expresan el problema que el sistema o producto de software debe resolver. Describen (en una o dos frases) una funcionalidad de software desde el punto de vista del usuario que desea dicha funcionalidad, generalmente siguiendo la estructura: *Como (Tipo de Usuario), necesito (algún objetivo) para poder (alguna razón)*. El foco está puesto en qué necesidades o problemas soluciona lo que se va a construir[12].

En nuestro caso, se buscó llevar a cabo esta etapa de captura y análisis de requerimientos a partir del desarrollo de historias de usuario y diagramas de clases, ya que estas herramientas agilizan la administración de los requerimientos al evitar dedicar tiempo en las formalidades que implican otros documentos técnicos escritos. Para ser exactos, una vez realizadas las historias de usuario, se realizó una primera versión del diagrama de clases.

En pocas palabras, un diagrama de clases es una herramienta de modelado en la ingeniería de software que representa la estructura estática y las relaciones entre las clases dentro de un sistema. Las clases son modelos de abstracción que representan entidades con características y comportamientos comunes. Por lo general, los diagramas de clase se definen en la etapa "Modelado del Software" de la Ingeniería de Software, pero en este proyecto fue fundamental desarrollar los primeros diagramas junto con el análisis de requerimientos, ya que permitió entender mejor la estructura que iba a tener el sistema, y así lograr una base más sólida.

A continuación dejamos a modo de ejemplo la historia de usuario definida para la funcionalidad "Ver listado de productos disponibles para comprar como usuario consumidor". El resto de las historias de usuario especificadas, se pueden encontrar siguiendo este link.

**Nombre de la historia de usuario:**
Ver listado de productos disponibles para comprar como usuario consumidor
**Descripción:**
Como usuario consumidor, quiero poder ver el listado de productos disponibles en el portal web para poder decidir y seleccionar qué productos comprar.
**Criterios de aceptación:**

1. Debe haber un botón o enlace claramente identificable en la interfaz del usuario para acceder al listado de productos.
2. Al hacer click en el botón o enlace, el sistema debe mostrar un listado de los productos disponibles, incluyendo su nombre, descripción, precio, productor asociado y una imagen.
3. El listado de productos debe ser fácil de navegar y buscar, con filtros, opciones de ordenamiento y paginado.
4. El sistema debe mostrar la disponibilidad de cada producto y permitir agregarlos al carrito de compras en caso de que haya stock.

Durante esta etapa, entonces, se trabajó en relación con los stakeholders[3] para comprender sus necesidades y expectativas. Nos encontramos con las típicas dificultades asociadas a esta tarea: los requerimientos no son obvios y provienen de distintas fuentes, son difíciles de expresar en palabras, cambian a lo largo del ciclo de desarrollo, y además esta tarea se complica, ya que el vocabulario de los usuarios no es el mismo al de los desarrolladores.

A pesar de las dificultades, se lograron extraer las características operacionales del software y sus restricciones, llegando a reconocer los elementos básicos del sistema a desarrollar lo más cercano a como lo percibía el usuario.

Al concluir esta etapa del desarrollo del software ya contábamos con una idea completa del problema que queríamos resolver con la realización de nuestro portal web, las expectativas de los usuarios y los distintos tipos de usua-

[3]La palabra "stakeholders" es un término inglés que se refiere a todas las partes interesadas o involucradas en un proyecto, empresa, organización o cualquier situación en particular. Los stakeholders son individuos, grupos o entidades que pueden afectar o ser afectados por las acciones, decisiones o resultados de una empresa u organización.

rios (consumidores, administradores y productores), y las clases de objetos que iban a coexistir en nuestro sistema. Esta comprensión fue fundamental para posteriormente poder tomar decisiones relacionadas con las tecnologías a utilizar.

## 4.2. Modelado del software

La etapa de modelado del software es una fase dentro del ciclo de vida del desarrollo de software en la cual se construyen representaciones abstractas y visuales del sistema que se está desarrollando. Estos modelos ayudan a desarrolladores, diseñadores y stakeholders a comprender, visualizar y especificar diferentes aspectos del software antes de su implementación, logrando validar ideas y tomar decisiones informadas antes de pasar a la fase de implementación.

Un modelo es una representación de un concepto, de un objeto o de un sistema, que permite explicar algunas propiedades específicas. Por ejemplo: la forma, el comportamiento, la estructura o la relación de una entidad con otras entidades[10].

Se utilizan diferentes tipos de modelos, como:

**Modelos de requerimientos:** Representaciones de los requerimientos del usuario, funcionalidades y comportamientos esperados del sistema.

**Modelos de diseño:** Diagramas y representaciones que detallan la arquitectura del software, la interacción entre sus componentes, las relaciones entre módulos o clases, entre otros.

**Modelos de datos:** Diagramas de entidad-relación, diagramas de clases, entre otros, que describen la estructura y relaciones de la información que será manejada por el software.

**Modelos de comportamiento:** Diagramas de casos de uso, diagramas de secuencia, diagramas de actividad, que muestran cómo interactúan los elementos del sistema en diferentes escenarios y situaciones.

Entonces, modelamos para comprender mejor el sistema que estamos desarrollando: para visualizar lo que queremos que haga, especificar su estructura y comportamiento, y guiar su construcción. Además, modelar ayuda a

comunicarse con el cliente o usuario, a explorar posibles soluciones a un problema durante el diseño del sistema, y a documentar las decisiones tomadas durante el desarrollo.

Durante la etapa de modelado del software, los desarrolladores y diseñadores utilizan UML[4] como una herramienta visual para crear una variedad de diagramas que representan diferentes aspectos del sistema en desarrollo, con el objetivo de mejorar la interpretación y el entendimiento. Algunos de los más utilizados son:

**Diagrama de casos de uso:** Describe las interacciones entre usuarios y el sistema, mostrando los diferentes escenarios de uso del software.

**Diagrama de clases:** Representa las clases del sistema, sus atributos, métodos y relaciones, ayudando a visualizar la estructura estática del software.

**Diagrama de secuencia:** Muestra la secuencia de interacciones entre los distintos objetos o componentes del sistema, enfocándose en cómo se comunican y realizan actividades en un escenario específico.

**Diagrama de actividad:** Describe el flujo de actividades o procesos dentro del sistema, mostrando el comportamiento dinámico de un proceso o función.

**Diagrama de estado:** Representa los distintos estados por los que pasa un objeto durante su ciclo de vida, mostrando cómo responde a eventos específicos.

En este proyecto, los modelos que más se utilizaron fueron modelos de datos, de diseño y de comportamiento. Primero se desarrolló un diagrama de clases 4.1 para clarificar la estructura estática del sistema orientado a objetos al modelar las clases del sistema, sus atributos, métodos básicos y las relaciones entre ellas. Luego continuamos con el desarrollo de un diagrama de entidad-relación (el cual explicaremos más en detalle en la sección 4.3) para modelar la estructura lógica de la información y las relaciones entre las entidades, representando cómo los datos se almacenan y relacionan en la base de datos. Y una vez que tuvimos más en claro las características del

[4]Lenguaje de Modelado Unificado. UML por sus siglas en inglés: Unified Modelling Language

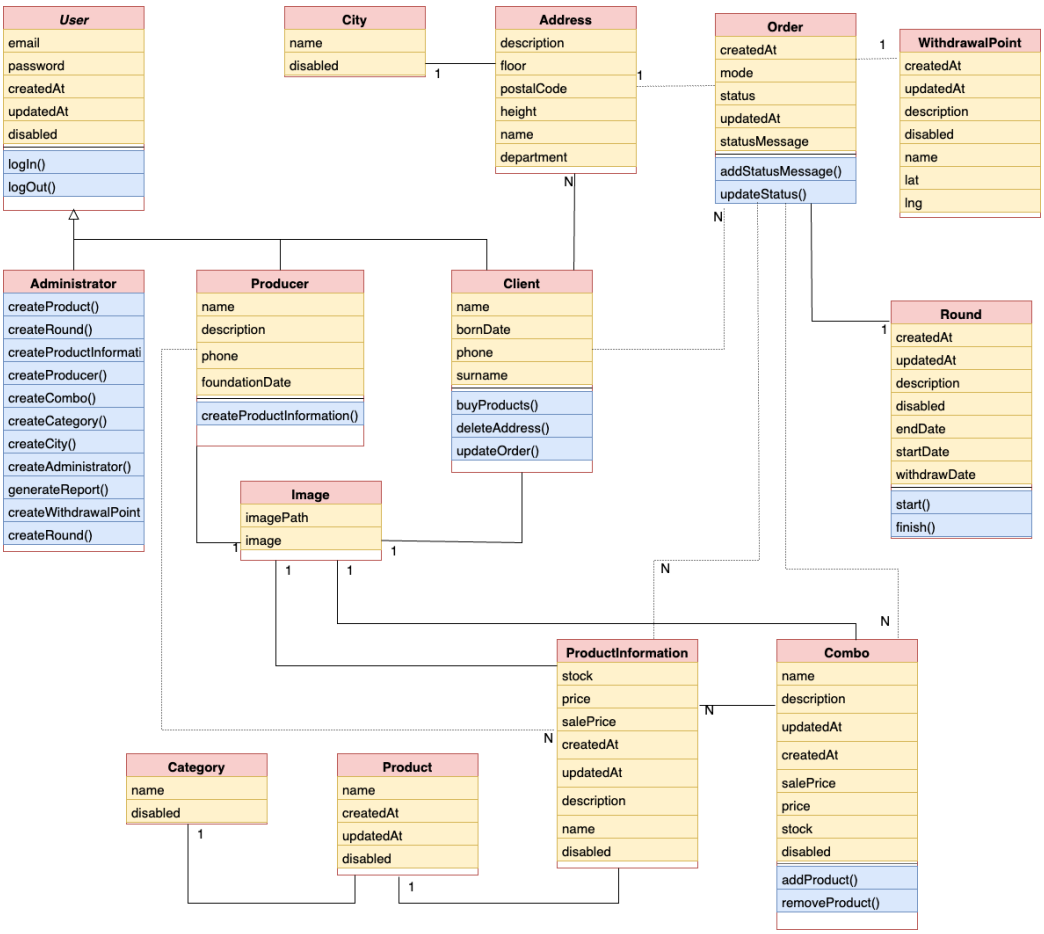


Figura 4.1: Diagrama de clases desarrollado en la etapa inicial de modelado

sistema, continuamos con el desarrollo de diagramas de secuencia para lograr entender las funcionalidades más complejas.

Los diagramas de secuencia se utilizan para representar la interacción entre objetos en un sistema a lo largo del tiempo, mostrando cómo los distintos objetos o componentes colaboran entre sí para realizar una acción o completar un proceso específico, enfocándose en el orden temporal de los mensajes intercambiados entre ellos. Cabe destacar que no todas las funcionalidades del sistema fueron modeladas a través de estos diagramas, sino solo las que se consideraron más complejas y difíciles de entender.

A modo de ejemplo, en la figura 4.2 se puede ver el diagrama de secuencia

realizado para ilustrar la funcionalidad de registrarse en el sistema. El consumidor es un actor, ya que es un participante externo al sistema que interactúa con él. Por otro lado, identificamos a los objetos "Cliente", "Servidor"y "Base de datos", los cuales interactúan para resolver el requerimiento, como se describe a continuación:

1. El usuario consumidor ingresa a la página principal de El Paseo, y selecciona la opción "Registrarme".

2. El cliente web le solicita al servidor la información necesaria para poder mostrar el formulario de registración. En este caso, se necesita el listado de ciudades disponibles, lo cual se ve representado con el llamado al endpoint **GET /api/v1/ciudades**

3. El servidor al recibir la petición, la resuelve solicitando a la base de datos las ciudades disponibles, y devolviendo el resultado al cliente, con el código HTTP 200 OK para indicar que la petición se resolvió correctamente.

4. Ante la respuesta exitosa, el cliente ya cuenta con la información necesaria para armar el formulario de registración. Una vez construido, lo muestra en la pantalla "Registrarme", en donde las ciudades obtenidas se mostrarán como un selector para que el consumidor elija a qué ciudad pertenece.

5. El usuario consumidor completa el formulario con los datos solicitados, y presiona el botón "Registrarme".

6. Esto dispara la ejecución de una función en el cliente para validar que los datos que son requeridos estén completos, representada en el diagrama con el nombre *validarCamposRequeridos()*. En caso de que los campos requeridos estén incompletos, se devuelve al usuario un error con el código HTTP 400 BAD REQUEST, representando que la solicitud es incorrecta, e indicándole al usuario con un mensaje descriptivo cuáles son los campos mandatorios que debe completar. En caso de que los campos requeridos estén completos, se procede al siguiente punto.

7. Se valida la edad ingresada por el consumidor a partir de la ejecución de la función *validarEdad()* del cliente. De la misma forma, en caso de que el usuario no sea mayor de 18 años, se devuelve una respuesta

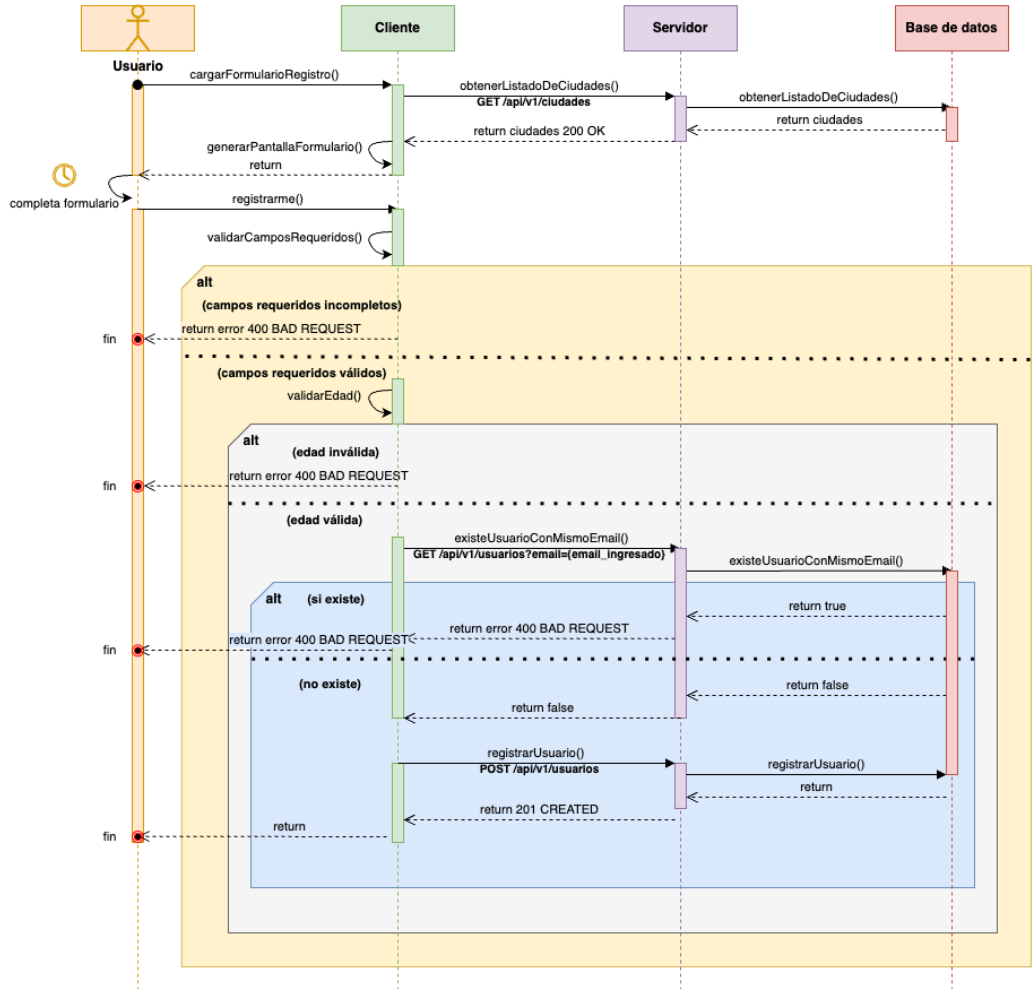


Figura 4.2: Diagrama de secuencia realizado para la funcionalidad "Registrarse en el sistema"

con el código HTTP 400 BAD REQUEST y un mensaje descriptivo indicando la razón del error. En caso de que los campos requeridos estén completos, se procede al siguiente punto.

8. Luego de que todas las validaciones de los datos sean correctas, se procede a chequear si ya existe un usuario registrado en el sistema con el mail ingresado. Para esto, el cliente realiza una petición al servidor al endpoint **GET /api/v1/usuarios?email=email_ingresado**. Para resolver la petición, el servidor realiza una consulta a la base de datos filtrando por email en la tabla de usuarios. Si se encuentra una fila como resultado de la consulta, significa que ya existe un usuario con ese email, por lo que se devuelve al cliente un error con el código HTTP 400 BAD REQUEST, y el cliente le informa al usuario la razón por la que no se puede realizar el registro. Si no existe un usuario con el mismo email, se procede al siguiente punto.

9. El cliente envía una petición al endpoint **POST /api/v1/usuarios** para crear el usuario, y retorna el código HTTP 201 CREATED para informar que la entidad se creó correctamente. El cliente, por su parte, le informa al usuario que el registro fue exitoso.

Es importante señalar que en el diagrama de secuencia no se consideraron deliberadamente los escenarios de error para simplificar la representación. No obstante, el sistema aborda cualquier solicitud que genere un error, ya sea al interactuar con el cliente, el servidor o la base de datos. En tales situaciones, se proporcionará un mensaje adecuado al usuario final para mantenerlo informado sobre el estado actual del sistema en todo momento.

## 4.3. Diseño del software

En la etapa del diseño del software, los requerimientos y especificaciones recopilados en etapas anteriores se utilizan para crear un plan detallado sobre cómo se construirá el sistema. Se determina la estructura del sistema de software y de sus datos antes de iniciar la etapa de codificación e implementación.

El objetivo principal de esta etapa es producir un diseño detallado y completo que sirva como guía para los desarrolladores durante la implementación.

Un buen diseño asegura que el sistema cumpla con la especificación de los requerimientos, sea eficiente, fácil de mantener y escalable.

Además, suele ser iterativo, lo que significa que se revisa y se ajusta según sea necesario a medida que se descubren nuevos detalles o se hacen cambios en los requerimientos del sistema. Esto permite mantener la flexibilidad y adaptabilidad del proceso de desarrollo.

Como mencionamos anteriormente, durante el análisis de requerimientos se responde a las preguntas ¿Qué? y ¿Cuál?, mientras que en el diseño del sistema de software se responde a la pregunta ¿Cómo? Se toman decisiones acerca de la forma en que se resolverá el problema, primero desde un nivel elevado de abstracción y luego empleando cada vez niveles de refinamiento que muestren más detalles.

La fase de diseño del sistema de software toma como base los resultados del análisis de los requerimientos, los cuales deben proporcionar los elementos claves para determinar la estructura del software. Durante el diseño del sistema también se decide la arquitectura del mismo, el diseño global, el diseño de los componentes, el diseño de las interfaces de los componentes y el diseño de los datos[10].

Se desea que un sistema de software sea **confiable**, es decir, con baja probabilidad de fallas; **robusto**, es decir, que se comporte razonablemente en circunstancias que no fueron anticipadas en los requerimientos; y **eficiente**, o sea, que haga buen uso de los recursos. Por otra parte, se considera que un diseño es bueno cuando es **simple** y **comprensible**; cuando es **flexible**, es decir que se pueda adaptar a otras circunstancias agregando o modificando módulos; y cuando se puede **reutilizar en otros sistemas** [10].

En esta etapa plasmamos nuestro diseño de base de datos a través de un diagrama de entidad-relación, a partir del cual se modela el mundo real como un conjunto de objetos básicos llamados entidades y las relaciones existentes entre ellas. En la figura 4.3 mostramos el primer diagrama realizado, pero cabe destacar que a lo largo del desarrollo el mismo fue sufriendo modificaciones para adaptarse a nuevas problemáticas que se iban encontrando.

Tal y como se muestra en el diagrama, el sistema de El Paseo engloba a tres categorías de usuarios: CLIENTS (consumidores), PRODUCERS (productores de mercaderías) y ADMINISTRATORS (administradores del sistema, personal administrativo de El Paseo). La información elemental de estos usuarios, como su correo electrónico y contraseña, está consolidada en la entidad USERS. Esto se hace con la intención de poder reutilizar la información básica y las funcionalidades de los usuarios en los diferentes roles

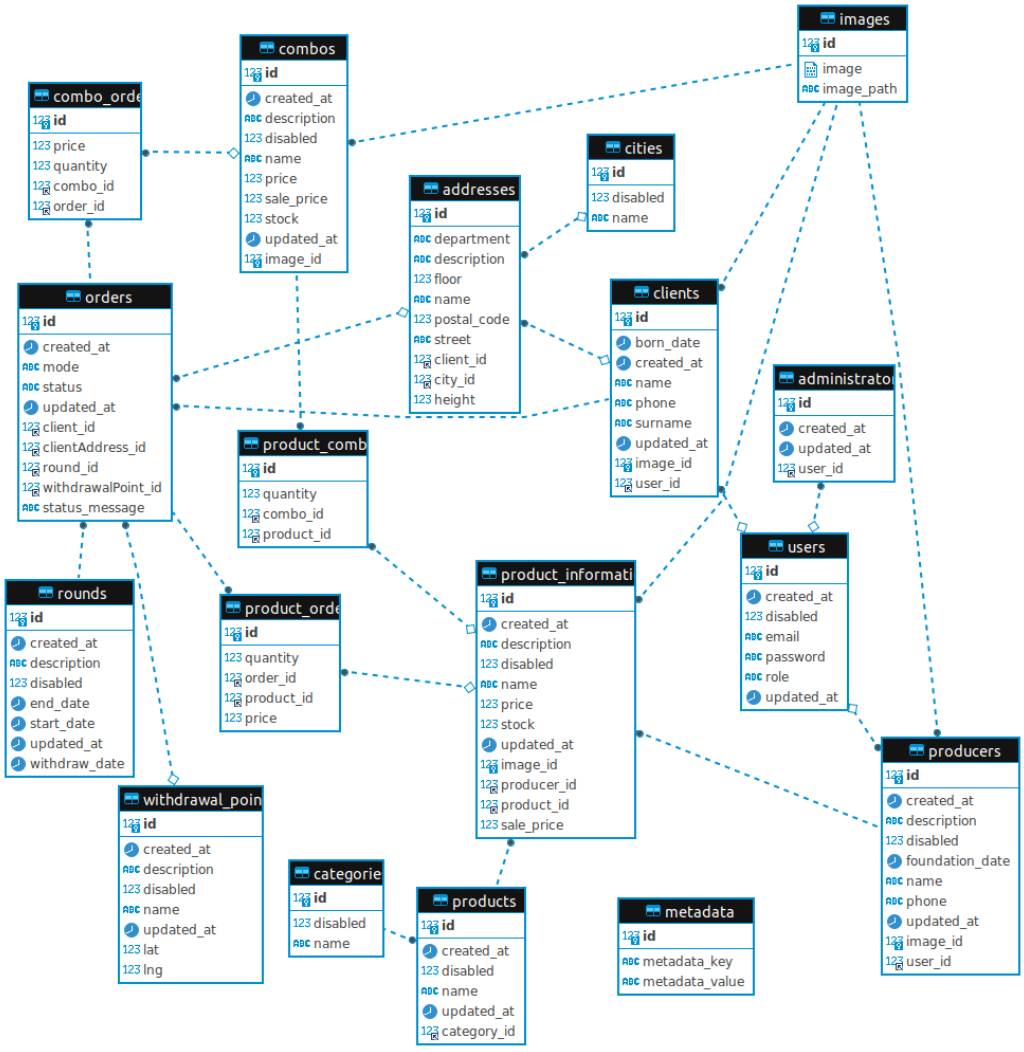


Figura 4.3: Primer diagrama de entidad-relación realizado para el sistema de El Paseo

que desempeñan (consumidores, productores o administradores).

Para representar la mercadería a la venta ofrecida por la comercializadora diseñamos dos entidades: PRODUCT_INFORMATIONS y COMBOS:

- La entidad PRODUCT_INFORMATIONS representa un producto en venta en particular y contiene información básica del mismo: precio, stock disponible, nombre, descripción, fecha de creación y última actualización, etc. Además, está asociado a la entidad PRODUCERS para representar a quienes producen cada mercadería, y asi permitir que se pueda conocer la información completa de los productores de cada producto. Por otro lado, se relacionan con las entidades PRODUCTS y CATEGORIES para representar a qué tipo de producto corresponden y a qué categoría pertenecen. Como ejemplo de categorías se podrían mencionar "Frutas", "Verduras", "Lácteos", "Indumentaria" o "Panificados"; mientras que como ejemplos de tipos de productos se podría indicar "Tomate", "Manzana", "Queso", "Bebida". Es así que se puede considerar a la entidad PRODUCTS como una subcategorización dentro de cada entidad CATEGORIES. La necesidad de esta entidad surgió dada la naturaleza de la comercializadora, ya que por ejemplo ante la necesidad de un consumidor de comprar "manzana roja", la categorización de "Frutas" queda muy amplia e incluye una gran cantidad de otros tipos de frutas, por lo que lo ideal sería poder seleccionar primero el filtro "Frutas" (entidad CATEGORIES) y luego por el tipo de producto "Manzana" (entidad PRODUCTS). Notar que los PRODUCT_INFORMATIONS asociados al tipo de producto "Manzana" pueden ser más de uno, por ejemplo una manzana roja, una manzana verde, una manzana Ambrosia, etc.

- La entidad COMBOS representa un conjunto de productos a vender y contiene datos básicos como el nombre, descripción, fecha de creación y última actualización, stock disponible, precio, entre otros. Los productos incluidos en un combo se vinculan mediante la tabla intermedia PRODUCT_COMBO, donde también se especifica la cantidad de unidades de un producto en particular que tiene el combo.

Cada pedido de mercadería realizado por parte de los consumidores queda plasmado en la entidad ORDERS. Esta entidad reúne información como el estado de la misma (para indicar si se encuentra IN_PROGRESS, CANCE-

LLED, DELVERED, etc.), la fecha en la que se hizo el pedido, y el consumidor que lo realizó. Además, especifica el modo de la orden, el cual puede ser DELIVERY o WITHDRAWAL_POINT, con el objetivo de diferenciar las órdenes que deben entregar los repartidores a las direcciones especificadas por los consumidores, o las órdenes que serán retiradas en un punto de retiro en particular seleccionado por el consumidor. Si el modo es DELIVERY la dirección donde debe entregarse el pedido quedará guardada en la propiedad *clientAddressId*, mientas que si el modo es WITHDRAWAL_POINT el punto de retiro seleccionado quedará almacenado en la propiedad *withdrawalPointId*, el cual es una referencia a la entidad WITHDRAWAL_POINTS que representa los puntos de retiro disponibles.

Además, las rondas de pedidos habilitadas quedan modeladas en la entidad ROUNDS, indicando fecha de comienzo, fecha de fin, fecha de retiro en punto de retiro, descripción, entre otros. Cada orden realizada por un consumidor queda asociada a una ronda en particular para ser concluida dentro de esas fechas.

La entidad CLIENTS que representa a los consumidores contiene una colección de ORDERS para representar el histórico de pedidos realizados a lo largo del tiempo por un consumidor. Estas órdenes cuentan con las asociaciones correspondientes a los productos y combos comprados. Es importante resaltar que estas asociaciones se llevan a cabo a través de tablas intermedias. Con respecto a los productos comprados para una orden en particular, la tabla intermedia se llama PRODUCT_ORDER. El objetivo de esta tabla es especificar la cantidad y el precio del producto comprado en ese momento en particular. Esto permite eliminar inconsistencia de datos a medida que el precio del producto cambia: es como una "foto" de las características de un producto en un momento en particular. Con respecto a los combos, la tabla intermedia llamada PRODUCT_COMBOS cumple la misma funcionalidad que la tabla que asocia órdenes con productos, pero relacionándolos con combos.

Por otro lado, las ciudades en donde la comercializadora tiene cobertura se encuentran modeladas mediante la entidad CITIES. Cada consumidor al ingresar por primera vez al sistema elige en qué ciudad se encuentra y así queda asociada la entidad CLIENTS con la entidad CITIES, y por ende los pedidos que realicen los consumidores con modalidad DELIVERY, también quedarán asociados a una ciudad en particular donde deberá realizarse la entrega. Esta representación nos facilita el control de las áreas donde se pueden despachar productos, al tiempo que nos permite obtener informes

detallados, como la concentración de consumidores por ciudad, el volumen de ventas en cada una, entre otros datos relevantes.

Por último, la entidad METADATA es una estructura que almacena datos con formato clave-valor. Su diseño original se centró en guardar configuraciones administrativas. Por ejemplo, en la página principal de El Paseo encontramos información sobre la historia de la comercializadora, detalles de contacto, y breves descripciones. La intención es permitir al equipo administrativo modificar estos detalles sin depender de los desarrolladores para realizar cambios en el código y desplegar actualizaciones. Al modelar esta estructura, los administradores podrán acceder a un portal de configuración donde tendrán la posibilidad de actualizar cada texto a su conveniencia, evitando el ida y vuelta con los desarrolladores. Estos cambios se reflejarán en la base de datos al actualizar, por ejemplo, la entidad METADATA con clave *subtitulo_página_principal* por un nuevo valor que represente el nuevo subtítulo especificado.

Si bien es sabido que utilizar una base de datos relacional para guardar información de tipo clave-calor no es correcto, en este caso se decidió guardar ésta información en la misma base de datos por las características del proyecto. En proyectos pequeños, los recursos (tanto humanos como financieros) pueden ser limitados. Implementar y mantener otra tecnología podría ser costoso y no justificado por el tamaño del proyecto. Además, el mantenimiento y la administración de una única base de datos es más sencillo. En un futuro, y considerando el crecimiento de la comercializadora, podría considerarse contratar una tecnología acorde a esta necesidad y hacer las migraciones correspondientes.

Como conclusión, el diseño de las herramientas mencionadas fue de mucha utilidad para comprender mejor el problema y las funcionalidades que nuestro sistema debía tener para resolver los problemas identificados. Además, en esta instancia fue que optamos por utilizar una herramienta de maquetado (en este caso Excalidraw[5]), para diseñar las primeras pantallas de la interfaz de usuario.

En pocas palabras, el maquetado es el proceso de crear representaciones visuales estáticas de las interfaces que formarán parte del sistema final. Estas maquetas son diseños estáticos que muestran la disposición de elementos en la interfaz de usuario, aunque no tienen funcionalidad interactiva. Son fundamentales para comunicar visualmente la estructura y el diseño antes

[5]Véase `https://excalidraw.com/`

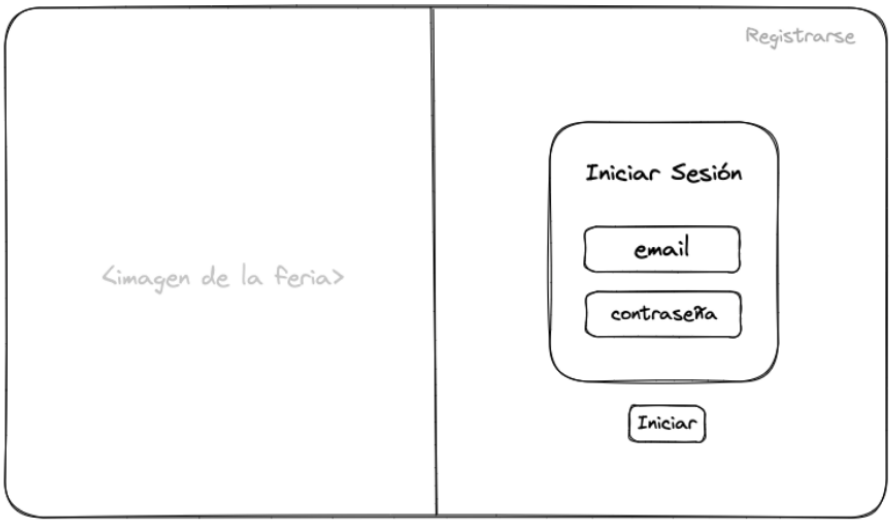


Figura 4.4: En este maquetado se puede visualizar el formulario de inicio de sesión con los datos necesarios para ingresar al sistema (email y contraseña). Además, se especifica que a la izquierda de la pantalla se podría insertar una imagen de la feria de El Paseo, y en la esquina superior derecha podría haber un botón para acceder a la página de registro.

de avanzar al desarrollo. En esta etapa, no es necesario detallar aspectos de diseño de estilos, simplemente se busca mostrar de forma clara las funciones que cada pantalla de nuestro portal web abordará. Se pueden ver ejemplos de estas primeras maquetas en las figuras 4.4 y 4.5.

En etapas posteriores, estas maquetas se transformaron en pantallas reales y para implementarlas se tuvieron en cuenta las especificaciones que nos brindaron el equipo de diseñadores de El Paseo en cuanto a colores, tipografías, logos e imágenes a utilizar.

Por otro lado, con respecto al diseño de la arquitectura del sistema, se definió que el mismo estaría basado en el modelo cliente-servidor, el cual es un enfoque de diseño de sistemas distribuidos en el que las tareas y la lógica de la aplicación se dividen entre dos tipos de entidades: el cliente y el servidor[13].

El **cliente** (también llamado *frontend*) es una aplicación o dispositivo que solicita servicios o recursos desde otro programa, llamado servidor. Es responsable de presentar la interfaz de usuario y proporcionar una experiencia interactiva para el usuario final, gestionando la entrada del usuario a través

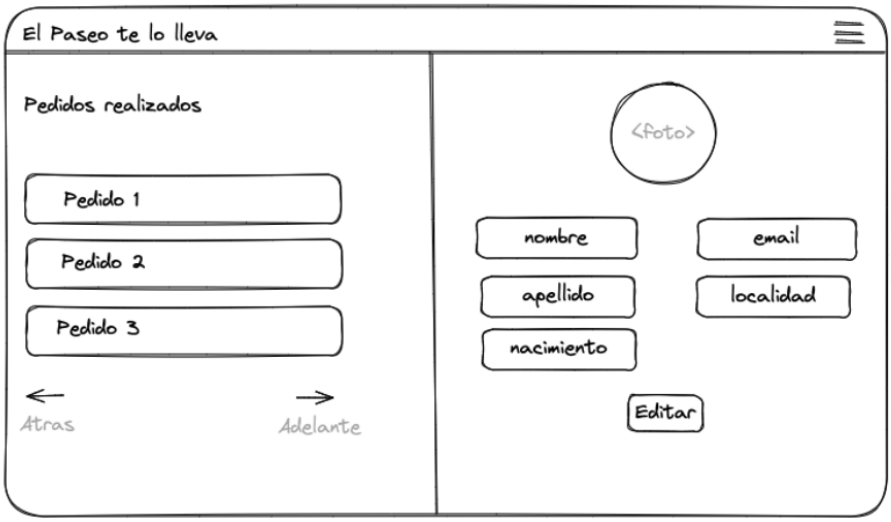


Figura 4.5: En este maquetado se puede visualizar la pantalla del perfil de un usuario consumidor. A la izquierda se muestra el historial de pedidos realizados con la posibilidad de paginarlos, y a la derecha la información personal del usuario (imagen, nombre completo, localidad y fecha de nacimiento, etc.) con la posibilidad de editar los datos.

de formularios, botones y otros elementos interactivos. Además, se encarga de mostrar datos y contenido de manera visualmente atractiva y comprensible, y realiza solicitudes al servidor para obtener o enviar datos, en nuestro caso a través de llamadas de API[6], actualizando instantáneamente el estado de la aplicación en el lado del cliente para proporcionar una experiencia fluida.

El **servidor** (también llamado *backend*) es un programa o dispositivo que proporciona servicios, recursos o datos solicitados por los clientes. Contiene la lógica del negocio y la funcionalidad principal de la aplicación. Procesa datos, aplica reglas empresariales, toma decisiones y maneja errores, realizando operaciones de lectura y escritura en la base de datos, asegurando la persistencia y recuperación de datos. Además, maneja la autenticación y autorización de usuarios, gestionando sesiones y controlando el acceso a recursos protegidos, e implementa medidas de seguridad, como validación de entrada, prevención de ataques y gestión de permisos para proteger la aplicación y los datos. En nuestro proyecto, se exponen los servicios a través de una API que es consumida por el cliente.

Esta organización favorece y fomenta la división de responsabilidades, la escalabilidad y centralización (permite escalar y distribuir la carga de trabajo, ya que múltiples clientes pueden conectarse a un servidor central o a una red de servidores para acceder a recursos compartidos), y la independencia tecnológica en la implementación de cada uno.

El diseño de arquitectura explicado anteriormente nos permite principalmente la reutilización de componentes, ya que en un futuro se podría reutilizar el mismo backend entre distintas aplicaciones web, móviles u otro sistema que necesite consumir los servicios. Esto, entre otras cosas, permite seguir fomentando la involucración de estudiantes de informática en problemáticas relacionadas con la ESS, como ocurrió con el proceso de digitalización de La Justa, ya que un grupo de alumnos desarrolló un backend con una aplicación web, mientras que otros más adelante reutilizaron el mismo backend para desarrollar la aplicación móvil de la comercializadora. Esto no sería posible si los sistemas se desarrollaban en una arquitectura monolítica, en donde todas las funcionalidades y componentes de una aplicación están interconectados y se ejecutan en un solo conjunto de código fuente y una única unidad de despliegue.

[6]API significa Ïnterfaz de Programación de Aplicaciones"(Application Programming Interface en inglés). Es un conjunto de reglas, protocolos y herramientas que permiten a diferentes sistemas informáticos comunicarse entre sí.

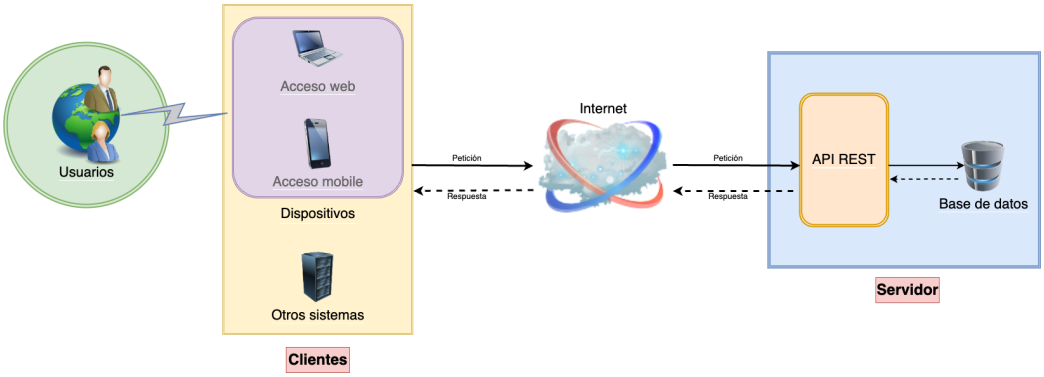


Figura 4.6: Diagrama de arquitectura para El Paseo: modelo cliente-servidor.

El diagrama de la figura 4.6 describe la arquitectura cliente-servidor adoptada para el sistema de El Paseo, ilustrando cómo es posible reutilizar el mismo backend para distintos clientes, como por ejemplo web, móvil, u otros sistemas.

Por último, la base de datos elegida es una base de datos relacional, la cual organiza los datos en tablas con filas y columnas. Este tipo de bases de datos es ampliamente utilizado en aplicaciones empresariales y sistemas donde la estructura de datos está bien definida y los requerimientos de consistencia y relaciones son importantes. Su estructura tabular y las relaciones bien definidas entre los datos hacen que sean eficientes para consultas complejas y análisis de datos.

En este proyecto, la naturaleza de los datos era ideal para usar una base de datos relacional: tienen un esquema fijo y bien definido que no cambia frecuentemente, y se podían organizar de forma natural en las tablas, filas y columnas. Además, hay dependencia de datos entre distintas entidades que se podía representar mediante relaciones (uno a uno, uno a muchos, muchos a muchos).

## 4.4. Codificación

En la etapa de codificación, los desarrolladores traducen los diseños, requisitos y especificaciones previamente establecidos en código fuente que luego será ejecutado por una computadora, a través de un lenguaje de programación en específico. Esta etapa implica escribir, probar y depurar el código

para implementar la lógica y la funcionalidad de la aplicación o sistema. Algunos de sus aspectos claves son:

**Traducción de diseños:** Los diseños y modelos previamente creados (como diagramas, prototipos, especificaciones) se convierten en código utilizando lenguajes de programación.

**Desarrollo iterativo:** Los desarrolladores escriben código en ciclos iterativos, implementando pequeñas porciones de funcionalidad a la vez. Esto permite realizar pruebas y ajustes continuos.

**Seguimiento de estándares y buenas prácticas:** Los desarrolladores siguen las convenciones de codificación, normas y buenas prácticas para asegurar la legibilidad, mantenibilidad y eficiencia del código.

**Pruebas unitarias:** Durante esta etapa, se suelen escribir pruebas unitarias para asegurar que cada componente funcione como se espera y para detectar errores lo antes posible.

**Revisión de código:** En algunos equipos, se lleva a cabo una revisión de código por parte de otros desarrolladores para garantizar la calidad y la coherencia del código.

**Depuración y optimización:** Se buscan y corrigen errores y se optimiza el código para mejorar su rendimiento y eficiencia.

**Documentación:** Se puede generar documentación que explique la funcionalidad y el funcionamiento del código para futuras referencias. Esta documentación será de mucha utilidad para otros desarrolladores que deseen o necesiten colaborar con el sistema en un futuro.

La etapa de codificación entonces es un proceso creativo y técnico donde se transforman los requisitos y diseños en un producto funcional. Es crucial para el éxito del proyecto tener un código bien escrito, organizado y probado para asegurar que la aplicación cumpla con los requisitos y funcione correctamente, y facilitar su mantenimiento a lo largo del tiempo. Producir un código legible y que sea fácil de mantener es tan importante como que logre cumplir con todos los requerimientos especificados.

La codificación del portal web de El Paseo se dividió en dos instancias. A partir del inicio de la cursada de la asignatura “Java y Aplicaciones Avanzadas sobre Internet” se comenzó a trabajar en el servidor de la aplicación.

Luego, al final de la cursada y cuando ya la API del backend estaba completa, se integró el cliente.

Cabe aclarar que en este apartado solamente se explicará cómo fue la organización para codificar todos los componentes del sistema, mientras que en el siguiente capítulo se entrará en detalle sobre las tecnologías elegidas para cada componente junto con la razón de su elección.

### 4.4.1. Codificación del servidor

La arquitectura del código del servidor está dividida en tres capas:

**Controladores:** Son los responsables de manejar las solicitudes HTTP entrantes desde el cliente. Su función principal es recibir las peticiones, interpretarlas y coordinar la respuesta. En general, los controladores interactúan directamente con las capas superiores (interfaz de usuario o capa de presentación) y pueden llamar a servicios para obtener datos o realizar operaciones de negocio.

**Servicios:** Encapsulan la lógica de negocio de la aplicación. Contienen métodos o funciones que realizan operaciones específicas, como procesar datos, realizar cálculos, interactuar con bases de datos a través de los repositorios o integrarse con otros servicios externos. Además, realizan el manejo de errores, lanzando excepciones específicas para los casos de errores conocidos. Los controladores pueden llamar a estos servicios para realizar tareas específicas sin preocuparse por la lógica subyacente.

**Repositorios:** Los repositorios interactúan directamente con la capa de persistencia, la cual es el componente del sistema que se encarga de almacenar, recuperar, actualizar y eliminar información en una base de datos o algún otro tipo de almacenamiento duradero. Los repositorios proveen métodos para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la base de datos, encapsulando la lógica de acceso a los datos. Los servicios suelen utilizar los repositorios para acceder y manipular datos en la base de datos.

Esta separación en capas ayuda a mantener un código más organizado, modular y de fácil mantenimiento, promoviendo la cohesión y reduciendo la dependencia entre las diferentes partes del sistema:

- Los controladores se centran en la gestión de solicitudes y respuestas, manteniendo la lógica de presentación separada.
- Los servicios encapsulan la lógica de negocio, permitiendo la reutilización y la separación de responsabilidades.
- Los repositorios manejan la interacción con la base de datos, aislando la lógica de persistencia de las demás capas. Esto nos permite principalmente ser independientes del tipo de tecnología elegida para persistir los datos, ya que si el día de mañana se decide hacer un cambio de la misma, solo haría falta modificar los repositorios, y este cambio sería transparente para el resto de las capas.

Con respecto a la organización de la codificación del servidor, se comenzó desarrollando la capa de persistencia. Para esto, se propuso la utilización del patrón DAO[7], ya que el mismo facilita la separación de la lógica de acceso a datos de la lógica de negocio al proporcionar una lógica de abstracción para el acceso a la fuente de datos. El patrón DAO entonces promueve la modularidad, la reutilización del código y facilita el cambio de la fuente de datos sin afectar la lógica de la aplicación. Se implementó a través del uso de los siguientes componentes:

**Interfaz DAO:** Es una interfaz que contiene métodos abstractos y genéricos para realizar operaciones CRUD (Crear, Leer, Actualizar, Borrar) en la fuente de datos, lo que significa que no tienen implementación y deben ser implementados por las clases que implementen la interfaz. Actúa como un contrato para la implementación concreta del acceso a datos. Al declarar una única interfaz genérica (es decir, que no está ligada directamente a un objeto concreto), fue muy sencillo reutilizar la lógica de los métodos básicos del CRUD en todas las entidades del sistema, siempre que fuera necesario.

**Clase abstracta JPA DAO:** Proporciona una implementación genérica, predeterminada y reutilizable de los métodos definidos en la interfaz IGenericDao utilizando JPA. Esta implementación común incluye la gestión de transacciones, la creación de consultas JPA, y otras tareas comunes asociadas con el acceso a datos utilizando JPA. Al ser una

[7]DAO por sus siglas en inglés: Data Access Object

clase abstracta no puede ser instanciada directamente, sino que otras clases deben extender de ella para utilizar sus métodos. Además, hay que tener en cuenta que esta implementación puede ser sobrescrita por clases concretas que extienden de la clase abstracta si se requiere un comportamiento específico para una entidad particular.

**Implementación DAO o repositorio:** Proporciona la implementación concreta de los métodos extendidos de la clase abstracta para un tipo específico de fuente de datos. Contiene la lógica real para realizar operaciones de acceso a datos utilizando consultas, procedimientos almacenados u otras operaciones específicas de la fuente de datos. Para cada entidad de nuestro sistema declaramos una implementación DAO la cual extendía las funcionalidades de la clase abstracta mencionada anteriormente, y, por lo tanto, también de la interfaz genérica. En caso de que el comportamiento predeterminado no sea útil para una entidad en particular, se sobrescribía el método en cuestión. Además, es posible agregar métodos personalizados para cada entidad.

A continuación mostramos cómo fue la implementación de la Interfaz genérica DAO junto con una implementación DAO en particular, a modo de ejemplo.

Listing 4.1: Interfáz genérica DAO

```
1  public interface IGenericDao<T extends Serializable> {
2    void setClazz(Class<T> clazzToSet);
3
4    T findOne(final long id);
5
6    List<T> findAll();
7
8    T create(final T entity);
9
10   T update(final T entity);
11
12   void delete(final T entity);
13
14   void deleteById(final long entityId);
15 }
```

Listing 4.2: Clase abstracta JPA DAO

```
1  public abstract class AbstractJpaDao<T extends
       Serializable> implements IGenericDao<T> {
2    private Class<T> clazz;
3    @Inject protected EntityManagerSingleton
       entityManagerFactory;
4
5    public final void setClazz(final Class<T> clazzToSet)
       {
6      this.clazz = clazzToSet;
7    }
8
9    public T findOne(final long id) {
10     EntityManager entityManager = entityManagerFactory.
         createEntityManager();
11     T entity = entityManager.find(clazz, id);
12     entityManager.close();
13     return entity;
14   }
15
16   public List<T> findAll() {
17     EntityManager entityManager = entityManagerFactory.
         createEntityManager();
18     List<T> entities = entityManager.createQuery("from "
           + clazz.getName()).getResultList();
19     entityManager.close();
20     return entities;
21   }
22
23   public T create(final T entity) {
24     EntityManager entityManager = entityManagerFactory.
         createEntityManager();
25     entityManager.getTransaction().begin();
26     entityManager.persist(entity);
27     entityManager.getTransaction().commit();
28     entityManager.close();
29     return entity;
30   }
31
32   public T update(final T entity) {
```

```
33      EntityManager entityManager = entityManagerFactory.
           createEntityManager();
34      entityManager.getTransaction().begin();
35      T result = entityManager.merge(entity);
36      entityManager.getTransaction().commit();
37      entityManager.close();
38      return result;
39    }
40
41    public void delete(final T entity) {
42      EntityManager entityManager = entityManagerFactory.
           createEntityManager();
43      entityManager.getTransaction().begin();
44      entityManager.remove(entityManager.contains(entity)
           ? entity : entityManager.merge(entity));
45      entityManager.getTransaction().commit();
46      entityManager.close();
47    }
48
49    public void deleteById(final long entityId) {
50      final T entity = findOne(entityId);
51      delete(entity);
52    }
53  }
```

Listing 4.3: Implementación DAO

```
1   public class AddressDao extends AbstractJpaDao<Address>
       {
2     public AddressDao() {
3       super();
4       this.setClazz(Address.class);
5     }
6
7     public List<Address> findAddressesByClientId(final
         Long clientId) {
8       EntityManager entityManager = super.
           entityManagerFactory.createEntityManager();
9
10      StringBuilder queryString =
11          new StringBuilder("SELECT a ")
```

```
12              .append(String.format("FROM %s a ", Address.
                    class.getSimpleName()))
13              .append("WHERE a.client.id =:clientId ORDER
                    BY a.id DESC");
14
15      Query query = entityManager.createQuery(queryString.
            toString(), Address.class);
16      query.setParameter("clientId", clientId);
17
18      List<Address> response = query.getResultList();
19
20      entityManager.close();
21      return response;
22    }
23  }
```

En este ejemplo, se puede observar como la clase AddressDao hereda la implementación de los métodos de la interface genérica a partir de extender de la clase AbstractJpaDao, pero además declara el método findAddressesByClientId(final Long clientId) necesario para encontrar las direcciones que pertenecen a un cliente en particular. De esta forma, queda demostrado como se puede personalizar el acceso a datos en cada entidad, compartiendo siempre entre ellas los métodos básicos de acceso.

Una vez que abarcamos todas las operaciones básicas de los repositorios de cada una de las entidades que coexisten en el sistema, continuamos con la codificación de los servicios. Primero nos concentramos en cubrir las llamadas a los métodos básicos de los repositorios, y luego avanzamos en resolver problemáticas mas complejas y personalizadas para cada entidad.

Entre las responsabilidades de los servicios, se encuentra la definición centralizada del manejo de errores con el objetivo de unificar la forma de detección y retorno de los mismos. Además, los servicios se encargan transformar los modelos (entidades de la base de datos) a DTOs[8] (objetos de transferencia de datos), lo cual es un patrón de diseño que se utiliza comúnmente en el desarrollo de software para transferir datos entre subsistemas o capas de una aplicación. En resumen, el patrón DTO ayuda a optimizar la transferencia de datos entre diferentes partes de una aplicación al proporcionar una representación estructurada y eficiente de la información que necesita

[8]DTO por sus siglas en inglés: Data Transfer Objects

ser transmitida. Esto resulta especialmente útil en arquitecturas distribuidas o en sistemas con capas bien definidas.

Una vez que los repositorios y servicios estaban desarrollados, continuamos con la codificación de los controladores, los cuales exponen cada funcionalidad a través de endpoints accesibles mediante el protocolo HTTP a través de una API web RESTful.

Una API[9] o interfaz de programación de aplicaciones es un conjunto de reglas, protocolos y herramientas que permite la interacción entre diferentes aplicaciones de software. De forma sencilla, se puede definir como un puente que permite que distintos programas se comuniquen entre sí, compartan datos y funcionen juntos de manera integrada. Definen las formas en que los distintos componentes de software pueden interactuar, permitiendo a una aplicación acceder a las funciones o datos de otra de manera controlada y segura. Esto se logra estableciendo reglas claras sobre cómo los distintos sistemas pueden solicitar o proporcionar información y qué tipo de respuestas pueden esperar.

Una API web, por su parte, es una interfaz que permite la comunicación a través de la web, generalmente utilizando protocolos como HTTP o HTTPS. Dentro de las APIs web se encuentran las API RESTful que siguen los principios y restricciones del estilo arquitectónico REST[10], el cual es un conjunto de principios de diseño que define cómo deben ser creadas las APIs para sistemas distribuidos, permitiendo la comunicación entre sistemas de manera estándar y eficiente. Están diseñadas para ser sencillas, escalables, y permitir una interacción eficiente entre servicios y aplicaciones[14]. Algunas de sus características clave son:

**Basada en estándares HTTP:** Utiliza los métodos estándar de HTTP (**GET, POST, PUT, DELETE**) para realizar operaciones sobre recursos. Esto además hace que provea una interfaz uniforme, ya que este conjunto de operaciones bien definidas se aplica de igual manera a todos los recursos.

**Recursos y URIs:** Los datos son representados como recursos, y cada recurso es accesible a través de un identificador único, generalmente representado por una URI[11][14]. Por ejemplo, una URI podría ser algo

[9] API por sus siglas en inglés: Application Programming Interface
[10] REST por sus siglas en inglés: Representational State Transfer
[11] URI por sus siglas en inglés: Uniform Resource Identifier

como **/usuarios** para acceder a una lista de usuarios o **/usuarios/123** para acceder a un usuario específico con el identificador 123.

**Operaciones CRUD:** Emplea operaciones CRUD sobre los recursos utilizando los métodos HTTP: **GET** para leer, **POST** para crear, **PUT/-PATCH** para actualizar y **DELETE** para eliminar.

**Sin estado (Stateless):** Cada solicitud al servidor contiene toda la información necesaria para procesar esa solicitud. El servidor no guarda el estado de la sesión entre peticiones, lo que significa que cada solicitud es independiente y autocontenida.

**Uso de representaciones:** Los recursos pueden tener diferentes representaciones (**JSON, XML**, etc.) según las necesidades del cliente. La API puede responder con diferentes formatos de datos basados en los encabezados de solicitud del cliente.

Para codificar nuestra API, primero definimos las rutas y endpoints con sus parámetros de entrada, luego la gestión de los datos a través de la llamada a los servicios, y para finalizar se definió los tipos de respuesta adecuados (JSON, XML, etc.) con los códigos de estado HTTP correspondientes para indicar el resultado de cada solicitud.

Toda la API se documentó de una forma clara y detallada usando Swagger, especificando información de cada endpoint, parámetros, tipos de respuesta y ejemplos de uso. Además, implementamos medidas de seguridad a través de tokens de acceso para cada endpoint en específico, según sea necesario.

### 4.4.2. Codificación del cliente

A lo largo del tiempo, los sitios web estáticos quedaron casi totalmente reemplazados por sitios dinámicos debido a sus indiscutibles ventajas: los sitios estáticos muestran contenido fijo (no cambian a partir de la interacción del usuario) y requieren actualizaciones manuales en archivos HTML, siendo cada archivo una página distinta; mientras que los dinámicos generan contenido interactivo según la interacción del usuario o datos de una base de datos, utilizando lenguajes de programación para actualizaciones automáticas, lo que los hace más complejos en su estructura y funcionamiento.

Para este proyecto, dadas las ventajas de los sitios web dinámicos, decidimos desarrollar el cliente como una aplicación de una sola página o SPA[12], la cual funciona en un único documento HTML y ofrece una experiencia similar a la de una aplicación de escritorio o móvil. A diferencia de las aplicaciones web tradicionales, que cargan múltiples páginas HTML completas cuando el usuario navega entre secciones, una SPA carga una sola página al inicio y luego actualiza dinámicamente el contenido según las interacciones del usuario, sin necesidad de recargar la página completa. Sus características principales son:

**Carga Inicial:** Al principio, la SPA carga todo el código necesario: HTML, CSS y JavaScript. A partir de ahí, las interacciones del usuario no requieren cargar nuevas páginas HTML; en su lugar, se realizan mediante interacciones dinámicas en la misma página.

**Interfaz Dinámica:** Utiliza JavaScript para manejar las interacciones del usuario. Cuando el usuario navega o realiza acciones, como hacer clic en un enlace, enviar un formulario, etc., la SPA actualiza y muestra el contenido relevante sin recargar la página completa.

**Enrutamiento en el Cliente:** Suelen utilizar un enrutamiento en el lado del cliente, lo que significa que el enrutamiento y la gestión de las diferentes vistas o secciones se manejan mediante JavaScript en lugar de depender del servidor para cargar nuevas páginas.

**APIs y Servicios:** Son ideales para trabajar con APIs, ya que pueden realizar solicitudes a servicios web o APIs RESTful en segundo plano y actualizar la interfaz de usuario en consecuencia, sin necesidad de recargar la página.

**Experiencia de usuario similar a la de una aplicación nativa:** Debido a su naturaleza dinámica y fluidez en la navegación, pueden ofrecer una experiencia de usuario similar a la de una aplicación nativa, lo que las hace populares para aplicaciones web complejas y ricas en interactividad.

Así es que en arquitecturas cliente-servidor, las SPAs se basan en un modelo donde el cliente (navegador) solicita recursos al servidor, y este responde

[12]SPA por sus siglas en inglés: Single Page Application

con datos, usualmente en formatos como JSON o XML. El cliente, mediante JavaScript y frameworks como React, Angular o Vue, maneja la lógica de presentación y actualización del contenido, realizando peticiones al servidor solo para obtener o enviar datos, minimizando la necesidad de cargar nuevas páginas completas. Esto crea una experiencia más fluida para el usuario al interactuar con la aplicación[15].

Teniendo todo esto en cuenta, la codificación del sistema cliente se llevó a cabo de una forma mucho más directa y sencilla que la codificación del servidor explicada anteriormente, en gran parte gracias a la documentación existente de la API del backend que facilitó la integración con el frontend.

En lo que respecta a la organización de la codificación del cliente, nuestro enfoque inicial se centró en desarrollar las interfaces correspondientes a las maquetas presentadas en la fase de Diseño 4.3 detallada anteriormente. Inicialmente, no integramos funciones de seguridad para el acceso a estas interfaces, sino que simplemente establecimos las rutas para cada componente individual. Además, procuramos mantener una coherencia visual y de estilos en todas las pantallas diseñadas, utilizando los colores, tipografías, imágenes y logos proporcionados por el equipo de diseño de El Paseo.

Una vez que completamos el desarrollo de todas las pantallas, procedimos a implementar medidas de seguridad para el sistema web. Implementamos un control que verifica el rol de cada usuario (CONSUMIDOR, PRODUCTOR o ADMINISTRADOR) para permitir el acceso a las páginas correspondientes según su rol asignado. Adicionalmente, existen interfaces que solo pueden ser accedidas por el mismo usuario autenticado. Por ejemplo, el usuario X con rol CONSUMIDOR no podrá acceder a la pantalla de perfil del consumidor Y. Se ha implementado este control adicional para gestionar adecuadamente el acceso a dichas interfaces.

Por otro lado, comenzamos a evaluar qué otras APIs serían beneficiosas de incorporar. Esta consideración nos llevó a pensar en la integración de la API de Google Maps[13], lo cual nos permitiría mostrar los puntos de retiro disponibles en un mapa en la página principal para que siempre puedan ser accedidos por los usuarios consumidores. También consideramos la API de WhatsApp[14] para simplificar y promover la comunicación directa del usuario con El Paseo.

[13] Véase `https://developers.google.com/maps/apis-by-platform?hl=es-419`
[14] Véase `https://business.whatsapp.com/developers/developer-hub?lang=es_LA`

## 4.5. Calidad, Pruebas, Gestión de la Configuración y Mantenimiento del software

### 4.5.1. Calidad del software

La definición de calidad es subjetiva y depende del enfoque que se le dé, pudiendo variar según las percepciones y expectativas individuales de cada individuo involucrado. Algunos autores se enfocan únicamente en el producto y establecen que un software es de calidad cuando tiene buenas características, dejando de lado lo que un cliente o usuario solicite. Por otro lado, otros autores se centran en la satisfacción del cliente, estableciendo que la calidad es una percepción que solo los clientes pueden definir.

Si unificamos ambos enfoques, podemos concluir que un sistema de software es de calidad cuando cumple con los requerimientos especificados y sirve para el propósito para el que fue creado.

Desarrollar un software de calidad significa que:

- El sistema satisface plenamente los requerimientos funcionales especificados, expectativas y necesidades del cliente.
- Durante todo el desarrollo del sistema se consideraron los requerimientos no funcionales tales como portabilidad, seguridad, eficiencia, reusabilidad, etc.
- El sistema desarrollado es correcto, eficiente, flexible, confiable y seguro.
- La calidad ha sido asegurada, medida y controlada a través de cada una de las fases de desarrollo del sistema.

La gestión de la calidad es el conjunto de actividades destinadas a la detección y corrección temprana de defectos en el sistema que se está produciendo, tales como:

- Revisión entre colegas: Se somete a revisión un artefacto en particular (documento de diseño, código, etc.) para realizar observaciones e iterar la solución final, alcanzando un consenso con los demás colegas.
- Inspecciones del software: Se revisa la consistencia entre los artefactos del sistema y se verifica que se cumpla con las buenas prácticas (previamente definidas y documentadas).

- Pruebas del software: Se ejecuta el sistema haciendo uso de la funcionalidad para la que fue diseñado, con el objetivo de verificar que se ejecute correctamente.

En este proyecto se realizaron diversas tareas de gestión de calidad con el objetivo de que el sistema alcanzara un nivel óptimo de calidad. Por ejemplo: se gestionaron revisiones periódicas de documentos técnicos y de diseño, y principalmente del código de los distintos componentes. Cada vez que se quería sumar una funcionalidad nueva al sistema, se hacía una revisión detallada del nuevo código a incorporar, intentando encontrar errores o cuestiones de diseño que se podían mejorar, y también promover la coherencia del código en cuanto a legibilidad y prolijidad, para facilitar las tareas de mantenimiento que se llevarán a cabo en el futuro, una vez que el sistema esté productivo.

### 4.5.2. Pruebas

La fase de pruebas en indispensable en todo proyecto de software, ya que ningún producto de software está exento de defectos que pueden generar fallas. Es así que no es viable para un desarrollador entregar un producto sin haber pasado por un proceso de pruebas que proporcione los niveles de confianza necesarios para poder usar el sistema.

El estándar ANSI/IEEE 610.12-1990 define prueba como "el proceso de operar un sistema o componente bajo condiciones específicas, observar y registrar los resultados, y realizar una evaluación de algún aspecto del sistema o componente".

Las pruebas del software consisten en un conjunto de actividades encaminadas a identificar defectos, tanto durante el desarrollo del producto de software como durante la ejecución del mismo, abarcando cada una de las fases de desarrollo del producto. Al ejecutar una prueba se contrasta el comportamiento observado contra el comportamiento esperado del sistema o componente, documentando las diferencias para un posterior análisis.

Los tipos de prueba pueden clasificarse a partir del objeto que se desea probar. De esta forma podemos identificar a las *pruebas unitarias*, las cuales realizan pruebas en un módulo por separado; las *pruebas de integración*, llevadas a cabo para mostrar que, aunque los módulos hayan pasado sus pruebas unitarias, hay problemas al momento de integrar unos con otros; y por último, las *pruebas de sistema*, las cuales prueban al sistema entero y se llevan a cabo para probar los requisitos no funcionales (por ejemplo,

seguridad, desempeño, exactitud y confiabilidad). Por lo general, las fallas de funcionamiento se detectan durante las pruebas unitarias y de integración.

### Pruebas de aceptación

Las pruebas de aceptación representan la fase del ciclo de vida de desarrollo de software en el que el equipo de desarrollo y los usuarios de un sistema de software tienen que garantizar que el sistema desarrollado se corresponde con los requerimientos definidos [16].

Son pruebas cruciales, ya que determinan si el software está listo para ser lanzado al mercado o para su uso en un entorno productivo, enfocándose en validar la funcionalidad, usabilidad, y otros aspectos del sistema desde la perspectiva del usuario final.

El equipo de desarrollo es el encargado de diseñar las pruebas, basándose en los requisitos funcionales especificados en la etapa de Análisis de Requerimientos, y en las expectativas de los usuarios y cliente. Luego, son ejecutadas por el propio usuario final.

Dependiendo de la complejidad del sistema a probar, si está o no dividido por módulos, etc., la realización de las pruebas se ejecuta de forma diferente. Por lo general, pueden llevarse a cabo mediante dos tipos de procedimientos: pruebas alfa y pruebas beta.

**Pruebas alfa**: Se le entrega a un subgrupo de usuarios finales el producto terminado junto a su documentación correspondiente. El usuario es el que se encargará de realizar las pruebas por su cuenta, utilizando la documentación como guía, e irá informando las inconsistencias y errores que vaya encontrando en las distintas funcionalidades. Estas pruebas se llevan a cabo en un entorno específico previamente preparado para las pruebas, y en presencia del desarrollador del sistema.

**Pruebas beta**: Por lo general se realizan a continuación de las pruebas alfa. Los usuarios finales reales prueban el software situados en sus lugares concretos de puesto de trabajo (en su entorno real). Sin la presencia del equipo de desarrollo, los usuarios se encargan de emitir los informes de resultados e impresiones del software.

En este proyecto, para organizar las pruebas funcionales, primero realizamos la planificación de las mismas, definiendo criterios de aceptación, casos de prueba, y preparando el entorno de pruebas. Con respecto al entorno de pruebas, lo que se hizo fue publicar una imágen en DockerHub que definía los containers necesarios para levantar el ambiente de El Paseo

(frontend, backend y base de datos), y esa imágen fue instalada en un servidor perteneciente a la facultad, quedando desplegada en el siguiente link https://elpaseo.linti.unlp.edu.ar/frontend/.

La etapa de pruebas entonces quedó dividida en dos intancias: una primer instancia dedicada a los usuarios administrativos del sistema, y otra posterior dedicada a los usuarios consumidores, tanto habituales como nuevos.

Una vez que se terminó con las tareas de planificación y contábamos con un ambiente de pruebas listo, permitimos a los usuarios finales ejecutar los distintos casos de prueba, registrando los resultados y cualquier problema o defecto encontrado. Para esto, les compartimos a los usuarios adminsitrativos el Formulario de Google Pruebas UAT - El Paseo, y a los usuarios consumidores el Formulario de Google Pruebas del sistema de El Paseo te lo lleva. En estos formularios se indicaban los pasos a seguir para ejecutar cada prueba para cada tipo de usuario, con el objetivo de que sirva como guía. Una vez ejecutadas las pruebas por parte de los usuarios, debían indicar si aceptaba la funcionalidad junto con las observaciones encontradas.

Por último, analizamos estos resultados comparándolos con los criterios de aceptación, identificando problemas y reorganizando el trabajo para priorizar y resolver cualquier defecto que se haya identificado en el sistema. Esta tarea se nos simplificó, ya que el Formulario de Google que utilizamos te permite exportar los resultados obtenidos a un Documento de Excel, el cual utilizamos para analizar los resultados de cada caso de prueba.

### 4.5.3. Gestión de la configuración

La gestión de la configuración tiene como objetivo controlar los cambios que se generan a lo largo de un proyecto de software para que los elementos que lo componen (código, documentos y datos) estén actualizados y contengan la información correcta. Es en sí una fase transversal del proceso de desarrollo, y tiene una gran importancia en la gestión de la evolución de un sistema de software. Inicia cuando comienza el proceso de desarrollo y concluye cuando el software ha sido liberado. Durante la gestión de la configuración se desarrollan y aplican estándares y procedimientos que permitan administrar la evolución, las versiones y los cambios del mismo.

Las actividades básicas de la gestión de la configuración son:

1. Identificación de los elementos de la configuración del sistema: Se proporciona un identificador a cada uno de los elementos del sistema (a

cada módulo, a cada documento y a los elementos de la base de datos).

2. Control de cambios: Es un subproceso en el que se evalúa la necesidad y el costo de cada cambio. En caso de que el cambio se apruebe, se genera la orden de cambio, y se le da seguimiento de forma organizada a su implantación, análisis, revisión y a la nueva versión que este origina.

3. Control de versiones: Una versión de un sistema de software es una instancia o copia del sistema, que difiere de alguna forma de otras descripciones. Con el control de versiones, se establece un sistema para nombrar las distintas versiones del software. Se refiere al uso de procedimientos y herramientas para gestionar las versiones de los elementos de configuración del sistema de software, tales como: las especificaciones de los requerimientos, del diseño arquitectónico, o del diseño detallado, las estructuras de datos, los programas fuentes, etc. Todo control de cambios conlleva un control de versiones.

En la sección 5 analizamos con más detalle el control de cambios y de versiones adoptados para este proyecto.

### 4.5.4. Soporte técnico y Mantenimiento

El soporte técnico es el servicio que se le brinda al cliente cuando el sistema ya está en operación con el propósito de solucionar los problemas que aparecen en producción. Incluye la capacitación del usuario para que pueda trabajar con el sistema, la solución de dudas y, cuando es necesario, corregir las fallas detectadas en el sistema. La asistencia se puede proporcionar a los usuarios finales, clientes o propietarios de un producto de software o hardware, ante los problemas y dificultades que puedan surgir durante su uso.

Por otro lado, el mantenimiento es el conjunto de actividades que se llevan a cabo para hacer actualizaciones de un sistema de software cuando ya está en operación. Entonces, es una fase posterior al desarrollo y liberación del producto, que por lo general busca:

- Incorporar nueva funcionalidad al producto de software.
- Mejorar la funcionalidad presente.
- Mejorar el rendimiento del producto.

- Corregir defectos no detectados durante la fase de pruebas.

Un sistema de software desarrollado, atendiendo a los principios de reusabilidad y extensibilidad, garantizará un proceso de mantenimiento flexible. Si se respetaron las etapas del proceso de desarrollo de software y se veló por la calidad del software, entonces el mantenimiento estará orientado a la incorporación de mejoras a la funcionalidad del sistema software, y no tanto a la corrección de errores.

Dado que el estado actual de este proyecto de software es la fase de pruebas, las tareas de mantenimiento y soporte técnico se llevarán a cabo una vez que el sistema esté en producción.

Igualmente, es importante destacar que a lo largo de las etapas descritas anteriormente se buscó trabajar para garantizar que las tareas futuras de mantenimiento y soporte técnico no sean complejas, ya que, por ejemplo: buscamos desarrollar un software intuitivo para que los usuarios naveguen fácilmente sin complicaciones (es decir, sin la necesidad de un soporte técnico), les brindamos a los administradores herramientas claras para poder hacer troubleshooting[15] de los problemas que puedan surgir, y buscamos que el software abarque de forma completa las funcionalidades que El Paseo necesitaba para trabajar, con el objetivo de evitar tareas de mantenimiento para sumar funcionalidades importantes para el negocio, entre otras cosas.

[15] Término utilizado en sistemas para indicar el proceso que se lleva a cabo para resolver problemas complicados en un sistema.

# Capítulo 5

# Implementación del portal: tecnologías y metodologías adoptadas

A lo largo de este capítulo, describiremos las tecnologías empleadas en la codificación del portal de El Paseo. Es importante resaltar que la selección de estas tecnologías se basó en varios aspectos fundamentales:

1. La incorporación de ciertas tecnologías surgió durante la cursada de la asignatura "Java y Aplicaciones Avanzadas sobre Internet", donde se requería utilizarlas como parte del plan de estudios para mantener coherencia con el resto de los compañeros y aprender los temas principales de la materia.

2. Una vez terminada la cursada, a la hora de elegir qué tecnologías usar para la codificación de funcionalidades que estaban por fuera del alcance de la asignatura, se consideraron:

   *a*) Mi experiencia profesional trabajando como desarrolladora de software en diversas empresas.

   *b*) La alineación con estándares de software libre y código fuente abierto.

   *c*) La necesidad de contar con un sistema en producción adecuado tanto a las necesidades de la comercializadora, como a su capacidad económica.

## 5.1. Software libre y de código abierto

Como mencionamos anteriormente, uno de los criterios utilizados a la hora de seleccionar qué tecnología utilizar para el desarrollo de ciertas funcionalidades fue la adhesión a los estándares y principios del software libre y de código abierto.

El **software libre** se refiere a programas informáticos cuyo código fuente está disponible para ser estudiado, modificado y distribuido por cualquier persona. Este concepto se basa en cuatro libertades fundamentales: la libertad de usar el software para cualquier propósito, estudiar cómo funciona, adaptarlo a las necesidades específicas y distribuir copias del mismo, ya sea de forma gratuita o por un precio.

El movimiento del software libre fue iniciado por el físico estadounidense Richard Stallman en el Instituto de Tecnología de Massachusetts (MIT) en la década de 1980. Stallman, preocupado por las restricciones impuestas por el software propietario (los sistemas que impiden compartir o modificar el software), fundó la Free Software Foundation (FSF) en 1985 y desarrolló la Licencia Pública General de GNU (GPL[1]), una licencia que asegura que el software bajo sus términos siempre sea libre para ser utilizado, modificado y compartido.

El término "GNU" significa "GNU's Not Unix"[2] y representa un esfuerzo para crear un sistema operativo completamente libre y de código abierto, similar a Unix pero sin las restricciones de uso.

El movimiento del software libre se expandió con el tiempo, inspirando a comunidades de desarrolladores y usuarios a colaborar en la creación y distribución de software que respetara estas libertades. Este enfoque ha influido en la cultura tecnológica y ha llevado al surgimiento de proyectos y organizaciones que promueven el uso y la creación de software libre en todo el mundo.

Stallman aclara en su libro "Software Libre para una sociedad libre" que el término de *software libre* no tiene ninguna relación con el precio, sino que busca hacer énfasis en la *libertad* [17]. Entonces, un programa es software libre para el usuario siempre que tengas la libertad de:

1. Ejecutar el programa sea cual sea el propósito.

[1] GPL por sus siglas en inglés: General Public License

[2] En castellano, ≪GNU No es Unix≫. Sitio oficial del sistema operativo GNU: `https://www.gnu.org/philosophy/free-sw.es.html`

2. Modificarlo para ajustarlo a tus necesidades. Además, para que se trate de una libertad efectiva en la práctica, deberás tener acceso al código fuente, dado que sin él la tarea de incorporar cambios en un programa es extremadamente difícil.

3. Redistribuir copias, ya sea de forma gratuita o a cambio del pago de un precio.

4. Distribuir versiones modificadas del programa, de tal forma que la comunidad pueda aprovechar las mejoras introducidas.

A partir de 1998, parte de la comunidad abandonó el término de software libre y empezó a hablar de **código abierto** tratando de evitar la confusión entre libre y gratuito. Este nuevo término se refiere a un enfoque de desarrollo de software que se basa en la idea de hacer públicos el código fuente y el diseño del programa, permitiendo a cualquiera ver, modificar y distribuir el software de manera libre. Aunque se relaciona estrechamente con el software libre, se enfoca más en los aspectos prácticos y en los beneficios del desarrollo colaborativo y transparente. Cuando se trabaja con código abierto está permitida la distribución del código siempre que se respeten los términos de su licencia y no varíen desde su primera adquisición hasta su distribución.

Líderes del campo, como Eric S. Raymond[3] y Bruce Perens[4], decidieron adoptar este término para promover los beneficios prácticos y empresariales del software con código accesible. La intención era atraer a más empresas y desarrolladores hacia este modelo de desarrollo colaborativo, haciendo hincapié en los aspectos de calidad, seguridad, flexibilidad y costo-eficiencia del software.

[3]Desarrollador de software, autor y defensor del software de código abierto. Es reconocido por su papel en la promoción y la filosofía del código abierto. Raymond es conocido por su obra ´´The Cathedral and the Bazaar" (La Catedral y el Bazar), un ensayo que analiza los modelos de desarrollo de software, comparando el enfoque cerrado y jerárquico ("la Catedral") con el enfoque abierto y colaborativo ("el Bazar"). Este ensayo se convirtió en un texto influyente para comprender los principios del desarrollo de software de código abierto.

[4]Es reconocido por su importante papel en la promoción y defensa del software libre, así como por su contribución al desarrollo de la comunidad de código abierto. Perens es conocido por haber coescrito la "Definición de código abierto", un documento fundamental que establece los principios y criterios del movimiento del código abierto. Además, fue una figura clave en el proyecto Debian, una distribución de software libre de Linux, y fue el creador del término "Open Source Initiative" (Iniciativa de Código Abierto), que ayudó a popularizar la idea del código abierto entre las empresas y la comunidad de desarrollo.

El concepto de código abierto también se centra en la calidad del código, la revisión colaborativa y la idea de que la apertura puede llevar a un mejor software. Su filosofía se enfoca en cómo la apertura del código puede llevar a mejores prácticas de desarrollo y a productos de software superiores, independientemente de las cuestiones de libertad del usuario.

Stallman afirma que *"(...) El movimiento de software libre y el movimiento open source son hoy en día movimientos separados con diferentes puntos de vista y objetivos, aunque podamos y trabajemos juntos en algunos proyectos prácticos."*[17]. Además, remarca la diferencia entre ambos conceptos de la siguiente manera: *"La diferencia fundamental entre los dos movimientos está en sus valores, en su visión del mundo. Para el movimiento open source, la cuestión de si el software debe ser de fuente abierta es una cuestión práctica, no ética. Como lo expresó alguien, «el open source es un método de desarrollo; el software libre es un movimiento social». Para el movimiento open source, el software no libre es una solución ineficiente. Para el movimiento de software libre, el software no libre es un problema social y el software libre es la solución."*[17]

Si bien se pueden apreciar diferencias en los principios básicos, la realidad es que ambos conceptos se alinean en las recomendaciones prácticas para anteponerse al código propietario, por lo que en muchos proyectos trabajan en conjunto.

Los estudiantes de la Facultad de Informática de la UNLP no solo adquieren estos conocimientos al inicio de sus estudios universitarios, sino que también los ponen en práctica al experimentar directamente los beneficios de los desarrollos colaborativos basados en los principios del software libre y código abierto. Por esta razón, al llevar a cabo la implementación del portal web para El Paseo, consideramos fundamental adherirnos a sus valores con el propósito de contribuir a la comunidad y fomentar la continuación de estos ideales.

## 5.2. Metodologías ágiles

Las metodologías ágiles en el desarrollo de software son enfoques de gestión y desarrollo que priorizan la flexibilidad, la adaptabilidad y la colaboración en el proceso de construcción de software. Estas metodologías se centran en la entrega iterativa e incremental, la priorización de los requerimientos a desarrollar, la retroalimentación continua entre desarrolladores y clientes y

la respuesta rápida a los cambios en los requisitos del cliente [18]. Algunas de las metodologías ágiles más conocidas incluyen Scrum, Kanban, Extreme Programming (XP) y Crystal.

Las metodologías ágiles surgieron como una respuesta a los enfoques tradicionales de desarrollo de software, como el modelo en cascada, que a menudo enfrentaban dificultades en adaptarse a cambios en los requisitos y en satisfacer las necesidades cambiantes de los clientes. Los procesos de desarrollo de software tradicionales se caracterizaban por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas [19]. Al adoptar metodologías ágiles, los equipos de desarrollo buscan mejorar la eficiencia, la calidad y la satisfacción del cliente a lo largo del proceso de desarrollo de software. Además, dan como un hecho que los requerimientos van a cambiar durante el proceso de desarrollo.

En 2001 se crea el Manifiesto por el desarrollo ágil de software, documento en el que se acuerdan cuatro principios básicos, estableciendo prioridades y marcando diferencias de fondo frente a los sistemas tradicionales: individuos e interacciones, por encima de procesos y herramientas; software funcionando, por encima de documentación extensiva; colaboración con el cliente, por encima de negociación contractual; y respuesta ante el cambio, por encima de seguir un plan rígido [20].

### 5.2.1. Scrum

Se creó como un nuevo enfoque para el desarrollo de productos dirigido a incrementar su flexibilidad y rapidez a partir de la colaboración eficaz de equipos de proyectos estructurados. Es un marco de trabajo que emplea un conjunto de reglas y artefactos y define roles que generan la estructura necesaria para su correcto funcionamiento, con el objetivo de centrarse en la transparencia, inspección y adaptación, entregando productos de calidad[18].

Los llamados Equipos Scrum son autogestionados, multifuncionales y trabajan en iteraciones. La autogestión les permite elegir la mejor forma de hacer el trabajo, en vez de tener que seguir lineamientos de personas que no pertenecen al equipo y carecen de contexto. Los integrantes del equipo tienen todos los conocimientos necesarios (por ser multifuncionales) para llevar a cabo el trabajo. La entrega del producto se hace en iteraciones; en donde cada iteración crea nuevas funcionalidades, y los entregables son pequeños y concretos.

La intención de Scrum es la de maximizar la retroalimentación sobre el

desarrollo, pudiendo corregir problemas y mitigar riesgos de forma temprana. Sus principales características son:

**Roles:**

- Scrum Master: Facilita el proceso Scrum, elimina obstáculos y ayuda al equipo a mejorar continuamente.
- Product Owner: Representa al cliente y es responsable de definir y priorizar los requisitos del producto.
- Equipo de Desarrollo: Grupo multifuncional y autoorganizado que realiza el trabajo de desarrollo.

**Eventos (Ceremonias):**

- Planificación del Sprint: Reunión al inicio de cada sprint para definir las tareas que se realizarán durante ese periodo.
- Reuniones Diarias (Daily Scrum): Breves reuniones diarias de no más de 15 minutos para sincronizar al equipo y discutir el progreso, con el propósito de tener realimentación sobre las tareas de los recursos y los obstáculos que presentan.
- Revisión del Sprint: Sesión al final del sprint para demostrar y evaluar el trabajo completado y recibir retroalimentación.
- Retrospectiva del Sprint: Reflexión al final del sprint para identificar mejoras y ajustes para el próximo sprint.

**Artefactos:**

- Product Backlog: Lista priorizada de todas las características, mejoras y correcciones necesarias en el producto.
- Sprint Backlog: Conjunto de tareas que el equipo se compromete a completar durante un sprint.
- Incremento: Producto funcional y potencialmente entregable al final de cada sprint.

**Sprints:** Períodos de tiempo fijos y cortos (usualmente de 2 a 4 semanas) durante los cuales se desarrolla un incremento del producto. Son las iteraciones en las que se irá evolucionando el producto de software evolutivamente. Cada sprint tendrá su propio Sprint Backlog que será un subconjunto del Product Backlog con los requerimientos a ser construidos en el Sprint correspondiente[19]. Cabe destacar que Scrum se resiste a los cambios durante una iteración. Es por esto que se dan situaciones como la mencionada en el libro "Kanban y Scrum – obteniendo lo mejor de ambos": *Típicamente, un equipo Scrum diría algo como "No, lo siento, nos hemos comprometido a A+B+C+D en este sprint. Pero tómate la libertad de añadir E a la pila de producto. Si el dueño del producto considera que es de alta prioridad será añadido al siguiente sprint"*[21]

### 5.2.2. Kanban

Kanban es un enfoque ágil para la gestión y mejora de procesos que surgió en Toyota Production System a finales de los años 40. A diferencia de Scrum, Kanban no prescribe roles específicos ni eventos formales; en su lugar, se centra en la visualización del flujo de trabajo y la optimización continua, organizando el trabajo en señales visuales para gestionar el esfuerzo y dedicación del equipo de producción.

Kanban entonces utiliza tableros visuales para representar el flujo de trabajo. Los elementos del trabajo, a menudo representados por tarjetas, se mueven a través de columnas que representan las diferentes etapas del proceso, desde la planificación hasta la finalización.

Es un sistema de gestión de trabajo en curso (WIP[5]) donde se produce exactamente aquella cantidad de trabajo que el equipo es capaz de asumir, sin sobrecargas en el equipo de producción. En otras palabras, es un sistema de trabajo *just-in-time*, lo que significa que evita la inversión innecesaria de tiempo y esfuerzo en lo que no necesitaremos (o simplemente es menos prioritario), buscando así no sobrecargar al equipo.

Además, se busca limitar el trabajo en curso basándose en que está demostrado que cuanto más trabajo en curso se gestione a la vez, los índices de calidad disminuyen drásticamente. Al disminuir la cantidad de trabajo en curso se consigue que el enfoque en cada una de las tareas sea mayor y que el tiempo dedicado a todas ellas, sumado, sea menor que el empleado en

[5]WIP por sus siglas en inglés: Work in Progress

asumirlas todas de golpe[22].

Por otro lado, la gestión de tareas se basa en la demanda: las tareas se añaden al tablero Kanban a medida que hay capacidad para manejarlas. No hay planificación fija, y la priorización se realiza de manera dinámica según la demanda y la capacidad del equipo. Como se menciona en el libro "Kanban y Scrum – obteniendo lo mejor de ambos": *¿Qué ocurre si alguien cambia de idea y quiere añadir E al tablero? La respuesta de Kanban podría decir "Añade con libertad E a la columna Pendiente, pero hay un límite de 2 para esta columna, en consecuencia tendrás que quitar C o D. Ahora estamos trabajando en A y B, pero tan pronto como tengamos capacidad disponible cogeremos el primer elemento de la columna Pendiente".*[21]

### 5.2.3. Scrumban

A lo largo de la carrera universitaria y la experiencia profesional, hemos comprendido que no siempre resulta beneficioso adherirse de manera inflexible a un concepto teórico y seguirlo sin adaptaciones, limitándonos a una única herramienta. En cambio, suele ser más apropiado evaluar las características y el contexto específico de cada proyecto para determinar qué principio teórico puede aplicarse mejor a resolver cada desafío particular. Es por ello que, en el desarrollo del portal de El Paseo, no nos limitamos a seguir los lineamientos de una única metodología ágil, sino que se ha ajustado la forma de trabajo según las necesidades de cada situación.

En particular, consideramos que la metodología ágil adoptada resultó ser un híbrido entre el Kanban y el Scrum. Y además, podemos dividir el desarrollo del portal en dos etapas: la primera llevada a cabo durante la cursada de la asignatura "Java y Aplicaciones Avanzadas sobre Internet", y la segunda que se comenzó una vez concluida la materia, cuando se decidió que se iba a extender el proyecto para la realización de la tesina.

Entonces, en la primera etapa del desarrollo del portal, la forma de trabajo se podría identificar con el Kanban: no contábamos con una estructura de roles, iteraciones fijas o planificación de tareas. Una vez que el alcance del proyecto para la materia fue explicado, los alumnos fuimos "añadiendo tarjetas al tablero" a medida que teníamos que resolver las entregas dispuestas por la cátedra. En otras palabras, cada semana íbamos completando funcionalidades específicas para cada entrega, y a medida que se completaban las mismas se añadían nuevas tareas correspondientes a la siguiente entrega. Es por esto que nuestro tablero de tareas era persistente: nunca se limpiaba como

se acostumbra a hacer en Scrum, donde luego de cada iteración se eliminan las tareas abordadas y se comienza de cero en un nuevo sprint. Además, las instancias de entregas pautadas por la cátedra no tenían una duración fija, sino que variaban según la complejidad de cada funcionalidad abordada.

Generalmente, cada semana teníamos la muestra de las tareas completadas con el docente asignado a cada equipo de desarrollo. Se podría hacer un paralelismo entre esta dinámica y la ceremonia de Scrum de revisión de Sprint, coloquialmente llamada "demo" (por la acción de demostrar los avances del sistema). Pero cabe destacar que estas iteraciones no cumplían con los principios de Scrum de tiempo fijo, ya que como se mencionó anteriormente, los tiempos de las entregas se iban adaptando tanto a la complejidad de las funcionalidades a desarrollar, como al ritmo de las clases teóricas.

Además, naturalmente nuestro trabajo en progreso (WIP) estaba limitado por el estado del flujo de trabajo, como lo indica la metodología Kanban. Es decir, para avanzar en el desarrollo, la organización se basaba en iniciar una tarea (avanzando la tarjeta correspondiente al estado "En Progreso" en el tablero) y solo después de completada (y avanzada al estado "Terminada") se podía comenzar con el desarrollo de otra tarea. Esta dinámica surgía naturalmente, ya que en las clases de la materia, primero se proporcionaba a los alumnos una explicación teórica sobre las herramientas disponibles para abordar una problemática específica del sistema. Luego, continuábamos a la sección práctica en donde aplicábamos lo aprendido. Al final de esta sección, revisábamos los avances con el profesor asignado, lo que naturalmente implicaba que corrigiéramos y completáramos una tarea durante la misma clase. Por su parte, en Scrum no hay ninguna regla que impida que el equipo tenga todas las tareas en el estado "En Progreso" al mismo tiempo, pero esto hubiera significado una mala organización con nuestro docente asignado.

Los equipos de Kanban tratan de minimizar el tiempo de entrega y el nivel de flujo, por lo que, indirectamente, se crea un incentivo para descomponer los elementos en pedazos relativamente pequeños. Pero no hay ninguna norma explícita que indique que los elementos deben ser lo suficientemente pequeños como para caber en un intervalo de tiempo específico. En el mismo tablero podría haber un elemento que necesita un mes para terminarse y otro elemento que necesita un solo día[21]. En nuestro caso, en esta instancia del desarrollo guiada por la asignatura, en nuestro tablero podíamos tener tareas supersencillas como "Instalar el framework Jersey para el soporte de API REST", como también "Desarrollar toda la capa de persistencia", la cual naturalmente llevaba mucho más tiempo y esfuerzo que la anterior.

Además, a lo largo de la cursada no dedicábamos esfuerzo en prescribir la estimación de las tareas de nuestro tablero, práctica llevada a cabo en equipos de Scrum para luego medir la velocidad comprometida alcanzada en cada iteración. Nuestro foco simplemente estaba puesto en tomar las tareas sugeridas por la entrega en cuestión, y completarlas en tiempo y forma.

La segunda etapa del desarrollo del portal, que se inició una vez completada la asignatura "Java y Aplicaciones Avanzadas sobre Internet" se basó en una forma de trabajo que está más alineada con la metodología Scrum. El equipo de desarrollo se reorganizó con el equipo de El Paseo para determinar reuniones fijas y periódicas cada 15 días para mostrar el avance del desarrollo y recibir consultas/opiniones al respecto. En otras palabras, se podría decir que comenzamos a trabajar en sprints de 15 días, para los cuales nos comprometíamos a completar alguna funcionalidad específica dividida en tareas que debían llevarse a cabo dentro del alcance del sprint. Obviamente que se pueden generar los llamados *carry-overs* (tareas que no logran completarse en un sprint, y, por lo tanto, pasan al siguiente) pero esto sería un mal indicador si ocurriera con frecuencia.

Entonces, como preparación previa a cada sprint, una vez que se acordaba con el cliente qué funcionalidad se iba a abordar, se buscaba dividir el trabajo a realizar en subtareas pequeñas, estimadas en unos pocos puntos de esfuerzo. Esta estimación era informal, es decir, no seguía ningún lineamiento sugerido por la metodología Scrum (como por ejemplo la escala de Fibonacci) sino que era más un análisis para asegurar que la tarea no necesitara más de dos días de desarrollo aproximadamente. Si requería más, entonces procedíamos a subdividirla en tareas más pequeñas y acotadas.

Una de las diferencias con la teoría de la implementación de la metodología Scrum es que en nuestro equipo no había asignación de roles. Esto se daba principalmente porque el equipo de desarrollo estaba compuesto por una sola persona que se encargaba de las tareas de codificación, de planificar las tareas a realizar, estimarlas según su conocimiento y experiencia, y organizar las reuniones con el cliente (los stakeholders) para ir mostrando el avance del proyecto. De haberse tratado de un equipo de más integrantes, seguramente hubiera surgido la necesidad de contar con otros roles que se encarguen de garantizar que el trabajo en equipo se desarrolle de forma organizada, eficiente y basado en la mejora continua, como lo es el rol del *Scrum Master*. Además, hubiera sido necesaria la formalización de las ceremonias de planificación y refinamiento de las tareas para tener en cuenta las opiniones y experiencias de todos los integrantes del equipo, y el estableci-

miento de las reuniones diarias para organizar los esfuerzos del equipo sin afectar el trabajo del resto de los integrantes.

En conclusión, creemos que ajustamos el marco de trabajo según las necesidades específicas que surgían en cada fase del desarrollo. Limitarnos estrictamente a la teoría de una metodología ágil en particular no habría permitido alcanzar un flujo de trabajo organizado, eficiente, cómodo y flexible como el que logramos al fusionar lo mejor de ambas metodologías. Este enfoque nos llevó a trabajar en lo que podría denominarse **Scrumban**.

## 5.3. Versionado de código

El versionado de código es el proceso de gestionar y controlar las distintas versiones de un proyecto de software a lo largo del tiempo. Esto implica asignar identificadores a las distintas versiones del código fuente para distinguirlas unas de otras y poder rastrear los cambios realizados en cada una de ellas.

El versionado de código es crucial para el desarrollo de software colaborativo y para el mantenimiento de proyectos a largo plazo. Algunos de los sistemas más comunes son Git[6], Subversion (SVN)[7] o Mercurials (SCM)[8].

Los sistemas de control de versiones permiten a los desarrolladores trabajar de forma simultánea en diferentes partes del código, fusionar los cambios realizados por múltiples colaboradores y revertir los cambios no deseados en caso de que sea necesario. Además, proporcionan un historial detallado de todas las modificaciones realizadas en el código, lo que facilita la colaboración y la resolución de problemas, manteniendo la integridad y la coherencia del proyecto.

### 5.3.1. Git

Durante los primeros años del mantenimiento del Kernel de Linux (1991-2002) los cambios en el código se compartían entre los desarrolladores mediante "parches" o archivos. A partir de 2002 empezaron a utilizar un software de versionado de código privado denominado BitKeeper, pero ya en 2005, esta relación entre la comunidad a cargo del desarrollo del kernel y la

[6]Sitio oficial: `https://git-scm.com/`
[7]Sitio oficial: `https://subversion.apache.org/`
[8]Sitio oficial: `https://www.mercurial-scm.org/`

empresa se terminó repentinamente. Es por esto que la comunidad y principalmente Linus Trovalds[9], el creador de Linux, comenzaron a desarrollar su propia herramienta de versionado de código teniendo en cuenta las lecciones aprendidas a partir de la experiencia usando BitKeeper [23]. Algunos de los principales objetivos de este desarrollo eran:

- Garantizar velocidad.
- Que sea un desarrollo simple.
- Soportar desarrollo no linear con miles de ramas paralelas.
- Que sea totalmente distribuido.
- Capaz de soportar grandes proyectos de forma eficiente, como el proyecto del Kernel de Linux.

Así fue que nació Git, un sistema de control de versiones distribuido y de software libre que desde el 2005 evolucionó y maduró manteniendo estas cualidades, además de su destacada facilidad de uso. Hoy Git es una herramienta ampliamente utilizada para el desarrollo de software, con un increíble sistema de ramas que garantiza el desarrollo no linear, y una gran comunidad.

En este proyecto de tesina, utilizamos Git a partir de Gitlab, que es una plataforma de desarrollo colaborativo de software basada en Git.

Por otra parte, como dinámica de trabajo para el uso y organización del versionado de código del proyecto se adoptó el modelo de flujo de trabajo denominado Git Flow. Este modelo fue propuesto y popularizado por Vincent Driessen[10] en 2010 y se ha convertido en un enfoque popular para organizar el

[9]Linus Torvalds es un ingeniero de software finlandés conocido principalmente por ser el creador del kernel de Linux, uno de los sistemas operativos de código abierto más ampliamente utilizados en el mundo. Torvalds comenzó el desarrollo de Linux en 1991 mientras era estudiante de informática en la Universidad de Helsinki. Inspirado por el sistema operativo Unix, creó Linux como un proyecto personal y lo lanzó como software de código abierto bajo la Licencia Pública General de GNU (GPL).

[10]Vincent Driessen es un desarrollador de software conocido principalmente por su contribución al mundo del desarrollo colaborativo de software mediante la propuesta del modelo de flujo de trabajo Git Flow.

Driessen publicó por primera vez su artículo .ª successful Git branching model.en 2010, donde presentó el modelo Git Flow como un enfoque para organizar el proceso de desarrollo de software utilizando el sistema de control de versiones Git.

proceso de desarrollo en equipos. Este flujo de trabajo se basa en tener diferentes ramas para diferentes tipos de desarrollo, las cuales se van fusionando de manera sistemática.

Sus principales componentes son:

**Ramas principales:** Por un lado, se encuentra la rama *main* o *master*, la cual contiene el código estable y listo para ser desplegado en producción. Por otro lado, en la rama *develop* es donde se integran todas las funcionalidades completadas y se preparan para la próxima versión.

**Ramas de funcionalidades (feature branches):** Cada nueva funcionalidad o cambio en el código se desarrolla en su propia rama, que se deriva de la rama develop. Para la nomenclatura de la rama, por lo general se utiliza el prefijo *feature/* o *feat/* seguido de algún identificador corto de la funcionalidad en cuestión (por ejemplo, *feature/login-component*, en donde se desarrollaría la funcionalidad de inicio de sesión en el sistema). Una vez que la funcionalidad está completa, se fusiona nuevamente en la rama develop.

**Ramas de lanzamiento (release branches):** Cuando se está preparando una nueva versión para lanzamiento en producción, se crea una rama *release* a partir de la rama develop. En esta rama, se realizan tareas como pruebas finales, corrección de errores y documentación. Una vez que la versión está lista, se fusiona en master y develop, y se etiqueta con el número de versión correspondiente para identificarla fácilmente.

**Ramas de corrección (hotfix branches):** Si se detecta un error en la rama master que necesita ser corregido de inmediato, se crea una rama *hotfix* a partir de master. Una vez que se corrige el error, la rama se fusiona tanto en master como en develop para mantener los próximos desarrollos actualizados, y en la rama release en caso de que hubiera.

Entonces, en lugar de una única rama master, este flujo de trabajo utiliza dos ramas para registrar el historial del proyecto: la rama master almacena el historial de publicación oficial; mientras que la rama develop sirve como rama de integración para las nuevas funcionalidades del sistema. Asimismo, se recomienda etiquetar todas las confirmaciones o *commits* de la rama master con un número de versión incremental acorde a los cambios fusionados.

Todas las funciones nuevas deben residir en su propia rama, que se puede enviar al repositorio central (remoto) para copia de seguridad/colaboración. Sin embargo, en lugar de ramificarse de master, las ramas feature utilizan la rama develop como rama primaria, evitando así interactuar directamente con la rama master. Cuando una función está terminada, se vuelve a fusionar en develop.

Cuando develop haya adquirido suficientes funciones para una publicación (o se acerque una fecha de publicación predeterminada, por ejemplo), se debe bifurcar una rama release a partir de develop. Al crear esta rama, se inicia el siguiente ciclo de publicación, por lo que no pueden añadirse nuevas funciones una vez pasado este punto (en esta rama solo deben producirse las soluciones de errores, la generación de documentación y otras tareas orientadas a la publicación).

Cuando está lista para el lanzamiento, la rama release se fusiona en master y se etiqueta con un número de versión. Al finalizar, se elimina la rama release. Además, se deberá volver a fusionar en develop, ya que esta podría haber progresado desde que se iniciara la publicación. Este punto es importante, ya que podrían haberse añadido actualizaciones significativas a la rama release, y las funciones nuevas tienen que poder acceder a ellas. Además, este sería el lugar ideal para una solicitud de incorporación de cambios o "Pull Request" para garantizar que el equipo haga una revisión de código antes de mergear el código final.

Utilizar una rama específica para preparar publicaciones hace posible que un equipo perfeccione la publicación actual, mientras otro equipo sigue trabajando en las funciones para la siguiente publicación. Asimismo, crea fases de desarrollo bien definidas que ayudan a orientar y organizar al equipo de desarrollo (por ejemplo, es fácil decir: "esta semana nos estamos preparando para la versión 4.0" y verlo escrito en la estructura del repositorio).

Por último, las ramas de corrección o de hotfix sirven para reparar rápidamente las publicaciones de producción. Son muy similares a las ramas release y feature, salvo por el hecho de que se basan en la rama master y no en la develop (siendo así la única rama que debería bifurcarse directamente a partir de master). Cuando se haya terminado de aplicar la corrección, debería fusionarse en master y develop (o la rama release actual, si hubiera), y master debería etiquetarse con un número de versión estable actualizado.

Tener una línea de desarrollo específica para la corrección de errores permite que el equipo de desarrollo aborde las incidencias rápidamente sin interrumpir el resto del flujo de trabajo ni esperar al siguiente ciclo de publica-

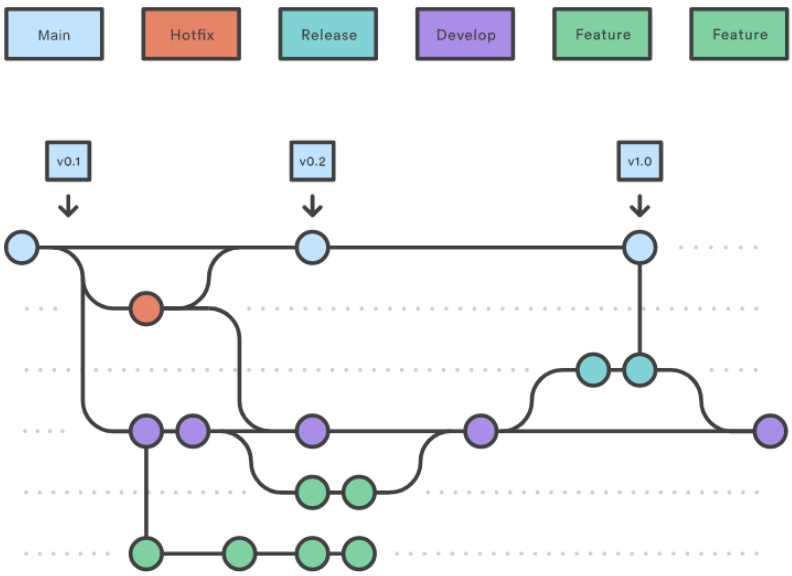


Figura 5.1: Flujo de trabajo basado en Git Flow, con el uso de las ramas master/main, hotfix, release, develop, y features.

ción.

En la figura 5.1 se puede apreciar gráficamente la organización y relación entre las ramas mencionadas anteriormente [24].

Por otro lado, también se acordó una nomenclatura específica para los mensajes de *commit* con el objetivo de garantizar mayor organización y claridad, y así poder detectar fácilmente qué cambio se introdujo en cada commit. La nomenclatura utilizada se basa en usar un prefijo preestablecido, el cual resume qué tipo de cambio es. Estos prefijos pueden ser: *feat* (para funcionalidades nuevas), *fix* (para arreglos de código), *refactor* (para hacer alusión a cambios que refactorizan un módulo), o *style* (para cambios en las plantillas de estilos).

Como conclusión, el flujo de trabajo Git Flow proporciona una estructura clara y predecible para el desarrollo de software, facilitando la colaboración en equipo, la gestión de versiones y la implementación de nuevas funcionalidades y correcciones de errores de manera controlada. Es especialmente útil en proyectos grandes y complejos con múltiples desarrolladores y lanzamientos

frecuentes.

Para este proyecto de tesina, fue clave para mantener el desarrollo del código ordenado, con la capacidad de volver a versiones anteriores siempre que sea necesario sin mayores inconvenientes, y también logrando una fácil identificación de las versiones estables del proyecto. Además, las fusiones a las ramas master y develop se hacían solo a partir de "Pull Requests" o solicitudes de incorporación de cambios, para garantizar que el código que llegue a estas ramas sea el esperado y esté revisado.

### 5.3.2. Versionado semántico

Como mencionamos anteriormente, a las ramas de release que se van creando y a las fusiones que se hacen en master se recomienda asignarles un número de versión para mantener el orden a lo largo del tiempo. Para versionar nuestro proyecto seguimos los lineamientos del versionado semántico, el cual es un enfoque estándar para asignar versiones a software, basado en un sistema de tres números separados por puntos: "X.Y.Z" [25].

El primer número (X) representa la versión principal, la cual se incrementa cuando se realizan cambios que rompen la compatibilidad hacia atrás con versiones anteriores, conocidos como *breaking changes*.

El segundo número (Y) representa la versión secundaria, la cual se incrementa cuando se agregan nuevas funcionalidades de una manera compatible con versiones anteriores.

Por último, el tercer número (Z) representa la versión de corrección o parche, la cual se incrementa cuando se realizan correcciones de errores que mantienen la compatibilidad con versiones anteriores.

Además, el versionado semántico permite agregar etiquetas adicionales para indicar versiones de desarrollo, como "alpha", "beta", o "rc" (haciendo referencia a *release candidate* o candidato a publicación).

Esta convención de versionado facilita la comprensión y la comunicación entre los desarrolladores y los usuarios sobre el estado y la naturaleza de una versión particular de un proyecto de software.

Para poner un ejemplo real, si un desarrollador está utilizando una dependencia en la versión v1.4.25 y desea hacer una actualización a la siguiente versión publicada que es la v2.0.0, solo con ver el nombre de la versión se dará cuenta de que la misma trae aparejados cambios que no mantienen retrocompatibilidad, por lo que la actualización implicará adaptar el código fuente con las nuevas especificaciones de la librería, para que el sistema siga funcionando

como se espera. Si la dependencia no usara versionado semántico, el desarrollador podría hacer la actualización sin darse cuenta que la misma implica que el código anterior ya no es compatible, y podría generar mayores errores en producción.

Para nuestros proyectos de frontend y backend utilizamos las mismas reglas basadas en versionado semántico: cuando se creaba una rama release la anotábamos con el nombre vX.Y.Z-rc, y una vez que se mergeaba la misma a master creábamos un *tag* de Git para guardar la referencia a ese punto específico en la historia del repositorio, el cual nombrábamos siguiendo la nomenclatura vX.Y.Z. De esta forma en las ramas release trabajábamos los release candidates para publicarlos, y una vez que estaban listos y en producción marcábamos la versión estable para que pueda ser referenciada fácilmente.

## 5.4. Tecnologías para el desarrollo del servidor

A continuación se describirán las tecnologías adoptadas para el desarrollo del servidor del portal de El Paseo. Se incluirán los lenguajes de programación, frameworks, librerías y bases de datos elegidos. Cabe destacar que todas las tecnologías adoptadas cuentan con una amplia comunidad de desarrolladores y una abundante documentación en línea, lo que facilitó la resolución de problemas y el aprendizaje de las mismas para implementarlas.

### 5.4.1. Java

El lenguaje de programación elegido fue Java[11], principalmente por su portabilidad, flexibilidad, y por ser orientado a objetos. Sus principales características son:

**Orientado a Objetos:** Sigue el paradigma de programación orientada a objetos: en Java todo es un objeto, y los programas se estructuran en clases y objetos que interactúan entre sí.

[11] Documentación oficial: `https://www.java.com/en/download/help/whatis_java.html`

**Portabilidad:** La plataforma Java se diseñó para ser independiente de la arquitectura y el sistema operativo. Los programas escritos en Java pueden ejecutarse en cualquier dispositivo que tenga una máquina virtual Java, lo que hace que sea un lenguaje de programación altamente portátil.

**Compilación e Interpretación:** El código fuente se compila en un formato intermedio llamado *bytecode*, que es ejecutado por la máquina virtual Java. Esto permite que los programas sean ejecutados en cualquier sistema que tenga una máquina virtual Java, sin necesidad de recompilación.

**Robusto y Seguro:** Incluye características diseñadas para hacer que los programas sean robustos y seguros, como la gestión automática de la memoria (a través del recolector de basura), verificación de límites de array y chequeo de tipos en tiempo de compilación.

**Dinámico y Estático:** Java es un lenguaje estáticamente tipado, lo que significa que los tipos de datos se deben declarar antes de su uso. Sin embargo, también incluye elementos dinámicos, como la reflexión, que permite la inspección y manipulación de clases y objetos en tiempo de ejecución.

**Gestión Automática de la Memoria:** Java utiliza un recolector de basura para gestionar automáticamente la liberación de memoria, evitando así muchos problemas asociados con la gestión manual de la memoria.

### 5.4.2. Jersey

Jersey[12] es un framework utilizado para la creación de servicios web RESTful basados en la especificación JAX-RS[13]. Es la implementación de referencia de JAX-RS y se utiliza comúnmente en proyectos Java para construir APIs REST. Con el uso de este framework logramos simplificar el desarrollo a partir de las anotaciones y API que provee. Sus principales características son:

[12]Documentación oficial: `https://eclipse-ee4j.github.io/jersey/`
[13]JAX-RS por sus siglas en inglés: Java API for RESTful Web Services

- Facilita la creación de servicios web RESTful al permitir a los desarrolladores definir recursos, rutas, y operaciones HTTP de manera sencilla, estructurada y declarativa, a partir de distintas anotaciones. Estas anotaciones resuelven tareas comunes, como el enrutamiento de URI, la manipulación de parámetros y la serialización/deserialización de datos, permitiendo a los desarrolladores centrarse en la lógica de la aplicación en lugar de la infraestructura subyacente. Entre las anotaciones mas usadas, podemos destacar:
  - **@Path**: Se utiliza para especificar la URI base de un recurso o un método de un recurso. Puede aplicarse a clases y métodos.
  - **@GET, @POST, @PUT, @DELETE,** etc.: Se utilizan para especificar el método HTTP que un recurso debe manejar. Se aplican a métodos dentro de una clase de recurso.
  - **@QueryParam**: Utilizada para extraer valores de parámetros de consulta de la URI.
  - **@PathParam**: Se utiliza para extraer valores de segmentos de la URI.
  - **@Consumes** y **@Produces**: Se utilizan para especificar los tipos de datos aceptados y producidos por un recurso.
- Proporciona soporte para la serialización y deserialización de datos en formatos comunes como JSON y XML. Esto facilita la transferencia de datos entre clientes y servicios RESTful.
- Permite la creación de interceptores y filtros que pueden ser utilizados para manipular las peticiones y respuestas HTTP antes de que lleguen a los recursos finales. Esto proporciona flexibilidad en la manipulación y validación de datos.

Este framework lo conocimos durante la cursada de la asignatura "Java y Aplicaciones Avanzadas sobre Internet", y podemos concluir que su adopción fue muy fluida y sin complicaciones significativas. En cuestión de días ya contábamos con la API base de nuestro backend en pleno funcionamiento, con todos los endpoints necesarios para realizar todas las operaciones CRUD de nuestras entidades.

El uso de un framework para el desarrollo de una API Rest nos permite evitar gestionar las configuraciones manualmente como el manejo y definición de las rutas de URL, la serialización/deserialización de los datos, o la

configuración del servidor, entre otras. Usando Jersey logramos acelerar el proceso de desarrollo al delegar todas estas configuraciones al framework.

### 5.4.3. HK2

HK2[14] es un framework ligero de inyección de dependencias y servicios en Java. La inyección de dependencias es una técnica de diseño de software para lograr una mayor modularidad y desacoplamiento en las aplicaciones. Fue desarrollado por Oracle y se utiliza principalmente en el contexto de aplicaciones Java empresariales en conjunto con el framework Jersey, especialmente en el ámbito de Java EE[15] y microservicios.

HK2 proporciona un sistema de gestión de servicios que permite registrar, localizar y consumir servicios en una aplicación. Esto facilita la creación de componentes reutilizables y modulares que pueden ser compartidos entre diferentes partes de una aplicación, mejorando así la gestión de dependencias entre componentes y promoviendo la reutilización del código. El framework provee anotaciones simples para configurar la inyección de dependencias, la gestión de componentes y el control del ciclo de vida de los mismos en la aplicación. Entre ellas, podemos mencionar:

- **@Service**: Marca una clase como un servicio, los cuales son componentes gestionados por HK2 y pueden ser inyectados en otras clases.
- **@Inject**: Indica a HK2 que inyecte una dependencia en un campo, método o constructor.
- **@Contract**: Define un contrato (interfaz o clase abstracta) que puede ser implementado por servicios.
- **@Scope**: Define el ámbito de vida de un componente. Puede ser utilizado para controlar cuándo se crea y se destruye un componente.
- **@Factory**: Marca un método de fábrica que puede ser utilizado para crear instancias de un componente.

En este proyecto, aprovechamos las ventajas de esta tecnología para que la inyección de dependencias sea gestionada automáticamente, lo que nos

[14] HK2 por sus siglas en inglés: Hundred Kilobytes Kernel. Documentación oficial: `https://javaee.github.io/hk2/`

[15] EE por sus siglas en inglés: Enterprise Edition

permitió entre otras cosas mejorar la modularidad y la capacidad de prueba de la aplicación. Además, el código se volvió notablemente más limpio y fácil de entender, ya que desacoplamos la creación de objetos de la lógica de negocio.

### 5.4.4. Swagger

Swagger[16] es un conjunto de herramientas que permite documentar, diseñar, construir y consumir servicios web RESTful. Inicialmente, se centró en la documentación de APIs REST, proporcionando una interfaz legible por humanos para describir y entender la funcionalidad de los servicios web, a través de un lenguaje de descripción de API que permite a los desarrolladores describir los recursos, operaciones y modelos de datos utilizados por su API. Además, provee una interfaz de usuario basada en web llamada Swagger UI que permite a los usuarios visualizar y probar APIs directamente desde la documentación generada, lo cual facilita y agiliza la integración de los desarrolladores con diferentes APIs. Swagger fue luego evolucionando a un estándar más amplio llamado OpenAPI[17], el cual se usa para describir y documentar servicios web RESTful.

En la figura 5.2 se puede apreciar parte de la documentación generada automáticamente por Swagger para nuestro bakend: a la izquierda el listado de algunos de los endpoints disponibles, y a la derecha una especificación de un endpoint en particular, indicando parámetros, request body, tipos de respuesta, etc.

El uso de esta herramienta de documentación facilita la colaboración entre distintos equipos. Actualmente, es muy común encontrar equipos integrados por desarrolladores backend y frontend. En un proyecto como este en donde se está desarrollando una aplicación web basada en una API Rest, los primeros se dedican a implementar los servicios que conforman la API, mientras que los segundos integran el cliente consumiendo estos servicios. En este tipo de equipos colaborativos, es esencial el uso de herramientas como Swagger, ya que permite que los desarrolladores frontend obtengan rápidamente la especificación de la API en donde acceden a los esquemas de los datos de entrada de cada servicio, parámetros requeridos, tipos de respuesta, errores esperados, etc. Es así que los equipos de frontend y backend pueden trabajar

[16]Documentación oficial: `https://swagger.io/tools/swagger-ui/`
[17]Documentación oficial: `https://www.openapis.org/`

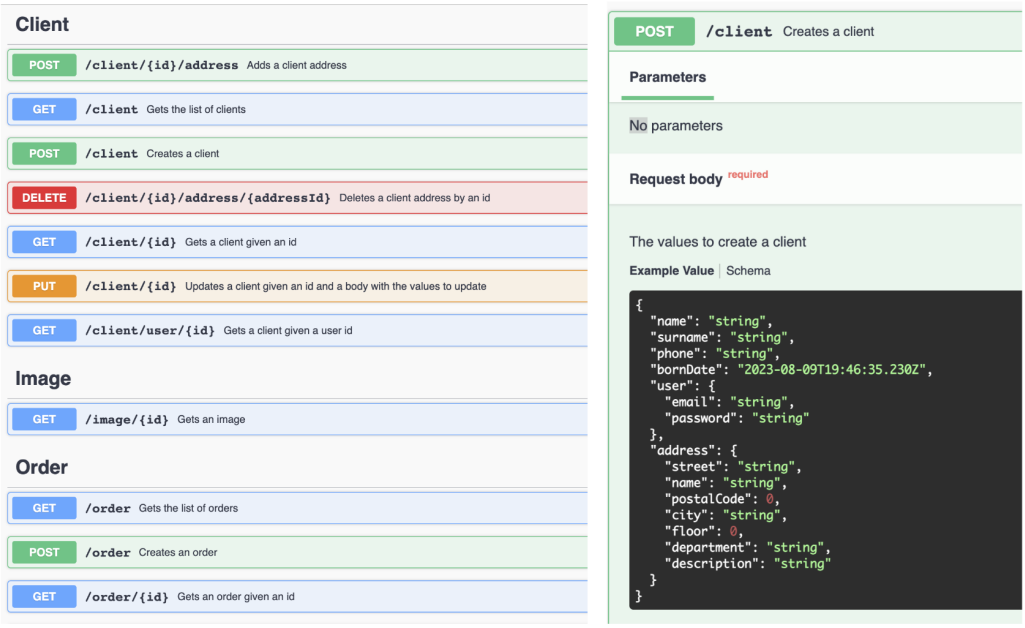


Figura 5.2: Documentación generada por Swagger y presentada por la interfaz Swagger UI

de manera más eficiente e independiente al tener una comprensión común de la API y sus capacidades.

Además, al utilizar Swagger, los desarrolladores del frontend pueden estar al tanto de los cambios en la API de manera más rápida y precisa: cualquier modificación en la definición de la API se reflejará inmediatamente en la documentación, evitando posibles desincronizaciones entre el frontend y el backend.

Aunque en este proyecto el equipo de desarrollo contaba con una sola persona para la codificación, el uso de Swagger resultó ser de gran utilidad. Dado que el frontend se desarrolló después del backend, Swagger permitió consumir los endpoints de la API de manera rápida, clara y sin mayores complicaciones.

### 5.4.5. Maven

Apache Maven[18] es una herramienta de gestión de proyectos y construcción de software basados en Java. Sus principales características y funcionalidades son:

**Gestión de Dependencias:** Gestiona automáticamente las dependencias del proyecto, descargando las bibliotecas y componentes necesarios de repositorios remotos y locales.

**Ciclo de Vida del Proyecto:** Define un conjunto de fases predefinidas y objetivos para la construcción y gestión de proyectos. Estas fases incluyen, entre otras, compilación, prueba, empaquetado, instalación y despliegue, y se ejecutan mediante comandos predefinidos, lo que automatiza la compilación, ejecución de pruebas y empaquetado.

**Plugins:** Utiliza plugins para extender sus capacidades. Hay plugins disponibles para diversas tareas, como generar documentación, ejecutar pruebas, desplegar aplicaciones, entre otras.

**Repositorios:** Utiliza repositorios para almacenar y gestionar dependencias y artefactos de proyectos. Es posible descargar dependencias desde repositorios remotos y también publicar artefactos en repositorios locales o remotos.

[18] Documentación oficial: `https://maven.apache.org/`

**Proyecto Basado en Configuración:** Utiliza un archivo de configuración llamado **pom.xml**[19] que define la configuración del proyecto, las dependencias, los plugins y otros detalles de construcción de forma centralizada. Las dependencias se describen en este archivo, y Maven se encarga de descargarlas automáticamente desde repositorios centralizados, simplificando así la resolución de dependencias.

**Escalabilidad:** Es ampliamente utilizado en proyectos de diversos tamaños y complejidades, desde pequeñas aplicaciones hasta grandes proyectos empresariales.

**Estructura del proyecto:** Impone una estructura de proyecto estándar. Los directorios como **src/main/java** para el código fuente, **src/test/java** para las pruebas, y otros, son convenciones que facilitan la navegación y organización del proyecto.

Sin algún gestor de paquetes, todas estas tareas se vuelven manuales, lo que naturalmente complejiza y relentiza el desarrollo de la aplicación.

### 5.4.6. MySQL

El sistema de gestión de bases de datos elegido fue MySQL[20], el cual es relacional y de código abierto. Utiliza el lenguaje de consulta estructurado llamado SQL[21] para gestionar bases de datos e interactuar con las mismas.

Entre sus características principales destacamos las propiedades ACID, las cuales garantizan la consistencia de los datos en las transacciones[26]:

- Atomicidad: Garantiza que una transacción se realice completamente o no se realice en absoluto. En otras palabras, todas las operaciones dentro de una transacción se ejecutan de manera indivisible: si alguna de ellas falla, la transacción se revierte al estado anterior a la ejecución de la transacción.

- Consistencia: Asegura que una transacción lleve la base de datos de un estado válido a otro. Si una transacción viola las reglas de integridad de la base de datos, se revierte y la base de datos se mantiene en un estado consistente.

[19] POM por sus siglas en inglés: Project Object Model
[20] Documentación oficial: `https://www.mysql.com/`
[21] SQL por sus siglas en inglés: Structured Query Language

- Aislamiento: Garantiza que una transacción en curso sea invisible para otras transacciones hasta que se complete. Cada transacción se ejecuta como si fuera la única en el sistema, independientemente de otras transacciones concurrentes, evitando interferencias y problemas de consistencia.

- Durabilidad: Garantiza que una vez que una transacción se ha completado con éxito, sus cambios persisten incluso en caso de falla del sistema, reinicios o apagones: los cambios realizados por una transacción confirmada son permanentes y no se perderán.

Además, permite la creación de índices para mejorar el rendimiento de las consultas. Un índice es una estructura que mejora la velocidad de recuperación de datos de una tabla en función de los valores de una o más columnas. Su propósito principal es acelerar el proceso de búsqueda y recuperación de registros, reduciendo así el tiempo necesario para realizar consultas. Pueden existir varios tipos de índices, como índices simples (basados en una sola columna) o índices compuestos (basados en múltiples columnas). También hay índices únicos que garantizan que no haya duplicados en la columna o conjunto de columnas indexadas.

Dado el extenso uso de esta tecnología y la participación activa de su comunidad, junto con mi experiencia previa positiva como desarrolladora al utilizarla, se puede afirmar que la elección de incorporarla para gestionar la base de datos simplificó la fase de implementación de la aplicación.

### 5.4.7. Java Persistence API y Hibernate

Java Persistence API (en adelante, JPA) es una especificación de Java que describe una interfaz común para el mapeo objeto-relacional (ORM[22]) en aplicaciones Java. JPA simplifica el acceso a bases de datos relacionales y permite a los desarrolladores interactuar con las mismas utilizando objetos Java en lugar de escribir consultas SQL directamente.

Hibernate[23], por su parte, es una implementación popular y ampliamente utilizada de la especificación JPA. Utilizándolo como proveedor de JPA, los desarrolladores pueden aprovechar las características de JPA para interactuar con bases de datos de manera más eficiente y orientada a objetos,

[22]ORM por sus siglas en inglés: Object Relational Mapping
[23]Documentación oficial: `https://hibernate.org/`

eliminando la interacción directa con la base de datos. Además, utiliza un lenguaje de consulta orientado a objetos llamado HQL[24] que permite a los desarrolladores realizar consultas utilizando la sintaxis de objetos en lugar de SQL, facilitando la manipulación de datos y evitando la exposición directa a detalles de la base de datos, impactando positivamente en la productividad de los desarrolladores.

Por otra parte, simplifica la gestión de transacciones al proporcionar soporte para transacciones declarativas y gestionando automáticamente el inicio, confirmación y reversión de las mismas.

### 5.4.8. Flyway

Flyway[25] es una herramienta de migración de bases de datos que se utiliza para gestionar y controlar los cambios en su estructura a lo largo del tiempo. Permite a los desarrolladores aplicar y gestionar de manera controlada los scripts de migración de bases de datos, lo que garantiza que los cambios en su estructura se apliquen de manera consistente y reproducible en diferentes entornos (como desarrollo, pruebas y producción), y con la posibilidad de revertirlos en caso de que sea necesario.

Las herramientas de migración de bases de datos son muy importantes, ya que ayudan a mantener un control preciso y organizado de las versiones de la base de datos, lo que le facilita notablemente el trabajo a los desarrolladores.

Por otro lado, a través de las migraciones automatizadas se garantiza que los cambios a la estructura de la base de datos se apliquen de manera consistente y completa en todos los entornos. Si se encuentra un error en la ejecución de la migración, los cambios se revierten automáticamente, dejando a la base de datos en un estado consistente, ayudando así a mantener la integridad a lo largo del ciclo de vida de la aplicación.

Además, esta tecnología se usó para hacer una carga inicial de datos en la base de datos, ya que el sistema requería de ciertas configuraciones inicializadas a un valor por defecto cuando iniciaba la aplicación.

[24] HQL por sus siglas en inglés: Hibernate Query Language
[25] Documentación oficial: `https://flywaydb.org/`

### 5.4.9. JWT

JWT[26] es un estándar abierto (RFC 7519) que define un formato compacto y autónomo para la representación de información entre dos partes de manera segura mediante objetos JSON. Son comúnmente utilizados para la autenticación y la transferencia segura y eficiente de información entre sistemas.

Un token JWT consta de tres partes separadas por puntos: el encabezado, el cual contiene información sobre cómo se debe procesar el token; la carga útil o payload, el cual contiene la información que se desea transmitir entre las partes; y la firma, la cual se utiliza para verificar que el remitente del JWT es quien dice ser y para garantizar que la carga útil no ha sido alterada.

Además, uno de los beneficios clave de JWT es que se puede incluir una fecha de expiración en la carga útil del token, lo que proporciona una capa adicional de seguridad al garantizar que el token sea válido solo por un tiempo limitado.

El manejo de la fecha de expiración es responsabilidad del servidor que genera el JWT y del cliente que lo consume: en el lado del servidor se debe verificar la fecha de expiración antes de procesar el token, mientras que en el lado del cliente se debe manejar adecuadamente la expiración y actuar en consecuencia (renovar el token o solicitar uno nuevo).

En este proyecto, usamos los tokens de JWT para las siguientes funciones:

**Autenticación:** Luego de que un usuario se autentica correctamente ingresando email y contraseña, el servidor emite un JWT que contiene información sobre el usuario y su rol asignado. Este token se envía al cliente, el cual lo incluye en las solicitudes subsiguientes para demostrar su identidad. Es decir, en cada petición que le envía al servidor en nombre del usuario loggeado, incluye el token obtenido en el encabezado "Authorization".

**Intercambio Seguro de Información:** Debido a su naturaleza autónoma y compacta, los JWT son utilizados para el intercambio seguro de información entre sistemas. Los sistemas pueden confiar en la integridad y autenticidad de la información contenida en un JWT.

[26] JWT por sus siglas en inglés: JSON Web Token. Documentación oficial: `https://jwt.io/`

**Autorización y Control de Acceso:** Pueden contener información sobre los permisos y roles de un usuario. Esto se utiliza para tomar decisiones de autorización y control de acceso en los sistemas. En nuestro caso, los tokens contienen la información de los roles de cada usuario, y esta información se usa para filtrar el acceso a determinados endpoints de nuestro backend. Por ejemplo, para acceder a las rutas del recurso **/administrador**, es necesario que el token que llega a partir de la petición HTTP contenga el rol ADMINISTRADOR. En caso contrario, el servidor responde con el código de error 401 DESAUTORIZADO.

### 5.4.10. Filtros

Los filtros en Java son componentes que interceptan y procesan las peticiones y respuestas HTTP antes de que lleguen a los recursos finales. Se pueden utilizar para realizar tareas como la autenticación, la autorización, la compresión, la manipulación de encabezados, y más. En este proyecto, creamos un filtro para todos los endpoints de nuestra API que son de uso privado y restringido, es decir, solo usuarios registrados en el sistema pueden acceder. Este filtro se encarga de obtener el token JWT proveniente del encabezado "Authorization" de la petición, desencodearlo y obtener así la información del usuario que está realizando la solicitud. Si no es un usuario registrado, el filtro devuelve un error con código HTTP 401 DESAUTORIZADO. Además, creamos filtros para cada uno de los roles disponibles, para poder filtrar que los usuarios que accedan a ciertos endpoints contengan un rol en particular, también desencodeando el JWT recibido en la petición.

En el fragmento de código a continuación, dejamos la definición del filtro *JWTFilter* mencionado anteriormente. Los filtros en Java deben implementar la interface *Filter* la cual declara el método *doFilter*, que es el encargado de aplicar las reglas de filtro. Si las reglas declaradas son evaluadas correctamente, se llama al método *chain.doFilter(request, response)* para pasar la petición al siguiente filtro de la cadena:

```
1 public class JWTFIlter extends HttpFilter implements
     Filter {
2   private JWTService jwtService = new JWTService();
3
4   public void doFilter(ServletRequest request,
       ServletResponse response, FilterChain chain)
5       throws IOException, ServletException {
```

```
6       HttpServletRequest httpRequest = (HttpServletRequest
            ) request;
7       String url = httpRequest.getRequestURL().toString();
8
9       if (!url.contains("auth")) {
10        String token = httpRequest.getHeader("
              Authorization");
11        try {
12          Jwt jwtTokenInformation = this.jwtService.
                parseJwtToken(token);
13          Claims jwtBody = (Claims) jwtTokenInformation.
                getBody();
14          request.setAttribute("role", jwtBody.get("role")
                );
15          System.out.println("Authorized request");
16        } catch (Exception e) {
17          System.out.println("Unauthorized request");
18          throw new UnauthorizedException();
19        }
20      }
21
22      chain.doFilter(request, response);
23    }
24  }
```

### 5.4.11. Otras librerías

**Lombok:** Lombok[27] ayuda a reducir la verbosidad del código fuente y mejora la legibilidad al automatizar la generación de ciertos métodos y constructores comunes. Fue diseñada para simplificar el desarrollo eliminando la necesidad de escribir código repetitivo como los constructores, *getters* y *setters*, *equals* y *hashCode*, etc. Además, simplifica la creación de instancias de loggers utilizando la anotación @Slf4j para integrar automáticamente un logger SLF4J en la clase.

**Gson:** Desarrollada por Google, Gson[28] se encarga de facilitar la conversión de objetos Java en su representación JSON y viceversa.

[27] Documentación oficial: https://projectlombok.org/
[28] Documentación oficial: https://github.com/google/gson

**Mapstruct:** Mapstruct[29] simplifica la implementación de mapeos entre objetos Java. Su objetivo principal es generar automáticamente el código necesario para realizar conversiones entre clases DTO y entidades de dominio, eliminando así la necesidad de escribir manualmente el código de mapeo.

**Java JSON Web Token:** Jjwt[30] se utiliza para trabajar con tokens JWT en Java.

## 5.5. Tecnologías para el desarrollo del cliente

A continuación se describirán las tecnologías adoptadas para el desarrollo del cliente del portal de El Paseo. Se incluirán los lenguajes de programación, frameworks y librerías elegidos. Al igual que se mencionó en el apartado de tecnologías adoptadas para el servidor, las que se eligieron para el desarrollo del cliente también cuentan con una amplia comunidad de desarrolladores y abundante documentación disponible en línea, lo que facilitó la resolución de problemas y el aprendizaje de las mismas para implementarlas.

### 5.5.1. JavaScript

El lenguaje de programación elegido para la codificación del cliente fue JavaScript[31], un lenguaje interpretado, dinámico y débilmente tipado, multiplataforma, event-driven, asíncrono y diseñado para la web:

**Lenguaje Interpretado:** El código fuente es ejecutado directamente por un motor JavaScript en un entorno de ejecución (como un navegador web) sin necesidad de compilación previa.

**Dinámico y Débilmente Tipado:** Es un lenguaje dinámico, ya que las variables pueden cambiar de tipo durante la ejecución del programa. Además, es débilmente tipado, lo que permite operaciones entre tipos de datos diferentes sin requerir conversiones explícitas.

[29] Documentación oficial: `https://mapstruct.org/`
[30] Documentación oficial: `https://github.com/jwtk/jjwt`
[31] Documentación oficial: `https://developer.mozilla.org/es/docs/Web/JavaScript`

**Multiplataforma:** Inicialmente diseñado para ejecutarse en navegadores web, JavaScript ahora es utilizado en una variedad de entornos, incluyendo servidores (Node.js) y aplicaciones de escritorio (Electron). Esto hace que JavaScript sea un lenguaje multiplataforma.

**Event-Driven y Asíncrono:** JavaScript es conocido por su modelo de ejecución asíncrona y basada en eventos. Esto es fundamental para manejar la interactividad en los navegadores, donde las acciones del usuario y las operaciones de red pueden ocurrir de manera concurrente.

**Diseñado para la Web:** Inicialmente creado para mejorar la interactividad en el navegador, JavaScript se ha convertido en el lenguaje de programación principal para el desarrollo web. Permite la manipulación dinámica del contenido de las páginas web y la interacción con el usuario.

### 5.5.2. TypeScript

TypeScript[32] es un lenguaje de programación de código abierto desarrollado y mantenido por Microsoft. Se describe a menudo como un "superset" de JavaScript, lo que significa que cualquier código JavaScript válido también es código TypeScript, pero TypeScript agrega funcionalidades adicionales y mejoras:

**Tipado Estático:** Introduce el concepto de tipado estático, lo que significa que es posible declarar el tipo de una variable, propiedad o función. Esto ayuda a detectar errores en tiempo de compilación y a mejorar la robustez del código.

**Tipos de Datos:** Incluye tipos de datos primitivos como number, string, boolean, etc., así como la capacidad de definir tipos de datos personalizados mediante interfaces y tipos.

**Inferencia de Tipos:** Aunque se puede declarar tipos explícitamente, TypeScript también es capaz de inferir automáticamente los tipos en muchas situaciones, lo que reduce la necesidad de redundancia en la escritura del código.

[32] Documentación oficial: `https://www.typescriptlang.org/`

**Clases y Herencia:** Admite programación orientada a objetos con la definición de clases, herencia, interfaces y módulos.

**Compilación a JavaScript:** Aunque se escribe en TypeScript, el código se compila a JavaScript para su ejecución en entornos de tiempo de ejecución JavaScript.

### 5.5.3. Angular

Como se mencionó previamente en este informe, la tecnología seleccionada para el desarrollo del cliente fue Angular[33]. Con esta tecnología es posible desarrollar aplicaciones denominadas "Single Page Application", es decir, basadas en una página web dinámica donde la mayoría del contenido se carga en una sola página y las actualizaciones se realizan de manera dinámica. Así se podrá lograr una experiencia de usuario más fluida y satisfactoria.

Angular es un framework de desarrollo de aplicaciones web desarrollado y mantenido por Google, que facilita la creación de aplicaciones del lado del cliente mediante el uso de TypeScript. Su estructura se basa en componentes, los cuales representan unidades autocontenidas de funcionalidad y presentación. De esta forma, el framework organiza la interfaz de usuario de una aplicación en una jerarquía de componentes, donde cada componente consiste de:

- Un archivo TypeScript que representa la clase del componente, donde se define su lógica, propiedades, métodos y las respuestas a determinados eventos. Además, en este archivo se define el ciclo de vida del componente, del cual se hablará más adelante.
- Un archivo HTML que representa la plantilla del componente, donde se define la estructura HTML y cómo se mostrará la interfaz del usuario.
- Un archivo con extensión .css o .scss donde se definen las reglas de estilo del componente.

Esta arquitectura basada en componentes permite que se organice el proyecto en distintas partes independientes, manejables y ordenadas con responsabilidades específicas. Como consecuencia, el código del proyecto tendrá una buena base para ser mantenible y escalable.

[33]Documentación oficial: `https://angular.io/`

Por otro lado, cuando se necesita compartir lógica entre distintos componentes, Angular te permite crear un servicio para inyectar código en distintos componentes, manejando el mismo desde una sola fuente de verdad.

Las principales características de Angular son:

**Arquitectura Modular:** Promueve una arquitectura modular, dividiendo la aplicación en módulos que se pueden cargar de manera independiente. Esto facilita el mantenimiento y la escalabilidad de las aplicaciones.

**Componentes:** Angular se basa en el concepto de componentes, los cuales son bloques de construcción fundamentales que contienen lógica, plantillas y estilos. Permiten centralizar y organizar una funcionalidad específica.

**Inyección de Dependencias:** Luego estos componentes pueden ser inyectados fácilmente en cualquier parte de la aplicación, ya que Angular utiliza un sistema de inyección de dependencias que facilita la gestión y la organización de componentes y servicios dentro de la aplicación.

**Directivas:** Permiten extender y modificar el DOM de manera declarativa. Hay directivas integradas y la posibilidad de crear personalizadas. Entre ellas podemos mencionar como ejemplo las siguientes:

- **ngIf**: Para mostrar u ocultar elementos según una expresión booleana.
- **ngFor**: Para realizar un bucle sobre una colección y renderizar elementos repetidos.
- **ngClass**: Permite agregar o eliminar clases de estilos de un elemento basado en una expresión.

**HTTP Client:** Angular proporciona un módulo HTTP Client que facilita la realización de solicitudes HTTP y la manipulación de respuestas.

**Enrutamiento:** Tiene un poderoso sistema de enrutamiento que permite la navegación entre diferentes vistas y la carga dinámica de contenido según la URL. Provee una navegación del lado del cliente y enrutamiento basado en los componentes de Angular.

**Testing:** Está diseñado para ser fácilmente testeable. Se incluyen herramientas para la escritura de pruebas unitarias y de integración.

**Manejo nativo de formularios:** Provee un sistema unificado para trabajar con formularios de forma simplificada e intuitiva, con la posibilidad de validarlos utilizando los validadores más usados (como por ejemplo, los formularios que son requeridos, los que solo admiten datos numéricos, o los que requieren de un email válido, entre otros).

**Angular CLI:** La Interface de Línea de Comandos de Angular es una herramienta que simplifica el proceso de desarrollo, creación y mantenimiento de proyectos Angular. Proporciona una serie de comandos que simplifican tareas comunes durante el ciclo de vida de desarrollo de la aplicación. Algunos de los comandos más utilizados incluyen la creación de nuevos componentes, servicios, módulos, y la construcción de la aplicación para producción:

- **ng new nombre-del-proyecto**: Crea un nuevo proyecto Angular con el nombre nombre-del-proyecto y con la estructura de carpetas y archivos iniciales.
- **ng generate component nombre-del-componente**: Crea un nuevo componente Angular con el nombre nombre-del-componente y con sus archivos asociados (archivo de componente, plantilla, estilo y pruebas unitarias).
- **ng generate service nombre-del-servicio**: Crea un nuevo servicio Angular con el nombre nombre-del-servicio junto con su archivo de pruebas unitarias.
- **ng serve**: Inicia el servidor de desarrollo y permite ejecutar la aplicación Angular localmente. La aplicación estará disponible en `http://localhost:4200/` por defecto.
- **ng build –prod**: Compila y construye la aplicación para producción. Los archivos resultantes estarán ubicados dentro de la carpeta *dist/*.
- **ng test**: Ejecuta las pruebas unitarias de la aplicación.

Como se mencionó anteriormente, en Angular los componentes pasan por una serie de eventos durante su ciclo de vida, desde la creación hasta la destrucción. Estos eventos se conocen como “hooks” y proporcionan puntos de intervención en el ciclo de vida del componente, lo que permite realizar acciones específicas en momentos clave.

El ciclo de vida de un componente comienza cuando Angular lo instancia y renderiza su vista junto con las vistas de sus componentes hijos, si los tuviera. Así, se continúa con la detección de los cambios a medida que Angular detecta cuando los datos enlazados cambian, y actualiza tanto la vista como la instancia del componente según se necesita. Este ciclo de vida finaliza cuando el framework destruye la instancia del componente y remueve su template renderizado del DOM.

Así es que luego de que la aplicación llama al constructor de un componente y lo instancia, Angular se encarga de llamar en el punto exacto de su ciclo de vida a los métodos hook que se implementaron, respetando el siguiente orden:

**ngOnChanges:** Se invoca cuando un componente tiene cambios en las propiedades de entrada (declarados mediante la directiva *@Input*). Típicamente, este método se utiliza para realizar acciones basadas en cambios en estas propiedades.

**ngOnInit:** Se invoca después de que Angular ha inicializado todas las propiedades del componente. Típicamente, se usa para inicializar datos, realizar llamadas a servicios y configurar el estado inicial del componente.

**ngDoCheck:** Se invoca durante cada ciclo de detección de cambios. Típicamente, es usado para aplicar lógica de detección de cambios personalizada.

**ngAfterContentInit:** Se invoca después de que el contenido proyectado (el contenido que se pasa al componente a través de las etiquetas *ng-content* ) ha sido inicializado. Típicamente, se usa para realizar acciones que dependen del contenido que se proyecta en el componente.

**ngAfterContentChecked:** Se invoca después de cada chequeo del contenido proyectado, y por lo general se usa para realizar acciones adicionales después de que este contenido ha sido verificado.

**ngAfterViewInit:** Se invoca después de que la vista del componente (y las vistas hijas) ha sido inicializada. Típicamente, se usa para realizar acciones que dependen de la vista del componente.

**ngAfterViewChecked:** Se invoca después de cada chequeo de la vista del componente. Típicamente, se usa para realizar acciones adicionales después de que la vista del componente ha sido verificada.

**ngOnDestroy:** Se invoca justo antes de que el componente sea destruido. Típicamente, se usa para limpiar recursos, cancelar suscripciones a observables y realizar otras tareas de limpieza.

Por otro lado, Angular provee un manejo nativo de los denominados *Interceptors*, los cuales se utilizan para inspeccionar, controlar y manejar peticiones HTTP desde la aplicación hacia un servidor. Los mismos interceptors también pueden transformar las respuestas de los servidores cuando vuelven hacia la aplicación. Sin esta herramienta, los desarrolladores deberían administrar esto de forma explícita en el código cada vez que se llame a algún método del cliente HTTP.

El uso más frecuente de los interceptors es para resolver tareas de autenticación o loggeo de mensajes informativos o de error. En nuestro caso, desarrollamos el *AuthInterceptor* que se encarga de interceptar cada petición saliente de nuestra aplicación de Angular para inyectarles el token JWT que el backend utiliza y requiere para autenticar las peticiones. Entonces, si no tuviéramos este interceptor, deberíamos inyectar el token JWT explícitamente cada vez que se envía una petición al backend (lo cual no solo genera código repetido, sino que también afecta la experiencia de codificación del desarrollador).

En el siguiente extracto de código se puede ver la forma en la que se define un interceptor en Angular, utilizando como ejemplo el *AuthInterceptor* mencionado anteriormente.

```
1  @Injectable()
2  export class AuthInterceptor implements HttpInterceptor
       {
3
4    constructor(private authService: AuthService) {}
5
6    intercept(request: HttpRequest<unknown>, next:
         HttpHandler): Observable<HttpEvent<unknown>> {
7      const authToken = this.authService.getToken();
8
9      const authReq = request.clone({
```

```
10        headers: request.headers.set('Authorization', `
              Bearer ${authToken}`)
11      });
12
13      return next.handle(authReq);
14    }
15  }
```

Además, Angular provee una herramienta denominada *Guards* para manejar y controlar la navegación entre rutas de la aplicación. En nuestro proyecto, se utilizaron para prevenir acceso desautorizado a ciertas rutas, y también para resolver tareas de autenticación en cada ruta. Por ejemplo:

- Se creó una función de guard para cada rol de nuestro sistema, es decir, para los roles administrador, consumidor y productor. Para las rutas a las que solo podía acceder un rol en particular, se usó la función guard correspondiente a ese rol para chequear que solo usuarios que tengan ese rol asignado puedan acceder a esas rutas. Por ejemplo, para las rutas administrativas de creación o edición de productos (**/admin/product/create** y **/admin/product/edit**) se usó la función *canActivateAdminLoggedIn* para que solo administradores tuvieran acceso a estas rutas. En caso de que el usuario que intente acceder a ellas no tenga el rol de administrador, el guard se encargará de bloquearle el acceso y redireccionarlo a la página de inicio de sesión.

- Se creó una función de guard que verifica que solo el usuario loggeado pueda acceder a una ruta en particular. Por ejemplo, sabemos que un usuario consumidor podría acceder a la ruta de edición de perfil, pero solo para editar su perfil particular, sin poder editar el perfil de otro consumidor. Así es que desarrollamos un guard que se encarga de chequear que ciertas rutas sean accedidas únicamente por el usuario actual loggeado. Por ejemplo, si el identificador del usuario loggeado es 1234 (obtenido a partir de su token JWT), el usuario podrá acceder a la ruta **user/edit/1234**, pero no a **user/edit/0088**, la cual solo podría ser accedida por el usuario cuyo identificador es 0088.

En el extracto de código a continuación, mostramos cómo se define una función guard en particular, tomando como ejemplo la función *canActivateAdminLoggedIn* mencionada anteriormente:

```
export const canActivateAdminLoggedIn: CanActivateFn = (
  route: ActivatedRouteSnapshot,
  state: RouterStateSnapshot
) => {
  const authService = inject(AuthService);
  const router = inject(Router);

  const isAdminLoggedIn = authService.isUserLogged() &&
     authService.hasAdminRole();

  if (isAdminLoggedIn) {
    return of(true);
  } else {
  authService.clearLoginInformation();
    router.navigate(['/login']);
    return of(false);
  }
};
```

Como conclusión, la selección del framework Angular resultó esencial para alcanzar una implementación ágil, sencilla y eficiente del cliente de El Paseo. Se aprovecharon la facilidad de codificación inherente a un proyecto Angular, las funcionalidades ofrecidas por Angular CLI, y la experiencia acumulada de uso de esta tecnología durante el transcurso de la carrera universitaria, fusionando estos aspectos para lograr un desarrollo efectivo y sin mayores complicaciones.

### 5.5.4. Angular Material

Angular Material[34] es una biblioteca de componentes de interfaz de usuario desarrollada por el equipo de Angular para facilitar la creación de aplicaciones web con un diseño visual atractivo y coherente. Proporciona una serie de componentes pre construidos y estilizados basados en el diseño de Material Design, una guía de diseño desarrollada por Google. Además de los componentes visuales, Angular Material también ofrece directivas, servicios y estilos que facilitan la construcción de interfaces de usuario consistentes y atractivas.

[34]Documentación oficial: `https://material.angular.io/`

La elección de Angular Material resultó fundamental, ya que nos liberó de la ardua tarea de codificar estilos básicos para los componentes HTML, aprovechando en su lugar los estilos y animaciones predefinidos por esta biblioteca. Esta decisión aceleró el proceso de codificación y aseguró una coherencia visual atractiva, permitiendo enfocarnos en la funcionalidad del cliente y no en cómo visualizar en la pantalla cada componente.

### 5.5.5. Otras librerías

**Angular Google Maps:** La librería agm-core[35] se usa para integrar mapas de Google Maps, al proporcionar componentes y directivas que facilitan la visualización de mapas y la interacción con ellos. En nuestro proyecto, la utilizamos para mostrar en un mapa la ubicación de los distintos puntos de retiro disponibles para retirar las órdenes realizadas por los clientes.

**Ng Bootstrap:** Ng-bootstrap[36] Es una librería de componentes para Angular que implementa Bootstrap[37], un popular framework de diseño y desarrollo front-end. Esta librería se utiliza para integrar los componentes de Bootstrap (su estética y funcionalidades) en aplicaciones Angular de manera más nativa y con una integración más eficiente.

**Ng Charts:** ng2-charts[38] es una herramienta popular utilizada para crear gráficos interactivos y visualizaciones de datos en aplicaciones Angular. Se basa en HTML5 y utiliza el elemento canvas para renderizar gráficos de manera dinámica en el navegador. La utilizamos para mostrar métricas de negocio en la pantalla "Métricas" a la que acceden los administradores.

**JWT Decode:** La librería jwt-decode[39] es utilizada para decodificar los JWT provenientes del servidor y así obtener la información almacenada en el payload de los tokens.

[35] Documentación oficial: `https://www.npmjs.com/package/@agm/core`
[36] Documentación oficial: `https://ng-bootstrap.github.io`
[37] Documentación oficial: `https://getbootstrap.com/docs/5.3/getting-started/introduction/`
[38] Doccumentación oficial: `https://www.npmjs.com/package/ng2-charts`
[39] Documentación oficial: `https://www.npmjs.com/package/jwt-decode`

**RxJS:** La librería rxjs[40] se utiliza para la programación reactiva en JavaScript. Se basa en el patrón de diseño Observable y proporciona una serie de operadores para trabajar con secuencias de eventos asíncronos. La utilización de esta librería nos permitió, entre otras cosas, reaccionar a ciertos eventos que ocurrían en las pantallas, como por ejemplo, recargar los listados cada vez que se añade o se hace una actualización en algún elemento.

**Sweet alert:** La biblioteca sweetalert2[41] proporciona una interfaz de usuario atractiva y personalizable para mostrar alertas y cuadros de diálogo modales en aplicaciones web. A diferencia de los cuadros de alerta predeterminados proporcionados por los navegadores, SweetAlert2 permite diseñar y personalizar las alertas de una manera más elegante y fácil de usar. Entonces, se utiliza comúnmente para mejorar la experiencia del usuario al mostrar mensajes, confirmaciones o solicitudes de entrada de manera más atractiva. En este proyecto, utilizamos las alertas que proporciona esta librería para mantener informado al usuario sobre el resultado de las operaciones que realizó. Por ejemplo, si un administrador quiso modificar un producto y por algún motivo la respuesta del servidor fue un error, se notifica este resultado al usuario a partir de una alerta mostrando el mensaje de error correspondiente.

**Flex layout:** La librería flex-layout[42] es un conjunto de directivas y servicios de Angular que facilita la creación de interfaces de usuario flexibles y responsivas (adaptables) utilizando el modelo de diseño flexible de CSS (Flexbox). Permite a los desarrolladores de Angular definir y gestionar fácilmente diseños complejos y dinámicos en sus aplicaciones.

[40] Documentación oficial: `https://rxjs.dev/`
[41] Documentación oficial: `https://sweetalert2.github.io/`
[42] Documentación oficial: `https://github.com/angular/flex-layout`

# Capítulo 6

# Funcionalidades destacadas

En este apartado se mencionarán las funcionalidades distinguidas del portal web de El Paseo, ya sea por su complejidad técnica o por su gran utilidad en el negocio de la comercializadora.

## 6.1. Restablecer contraseña y primer registro de contraseña para usuarios productores y administradores

Si pensamos en nuestra experiencia como usuarios de aplicaciones web, rápidamente podemos afirmar que el manejo de contraseñas nos permite, entre otras cosas, recuperar la misma en caso de que la hayamos olvidado. La aplicación web de El Paseo no es la excepción: si un usuario con rol administrador selecciona la opción "Restablecer contraseña" de un usuario en particular, el sistema ejecutará una serie de instrucciones para darle la posibilidad de registrar una contraseña nueva y así recuperar el acceso a su cuenta en caso de que lo haya perdido.

Entonces, si un administrador solicita el restablecimiento de la contraseña de un usuario, el frontend enviará una petición al backend con la información del administrador encriptada en un token JWT junto con el email del usuario en cuestión. El backend, por su parte, realizará las validaciones correspondientes para garantizar que solo los administradores o el propio usuario puedan realizar esta acción, y luego enviará un email a la dirección de correo electrónico recibida en la petición. En este email, se le informa que puede

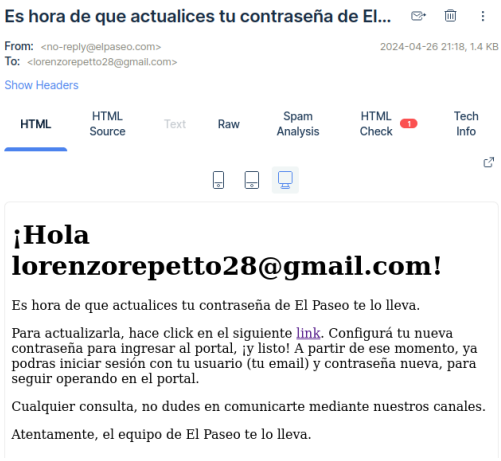


Figura 6.1: Email template para solicitar un registro de nueva contraseña

registrar una nueva contraseña siguiendo un link en particular, el cual envía como parámetro la información del usuario al que se le está restableciendo la contraseña en un token, para garantizar que solo este usuario pueda hacer el cambio. A continuación, en la figura 6.1, dejamos a modo de ejemplo un email enviado al correo electrónico *lorenzorepetto28@gmail.com*, solicitándole configurar su nueva contraseña.

Cuando el usuario sigue el link, se le solicita que ingrese una contraseña nueva y la confirme, como se muestra a continuación en la figura 6.2. Si los datos son correctos, el sistema registrará la nueva contraseña para el usuario. De esta forma, un usuario que perdió acceso a su cuenta podría recuperar la misma rápidamente y de forma segura. Además, esta funcionalidad se puede reutilizar para los casos en donde un usuario no perdió el acceso a su cuenta, sino que simplemente quiere hacer un cambio de la contraseña por seguridad. En este caso, el usuario podría solicitar el cambio a través de su perfil, y la información que se envía en el token de autorización es la del mismo usuario.

Por otro lado, sabemos que solo usuarios administradores pueden dar de alta usuarios productores u otros administradores. Para garantizar que solo los propios usuarios conozcan su contraseña, se reutilizó el mismo flujo mencionado anteriormente. Cuando un administrador quiere registrar otro administrador o un productor, solo debe ingresar el email del nuevo usuario. El sistema se encargará de enviar un correo electrónico al email ingresado, en el cual se le pedirá al nuevo usuario que siga un link en particular para

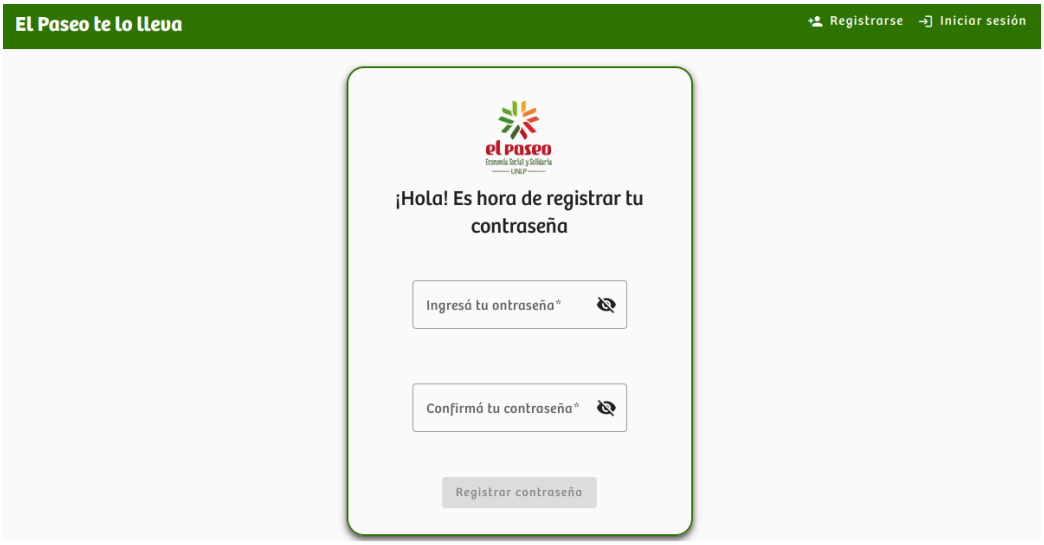


Figura 6.2: Pantalla de registro de contraseña para un nuevo usuario administrador o productor

registrar su primera contraseña y así poder ingresar al sistema. Este link contiene la información del nuevo usuario, y permite garantizar que solo el propio usuario tenga acceso a esta funcionalidad. De esta forma, el usuario dado de alta registrará una contraseña y el sistema le permitirá a partir de ese momento el inicio de sesión en el portal web.

## 6.2. Aplicación web adaptable a distintos tamaños de pantalla

El diseño *responsive* de aplicaciones web surgió como respuesta a la necesidad de adaptar los sitios web a la creciente diversidad de dispositivos y tamaños de pantalla utilizados para acceder a Internet.

Si un desarrollador desarrolla su sistema web sin tener en cuenta esta característica, cuando los usuarios eventualmente accedan al mismo desde sus celulares (como hoy en día estamos acostumbrados), todas las pantallas se deformarán complicando la navegación del usuario, los estilos definidos no se aplicarán correctamente, y habrá elementos o secciones que directamente no podrán ser accedidos.

Por esta razón, es de suma importancia que los desarrolladores utilicen técnicas de diseño para crear sitios web que puedan adaptarse de manera dinámica y elegante a diferentes tamaños de pantalla y dispositivos. Los beneficios de tener un único sitio web que se adapte automáticamente a cualquier dispositivo están a la vista, por lo que rápidamente se aceleró la adopción del sistema responsive, alcanzando a convertirse hoy en una práctica estándar de desarrollo web esencial para garantizar una experiencia de usuario óptima en la web moderna.

En el desarrollo del portal de El Paseo tuvimos en cuenta esta práctica, ya que entendemos que seguramente los usuarios consumidores accederán al portal a través de sus celulares, al menos la primera vez.

Sin embargo, consideramos que el diseño de una aplicación web responsive no suplanta el desarrollo de una aplicación móvil: si la mayor cantidad de usuarios utilizan sus teléfonos móviles para acceder y operar en el portal, o si se desea agregar nuevas funcionalidades que dependan de ciertas características o aspectos del celular (como cámara, acceso a GPS, etc), entonces debería considerarse desarrollar la aplicación móvil correspondiente para garantizar una experiencia de usuario adaptada y consistente.

Para garantizar la adaptabilidad del portal a los distintos tamaños de pantalla, utilizamos la librería flex-layout[1] que nos proporciona herramientas para tomar distintas decisiones y reacomodar los elementos de nuestra aplicación según el tamaño de pantalla del dispositivo. A modo de ejemplo, dejamos la figura 6.3 en donde se puede apreciar la pantalla de inicio de sesión adaptada al tamaño de pantalla de un móvil Galaxy S20.

## 6.3. Encriptación de contraseñas de usuarios

La encriptación de contraseñas protege la información personal y sensible de los usuarios. Si no se utilizara, al almacenar contraseñas en texto plano, cualquier persona que acceda a la base de datos podría ver las contraseñas y potencialmente usarlas para acceder a las cuentas, comprometiendo no solo la seguridad de los usuarios, sino también la reputación y confiabilidad de la empresa afectada. La encriptación garantiza que incluso si un atacante accede a la base de datos, las contraseñas estén protegidas y no puedan ser fácilmente descifradas.

[1] Documentación oficial: `https://github.com/angular/flex-layout`

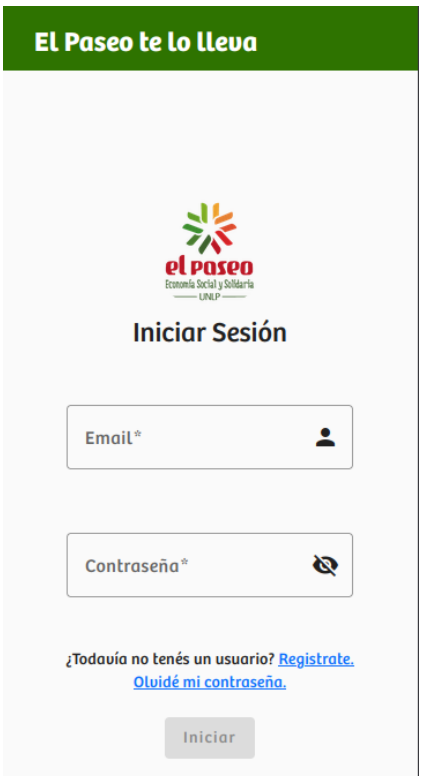


Figura 6.3: Diseño responsive de la pantalla de Inicio se Sesión

Es por esto que utilizar encriptación de contraseñas demuestra un compromiso con la seguridad y la privacidad de los usuarios. En este proyecto, utilizamos la función de hashing *bcrypt* para encriptar las contraseñas de todos los usuarios del sistema.

Bcrypt es una función de hashing diseñada para almacenar de forma segura contraseñas en aplicaciones y sistemas. Es una función de derivación de clave adaptativa que utiliza un algoritmo de cifrado basado en Blowfish. Es conocida por su seguridad y resistencia a ataques de fuerza bruta[2] y de diccionario[3]. Algunas de sus características principales son:

**Salting:** Automáticamente genera un valor aleatorio conocido como *salt* para cada contraseña antes de aplicar el algoritmo de hashing. El salt es almacenado junto con el hash de la contraseña en la base de datos, lo que ayuda a prevenir ataques como el de diccionario, ya que incluso contraseñas idénticas tendrán hashes diferentes debido a la presencia de diferentes valores de salt.

**Ronda de iteraciones:** Permite especificar el número de iteraciones del algoritmo de hashing, teniendo en cuenta que a mayor número de iteraciones, más costoso es el proceso de hashing y más difícil es para un atacante realizar un ataque de fuerza bruta. Este parámetro puede ajustarse para adaptarse a los requisitos de seguridad de la aplicación y al hardware disponible.

**Adaptabilidad:** La función bcrypt es "adaptativa", lo que significa que puede adaptarse a los avances en la tecnología de hardware y en las técnicas de ataque. A medida que la tecnología avanza y los ataques se vuelven más sofisticados, bcrypt puede ajustarse fácilmente, aumentando la ronda de iteraciones para mantener un nivel adecuado de seguridad.

En nuestro sistema, cuando un usuario crea o actualiza su contraseña, esta pasa a través de la función bcrypt para generar un hash seguro que

[2]Los ataques de fuerza bruta utilizan el método de prueba y error para adivinar información de inicio de sesión y claves de cifrado. Los atacantes prueban todas las combinaciones posibles hasta encontrar aquella que permite el acceso.

[3]Delito cibernético que se utiliza para acceder a las cuentas privadas de los usuarios, intentando adivinar la contraseña de la cuenta a través de una lista de palabras, frases y combinaciones numéricas de uso frecuente, o dicho de otra forma, utilizando todas las palabras del diccionario.

luego se almacena en la base de datos, junto con el salt autogenerado. Luego, cuando el usuario intenta iniciar sesión, la contraseña ingresada se compara con el hash almacenado en la base de datos utilizando bcrypt para verificar si coinciden y permitir así el ingreso al sistema.

## 6.4. Generación de reportes de negocio

Durante la fase inicial del proyecto, específicamente en la etapa de Análisis de Requerimientos, se identificó y destacó la necesidad fundamental de la comercializadora de contar con la capacidad de generar informes de negocio a partir de una variedad de fuentes de datos, incluyendo datos de ventas, comportamiento de los consumidores, información de los productores, entre otros.

Esta funcionalidad se identificó como crucial para El Paseo, ya que le brindaría la oportunidad de profundizar en el análisis de la información de negocio y, lo que es más importante, le proporcionaría herramientas fiables para llevar a cabo diversas tareas estratégicas, tales como análisis de riesgos o toma de decisiones basadas en datos.

Al contar con informes detallados y análisis de datos sólidos, la comercializadora estaría mejor equipada para entender las tendencias del mercado, identificar oportunidades de crecimiento, optimizar sus operaciones y, por ende, mejorar su competitividad en el sector. Este enfoque orientado a los datos no solo impulsaría la eficiencia y la eficacia de las operaciones comerciales, sino que también sentaría las bases para un crecimiento sostenible a largo plazo. Cabe destacar que este tipo de tareas se vuelven muy complejas cuando no se cuenta con una digitalización, ya que en el manejo manual de datos se pierde mucha información y, por lo tanto, capacidad de análisis.

Así fue que surgió la idea de agregar una sección de "Reportes" al panel administrativo. En esta sección, los administradores podrán generar una serie de reportes dinámicos, aplicando filtros útiles por productor, ciudad, ronda, etc. Hay que tener en cuenta que los filtros son opcionales, por lo que si no se especifica ninguno, el reporte incluiría toda la información disponible.

A continuación, analizaremos con más detalles cada reporte que se puede generar en el sistema de El Paseo:

### 6.4.1. Mercadería por ronda

Los administradores necesitan una forma fácil de reunir la información de todas las órdenes que deben completar en una semana en particular. Tomemos como ejemplo una semana cualquiera en El Paseo: al comienzo de la semana se van recibiendo los pedidos de los consumidores; a mediados de la misma se comunica a los productores la mercadería que deben entregar para completarlos, y al final de la semana se entregan los pedidos (tanto en modo delivery como en un punto de retiro en particular).

Entonces, al final de la semana es de crucial importancia que el administrador pueda reunir rápidamente todos los pedidos que deben prepararse para entregar en X punto de retiro, o en Y ciudad. Este reporte se creó para resolver esta cuestión: el administrador podrá seleccionar para qué ronda generar el reporte, pudiendo además aplicar diferentes filtros, como por ejemplo a qué productor, punto de retiro, y/o ciudad corresponde cada producto vendido.

En el portal administrativo, se provee la siguiente información para explicar la funcionalidad del reporte: *Este reporte recolecta todos los productos y combos que deben entregarse para una ronda en particular, y genera un documento Excel con los resultados. Si no se aplica ningún filtro, el reporte generado no discrimina puntos de retiro, productores, estados de orden o modos de entrega. En caso de que se quiera generar reportes más específicos, se podrá aplicar los filtros disponibles. Tener en cuenta que los filtros son acumulativos, es decir, que si se selecciona por ejemplo productor "X" punto de retiro "Y", entonces el reporte incluirá todos los productos que deben entregarse en el punto de retiro "Y" en la ronda en particular que corresponden al productor "X".*

Entonces, el resultado del reporte es un listado de productos y combos que deben entregarse en esa ronda, respetando los filtros seleccionados. De esta forma, el administrador podrá generar, por ejemplo:

- Un reporte por cada productor: Rápidamente obtendría la información de qué productos solicitarle a un productor en particular para entregarlos al final de la semana
- Un reporte por cada ciudad: Para obtener la información de cuáles y cuántos productos y/o combos deben entregarse en cada ciudad. Este reporte sería de mucha utilidad para los repartidores de pedidos, ya que

se podría utilizar para distribuir los pedidos a realizar según la ciudad en la que deben entregarse

- Un reporte por cada punto de retiro: Para poder agrupar todos los productos y combos que deben estar presentes en un punto de retiro, en particular para que puedan ser retirados por los consumidores que realizaron las compras

- Un reporte por cada estado de orden ("ACEPTADO", "CANCELADO", "ENTREGADO"): Para obtener los productos y combos involucrados en órdenes con un estado en particular

### 6.4.2. Mercadería por ronda y consumidores

Habíamos mencionado anteriormente que por la falta de digitalización, a la comercializadora se le complicaba atender los reclamos o consultas realizadas por los consumidores, ya que se perdía mucho tiempo y esfuerzo intentando reunir toda la información necesaria para entender el problema y poder resolverlo. Este reporte viene a solucionar esta cuestión: el administrador podrá rápidamente acceder a los pedidos realizados por un cliente en una ronda en particular, obteniendo toda la información de los productos y combos involucrados en sus compras. De esta forma, será más fácil responder a las dudas o reclamos de los consumidores.

En el portal administrativo, se provee la siguiente información para explicar la funcionalidad del reporte: *Éste reporte recolecta todos los productos y combos que están asociados a una orden en estado "ACEPTADO" para una ronda y cliente en particular, y genera un documento Excel con los resultados. En caso de que no se especifique un cliente, el reporte se genera teniendo en cuenta todos los clientes involucrados.*

Si bien parece un reporte similar a "Productos por ronda" o "Combos por ronda" mencionados anteriormente, la diferencia de éste es que la agrupación de los datos se hace por consumidor y no por producto como lo hacen los anteriores.

### 6.4.3. Órdenes por ronda y estado

El administrador necesita una forma fácil de obtener todas las órdenes que se deben completar (o ya se completaron) en una ronda en particular, según

sus estados. De esta forma, se podría rápidamente obtener, por ejemplo, todas las órdenes en estado "CANCELADO" y analizar los motivos de cancelación para aprender de los errores y evitar que se vuelvan a repetir.

En el portal administrativo, se provee la siguiente información para explicar la funcionalidad del reporte: *Éste reporte recolecta todos los pedidos para una ronda, estado de pedido, punto de retiro y/o ciudad de entrega en particular, y genera un documento Excel con los resultados. En caso de que no se especifique ningún filtro, el reporte se genera teniendo en cuenta todos los disponibles.*

Entonces, el resultado del reporte es un listado de órdenes asociadas a una ronda en particular, respetando los filtros seleccionados. De esta forma, el administrador podrá generar, por ejemplo:

- Un reporte para el estado de orden "CANCELADO": Para acceder a las órdenes que fueron canceladas tanto por los consumidores como por algún usuario administrador, junto con su razón de cancelación. Así, se podrá empezar a analizar cuáles son los puntos débiles de la comercializadora para trabajar sobre ellos y evitar órdenes canceladas.

- Un reporte para el estado de orden "PAGADO": Para obtener rápidamente las órdenes que ya están listas para entregarse en una semana en particular, y así poder tomar decisiones al respecto y organizarse

- Un reporte para el estado de orden "PENDIENTE DE PAGO": Para obtener rápidamente las órdenes que todavía no fueron abonadas. El administrador podría fácilmente comunicarse con los consumidores asociados a estas órdenes para gestionar el pago de las mismas, y así poder avanzar con los envíos

- Un reporte para el estado de orden "ENTREGADO": Para obtener lás órdenes que fueron entregadas exitosamente en una ronda (tanto en un punto de retiro como en una ciudad en particular), junto con la información de la misma

### 6.4.4. Mercadería sin stock

Es de vital importancia que los administradores puedan obtener rápidamente cuáles son los productos o combos que se quedaron sin stock, para así

poder avisarles a los productores correspondientes que se necesitan más unidades. Si un producto o combo se queda sin stock en el sistema, no se podrán realizar ventas de ese producto, por lo que si no se actualiza rápidamente el stock de los mismos se podrán perder oportunidades de venta.

Para resolver este inconveniente, creamos el reporte "Mercadería sin stock", el cual devuelve un listado de los productos y combos que actualmente se encuentran sin stock. De esta forma, el administrador podría organizarse para generar el reporte cada X cantidad de tiempo, y ocuparse de actualizar el stock de la mercadería que se quiere seguir vendiendo.

### 6.4.5. Recaudación total por familia productora

Antes de su digitalización, los trabajadores administrativos de El Paseo debían encargarse de generar reportes de forma manual sobre el dinero recaudado en una semana en particular por cada familia productora. Esta tarea era muy importante para determinar el monto que se le debía entregar a cada productor, y además calcular el porcentaje que le correspondía a la comercializadora. Por esta razón es que es esencial que este proceso se realice sin errores para garantizar que las partes involucradas no se vean perjudicadas.

Una vez que se desarrolló el portal, nos encargamos de que todas las tareas que antes eran manuales ahora estén automatizadas, y este reporte no fue la excepción.

A partir del reporte "Recaudación por familia productora" el administrador puede rápidamente generar un documento Excel en donde se detalla cada producto vendido por una familia productora en particular, junto con su cantidad y el cálculo del total recaudado.

En el portal administrativo, se provee la siguiente información para explicar la funcionalidad del reporte: *Este reporte recolecta todos los productos vendidos para una familia productora en una ronda en particular, y genera un documento Excel con los resultados. Si no se aplica ningún filtro, el reporte generado no discrimina familias productoras. El reporte generado contiene información de los productos vendidos junto con su cantidad y el total de dinero acumulado.*

### 6.4.6. Recaudación total por combo

Para complementar el reporte mencionado anteriormente, desarrollamos el reporte "Recaudación por combo" el cual recolecta el dinero recaudado

a partir de la venta de combos. Como ya sabemos, los combos involucran distintos productos de distintos productores, por lo que a partir de la generación de este reporte la comercializadora podría decidir qué porcentaje de recaudación le corresponde a cada productor involucrado.

En el portal administrativo, se provee la siguiente información para explicar la funcionalidad del reporte: *Este reporte recolecta todos los combos vendidos para una ronda en particular, y genera un documento Excel con los resultados. Si no se aplica ningún filtro, el reporte generado no discrimina combos. El reporte generado contiene información de los combos vendidos junto con su cantidad y el total de dinero acumulado.*

## 6.5. Panel de métricas

La sección llamada "Métricas" en el portal administrativo le permite a los administradores obtener rápidamente una visión general del funcionamiento de la comercializadora, pudiendo filtrar por distintas fechas. Se pueden visualizar distintas métricas de negocio y estadísticas, las cuales se transforman en herramientas útiles y confiables para analizar información y poder tomar decisiones basadas en datos o analizar riesgos. En la figura 6.4 se puede apreciar a modo de ejemplo cómo se muestran estas métricas.

Cabe aclarar que este panel de métricas es dinámico, ya que se puede cambiar la fecha para buscar las estadísticas de un período en particular. Por defecto, el panel de métricas muestra los datos obtenidos a partir de haber filtrado por el mes actual del año corriente.

Entre las métricas que se pueden visualizar se encuentran las siguientes:

**Ventas por ciudad:** Gráfico de barras en dónde se contabilizan la cantidad de ventas diferenciadas por cada ciudad en donde El Paseo tiene cobertura para realizar entregas de pedidos. Esta métrica podría servir para entender en qué ciudad la comercializadora tiene más ventas, y también permite trabajar en las ciudades con menos ventas para impulsar la participación de consumidores

**Ventas por punto de retiro:** Gráfico de barras en dónde se contabilizan la cantidad de ventas diferenciadas por punto de retiro

**Consumidores por ciudad:** Gráfico de barras en dónde se contabilizan la cantidad de consumidores registrados en el sistema diferenciados por

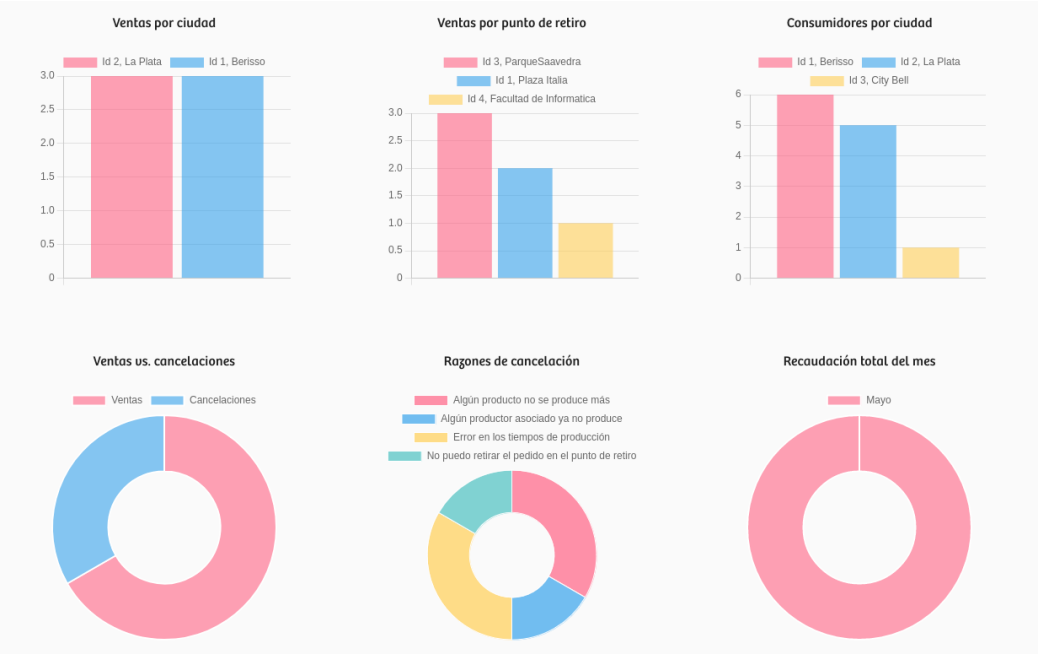


Figura 6.4: Panel de métricas administrativo

su ubicación geográfica. Esta información se obtiene a partir de las direcciones que fueron registradas por cada consumidor, y es de mucha utilidad a la hora de entender en dónde se encuentran ubicados los consumidores

**Ventas vs. cancelaciones:** Gráfico que compara cantidad de ventas exitosas con las ventas canceladas. Esta métrica permitiría a la comercializadora darse cuenta rápidamente si sus ventas se están cancelando, por lo que podría accionar justo a tiempo para revertirlo

**Razones de cancelación:** Gráfico que muestra la cantidad de ventas canceladas agrupadas por las razones de cancelación. Esta métrica permite analizar cuáles son las razones de cancelación más comunes, permitiendo a la comercializadora trabajar en sus puntos débiles para evitar que se sigan cancelando órdenes

**Recaudación total del mes:** Gráfico que muestra el total del dinero recaudado por la comercializadora en un mes en particular

**Familias Productoras con más ventas:** Gráfico que muestra las familias productoras con más ventas. Cabe destacar que el número de resultados que se muestra en el gráfico se puede modificar desde el panel administrativo de "Configuraciones". Por defecto, se muestran las tres familias productoras con más ventas

**Familias Productoras con menos ventas:** Gráfico que muestra las familias productoras con menos ventas. El número de resultados que se muestra en el gráfico se puede modificar desde el panel administrativo de "Configuraciones"

**Consumidores con más compras:** Gráfico que muestra los consumidores con más compras. El número de resultados que se muestra en el gráfico se puede modificar desde el panel administrativo de "Configuraciones"

**Consumidores con menos compras:** Gráfico que muestra los consumidores con menos compras. El número de resultados que se muestra en el gráfico se puede modificar desde el panel administrativo de "Configuraciones"

**Productos menos vendidos:** Gráfico que muestra los productos más vendidos. Esta métrica permitiría analizar el mercado para entender qué es lo que más compran los consumidores en determinada época del año. El número de resultados que se muestra en el gráfico se puede modificar desde el panel administrativo de "Configuraciones"

**Productos menos vendidos:** Gráfico que muestra los productos menos vendidos. El número de resultados que se muestra en el gráfico se puede modificar desde el panel administrativo de "Configuraciones"

## 6.6. Componente administrativo de configuración

A lo largo de nuestra carrera universitaria y nuestras experiencias como desarrolladores, hemos comprendido la importancia de crear productos de software que empoderen a los clientes y usuarios, permitiéndoles operar de manera autónoma y sin depender constantemente del equipo de desarrollo.

Teniendo esto en cuenta, desde el inicio de este proyecto nos hemos dedicado a garantizar que nuestro portal web sea altamente configurable por los administradores, asegurando así su dinamismo y evitando la presencia de información estática que pueda quedar obsoleta con el tiempo.

Para lograr este objetivo, hemos desarrollado un panel de configuración para el perfil administrador, para que tenga un control total sobre la información presente en el portal, permitiendo la modificación de cualquier aspecto que pueda cambiar a lo largo del tiempo. ¿Qué significa esto en la práctica? Simplemente que el administrador tiene la capacidad de ajustar y actualizar la información según las necesidades del momento, brindándole una total flexibilidad y dinamismo en la gestión del portal. Además, esto permite que los consumidores siempre hagan uso de información actualizada, lo cual es de suma importancia para el buen funcionamiento de la comercializadora y para garantizar una buena experiencia de los consumidores.

Tengamos en cuenta las siguientes ejemplificaciones para entender la funcionalidad y las ventajas que trae aparejadas:

Ningún texto informativo está *hardcodeado*[4], sino que cada uno es fácilmente modificable a partir de una configuración que puede controlar el administrador siempre que lo requiera. Tomemos como ejemplo los textos que se ven en la página principal, en las secciones "Conocenos" o "Contactanos". En estas secciones la comercializadora tiene la posibilidad de contar su historia y cuales son los medios disponibles para permitir que los consumidores se contacten. Es evidente que esta información puede variar a lo largo del tiempo, porque, por ejemplo, la comercializadora quiere contar su historia de una forma distinta, un teléfono se dejó de usar y ya no se quiere mostrar en la sección de contacto, o se sumó una nueva red social que se quiere difundir. En todos estos casos, si no existiera esta funcionalidad, los administradores deberían contactarse con los desarrolladores del portal para que ellos se encarguen de cambiar los textos directamente en el código para actualizar la información que se quiere mostrar. Obviamente, esto trae aparejado una espera por parte del cliente hasta que este nuevo desarrollo esté productivo. Para evitar esto fue que desarrollamos esta funcionalidad: los administradores podrían cambiar estos textos en todo momento, y las actualizaciones serían instantáneas. Esto trae aparejado muchas ventajas, entre ellas: poder

[4]Término que se utiliza en programación para describir una práctica en la que se insertan valores directamente en el código fuente en lugar de usar variables o configuraciones externas. Esto hace que no pueda ser modificado fácilmente sin editar el código en sí mismo.

actualizar información al instante y evitar desinformación en los consumidores, adaptar el portal a un período del año en particular (por ejemplo, se podría subir una imagen deseando feliz Navidad a fin de año), entre otras cosas.

Por otro lado, las fotos que se muestran en el carrusel de la página principal también pueden ser modificadas en el panel de configuraciones. La comercializadora puede tener la necesidad de añadir una foto nueva o eliminar una que ya quedó desactualizada, y no debería necesitar de los desarrolladores para hacer esto.

Creemos que esta funcionalidad es fundamental para la comercializadora, ya que le permite adaptarse a distintas situaciones y trabajar de forma independiente y flexible, con la posibilidad de que el portal se encuentre siempre brindando información actualizada.

## 6.7. Blog de noticias

Luego de haber estudiado la historia de El Paseo, su rol en la sociedad, y sus principios y valores, entendimos que el corazón de la comercializadora es su relación con los consumidores. Los consumidores eligen a la comercializadora no solo por sus productos, sino también por los lazos sociales que se fueron estableciendo y fortaleciendo a lo largo del tiempo.

El Paseo trabaja desde sus inicios consolidando lazos entre consumidores y familias productoras a partir de la divulgación de información de utilidad, prácticas saludables y conscientes con el medio ambiente, historias de las familias productoras, talleres y capacitaciones para los trabajadores de El Paseo, eventos sociales, entre otros.

Es por esto que cuando desarrollamos el portal, nos pareció de suma importancia que haya una sección dedicada a seguir nutriendo las relaciones no sólo con los consumidores sino también con la comunidad local. Así es que desarrollamos un Blog de noticias, en donde la comercializadora podrá publicar artículos de interés, como por ejemplo: historias de familias productoras, recetas saludables, herramientas para cuidar el medio ambiente, principios de la ESS, eventos en los que participa la comercializadora, etc. Estos artículos se verán en un listado paginado a los que pueden acceder cualquier usuario que ingrese al portal de El Paseo, sin necesidad de estar registrado o con una sesión iniciada.

Cada noticia estará comprendida por un título, subtítulo, fecha de crea-


Figura 6.5: Blog de noticias

ción, texto de la noticia, video asociado y publicación relacionada. Estos dos últimos datos son opcionales. A continuación, en la figura 6.5 dejamos una noticia a modo de ejemplo.

## 6.8. Mapa dinámico de puntos de retiro

Como mencionamos a lo largo de este informe, los usuarios administradores podrán crear, modificar o eliminar puntos de retiro, para representar los lugares que la comercializadora tiene disponibles para hacer retiro de mercaderías compradas por los consumidores.

Si pensamos en una dinámica de trabajo real, rápidamente nos damos cuenta de que los puntos de retiro no son estáticos, es decir, pueden cambiar a lo largo del tiempo. Quizás un día la comercializadora tiene un punto de retiro funcionando, pero en un futuro este se puede mudar cambiando de dirección, o pueden aparecer puntos nuevos. Es por esto que como desarrolladores nos pareció de suma importancia que sea el administrador el responsable de modificar los puntos de retiro disponibles, pudiendo crear nuevos o actualizar los existentes para siempre mantener la información actualizada y consistente.

Es por esto que también se debe considerar cómo se mantiene informado al consumidor de estas actualizaciones, ya que son los que requieren de esta información para poder retirar sus compras correctamente. A partir de esta


Figura 6.6: Mapa de puntos de retiro de la página principal de El Paseo

necesidad, nos surgió la idea de mostrar en la página principal de El Paseo el mapa con los puntos de retiro identificados, para que los consumidores no pierdan el acceso a la información de los mismos, y de esta forma rápidamente puedan identificar cuál es el punto de retiro más conveniente para retirar sus compras, o si hay alguno nuevo que puedan considerar.

Para visualizar el mapa, utilizamos la librería agm-core[5], la cual nos otorga herramientas para marcar y localizar puntos específicos indicando su latitud y longitud, como se muestra en la figura 6.6. Esta información es ingresada por los administradores cuando registran el nuevo punto de retiro.

[5]Documentación oficial: `https://www.npmjs.com/package/@agm/core`

# Capítulo 7

# Evaluación

Para evaluar la usabilidad del nuevo sistema de El Paseo, se hicieron pruebas con 4 usuarios administradores y con 7 usuarios consumidores habituales y nuevos. Con respecto a las pruebas de los administradores, se buscó determinar si el software facilitaba sus tareas diarias que debían llevar a cabo para administrar los pedidos y la información de los productos y familias productoras que necesitaban difundir. Por otro lado, el objetivo de las pruebas de los consumidores era determinar si el uso del portal web les era útil a la hora de gestionar la compra de productos.

Se organizaron los distintos casos de prueba con sus criterios de aceptación, y se definieron formularios de Google para compartirles a los usuarios, para que ellos mismos envíen sus observaciones una vez realizadas las distintas pruebas. Estos formularios también servían como guía para la prueba, ya que brindaban una explicación de como ejecutar un caso de prueba en particular siempre que sea necesario.

Además, se preparó un ambiente dedicado para las pruebas. Para esto, se publicó una imagen de Docker en Docker Hub la cual contenía todos los componentes necesarios para levantar el ambiente de El Paseo. Esta imagen fue enviada al área de soporte técnico del LINTI (Laboratorio de Investigación en Nuevas Tecnologías Informáticas) para ser instalada en un servidor de pruebas que el LINTI dispone para este tipo de trabajos (tesinas y trabajos de cátedras). Así fue que el portal web quedó publicado para hacer las pruebas en el siguiente dominio: https://elpaseo.linti.unlp.edu.ar/frontend/

En primer lugar, se realizaron pruebas con los administradores del sistema. Cada administrador utilizó su computadora personal para acceder al portal web y ejecutar los distintos casos de prueba. Estas pruebas se llevaron

a cabo durante una reunión en Google Meet, junto con la tesista. Durante esta sesión de prueba, la tesista no intervino en la ejecución de las pruebas con la intención que los administradores las realizaran con total autonomía, y que solo recurrieran a tesista en caso de no saber cómo proceder.

Al terminar con los casos de prueba, y luego de analizar los resultados obtenidos, se pudo concluir que el sistema les resolvía todas las tareas que los administradores hacían hasta el momento de forma manual, de una forma rápida, clara e intuitiva. La respuesta de los administradores fue muy positiva, quienes pudieron completar las pruebas autónomamente sin requerir la ayuda de la tesista .

Se registraron sugerencias para mejorar el software, pero ninguna de ellas fue especialmente significativa ni implicó cambios importantes. Entre ellas, podemos mencionar a modo de ejemplo disminuir la información requerida para registrarse en el sistema para que el registro sea más sencillo, eliminar algunas restricciones que se imponían a la hora de registrar una contraseña para simplificar este proceso, o cambiar algunos colores de la interfaz gráfica para lograr una mayor coherencia con los colores que identifican a El Paseo. Estas recomendaciones se tradujeron en la incorporación de algunas tareas menores a la planificación, las cuales se resolvieron sin mayor esfuerzo.

Luego, una vez confirmada la utilidad del sistema por parte de los administradores, comenzaron las pruebas con consumidores. La idea fue probar el sistema en un entorno real, es decir con datos reales, por lo que se le solicitó al personal de El Paseo que cargaran información en el sistema, y eligieran consumidores para la sesión de prueba del sistema. El criterio de selección de los consumidores fue el siguiente: consumidores habituales de la comercializadora que conozcan la calidad de los productos y a las familias productores y también consumidores nuevos, es decir que nunca habían comprado en la comercializadora. Ambos tipos de consumidores debieron primero registrarse en el portal web, y luego realizar sus pedidos.

Una vez que usaron el sistema para hacer sus compras y enviaron sus observaciones a través del formulario, nos dedicamos a analizar las respuestas y a reorganizar el trabajo de acuerdo a los resultados, los cuales se pueden observar en el documento Pruebas del sistema de El Paseo te lo lleva - Respuestas de formulario. A continuación, se hará un análisis de los resultados obtenidos.

Todos los consumidores indicaron que pudieron registrarse exitosamente en el sistema, sin mayores inconvenientes a la hora de completar los datos requeridos. Dentro de las mejoras que propusieron para esta funcionalidad,

indicaron que para simplificar un poco más el registro, se podría considerar el pedido de una contraseña más simple. Actualmente y por temas de seguridad, el sistema solicita una contraseña que contenga al menos una minúscula, una mayúscula, un número y un caracter especial. Por otro lado, dado que a la hora de registrar una dirección el sistema te permite seleccionar el barrio al que pertenece, mencionaron que se podría evitar solicitar luego el código postal, ya que es un dato redundante.

A la hora de hacer el pedido, todos los consumidores indicaron que encontraron sin dificultad los productos que deseaban comprar. Hubo un consumidor en particular que destacó el menú lateral izquierdo de la interfaz gráfica, el cual permite hacer una búsqueda por nombre de producto, y además muestra una lista para poder filtrar el listado de productos de acuerdo a su categoría o familia productora asociada. El consumidor mencionó que este componente le fue muy útil a la hora de buscar el producto que quería comprar.

Por otro lado, hubo consumidores que necesitaron modificar el pedido. Estos consumidores indicaron que lograron hacerlo sin inconvenientes, agregando o quitando productos, o modificando el estado del pedido para cancelarlo en caso de que sea necesario.

Al responder si encontraron información útil sobre las familias productoras, solo un consumidor indicó que no encontró información detallada de los productores, como por ejemplo, contacto de Instagram, Facebook o sitio web oficial. Esto se debió a que durante la carga de los datos necesarios para hacer las pruebas, no se armó un texto real para describir a las familias productoras. El sistema simplemente mostraba el texto de prueba en la descripción de los productores, y esto hizo que se obtenga esta respuesta negativa.

En un futuro, cuando el sistema esté publicado en producción, es importante que todos los textos que se carguen sean reales y completos, para poder aprovechar al máximo las funcionalidades que provee el portal web. Igualmente, cabe destacar que el sistema le permite a los administradores modificar los textos en todo momento fácilmente, por lo que siempre que se detecte que hay alguna información incompleta o errónea, ellos mismos podrán hacer los cambios necesarios.

Por otro lado, dos consumidores indicaron que no lograron acceder a información actualizada sobre el estado del pedido, el contenido del mismo, o la forma de retiro o envío. Analizando estos casos, vimos que la razón fue que al momento de intentar crear sus pedidos, no había en el sistema una ronda creada, lo que imposibilitaba al consumidor avanzar en la compra.

El concepto de ronda es una forma que tienen los administradores del

sistema de manejar y controlar la toma de pedidos. A partir de las mismas, ellos pueden indicar en qué fecha pueden recibir pedidos nuevos, y en qué fecha no, dado que necesitan esos días para organizar los pedidos registrados. Es por esto que es importante que el personal administrativo esté al tanto de cómo configurar las fechas de las rondas. Gracias a esta prueba, podemos concluir que es necesario hacer un entrenamiento formal para el personal administrativo sobre cómo utilizar el sistema, para evitar estos errores.

El resto de los consumidores que hicieron sus pedidos cuando había una ronda habilitada, indicaron que el proceso fue muy sencillo e intuitivo. Resaltaron que les gustó mucho la interfaz gráfica de la aplicación, y los mensajes que devuelve el sistema cada vez que se realiza una acción, lo que les permitió entender en todo momento los resultados que tenían sus interacciones con el sistema. Además, estos mensajes también les facilitaron entender el estado de su pedido en todo momento.

Por último, un consumidor que accedió al sistema desde su celular, indicó que algunas veces las imágenes no se veían de un tamaño correcto. Si bien a la hora de desarrollar el sistema se hizo hincapié en que sea responsive, gracias a estas pruebas se puede concluir que todavía hay componentes que pueden mejorar su adaptación de tamaño a las distintas pantallas. Para mejorar este aspecto, se sumaron tareas a la planificación para revisar el comportamiento de cada componente, y readaptar su tamaño en caso de que sea necesario.

En general, los consumidores destacaron que el proceso de registración y compra fue muy claro e intuitivo, lo que les permitió registrarse y hacer un pedido en muy poco tiempo sin inconvenientes.

Para concluir, las pruebas realizadas en el sistema arrojaron resultados en general positivos. Aunque se sugirieron algunas mejoras, estas no representan modificaciones significativas que indiquen fallas en alguna de las etapas del desarrollo de software, como una mala Ingeniería de Requerimientos o una codificación deficiente que resultara en un producto que no cumpla con las expectativas o las necesidades de los clientes y usuarios.

Las mejoras mencionadas se refieren principalmente a la interfaz gráfica, los datos solicitados para registrarse en el sistema, y la claridad de la información mostrada, lo que no está relacionado directamente con el desarrollo del sistema. Además, dado que el sistema es altamente configurable, los administradores podrán actualizar esta información de manera rápida siempre que consideren que no está completa o que algo no queda claro para los consumidores.

Entonces, en este proyecto, el software demostró cumplir con las necesi-

dades del cliente y resolver el problema de manera efectiva, permitiendo la comercialización de productos en línea de forma clara y rápida, tanto para consumidores habituales como para consumidores que no conocían a la comercializadora.

# Capítulo 8

# Conclusiones y trabajo futuro

## 8.1. Conclusiones

Desde el punto de vista personal, fue todo un desafío hacerme responsable de principio a fin de un proceso de software real y completo, con la complejidad de un problema auténtico.

Antes de comenzar con el proyecto, si bien como estudiante de la universidad sentía que había ganado un montón de herramientas a lo largo de la carrera universitaria, era muy desafiante pensar en desarrollar desde cero un sistema completo como el portal web de El Paseo, y además alcanzar las expectativas de la comercializadora.

Si bien en lo personal ya contaba con experiencias trabajando como desarrolladora en distintas empresas, ningún proyecto se asemejaba al planteado: no solo se debía desarrollar un sistema (lo cual es mi punto fuerte), sino que también me debía hacer cargo de tareas como planificación con el cliente, gestión de metodología ágil para organizar el desarrollo, administración de recursos (devops), documentación de cada etapa de desarrollo, entre otras.

Además, el desarrollo del sistema significó adentrarme en el uso de tecnologías nuevas, con las cuales todavía no había tenido experiencia. Una vez finalizado el desarrollo, pude concluir que las herramientas que nos proporciona la carrera universitaria nos permiten a los estudiantes de Informática volvernos agnósticos a las tecnologías, permitiéndonos adaptarnos a una tecnología nueva sin demasiadas complicaciones. En otras palabras, comprobé que no estamos atados o limitados a una tecnología específica o a un conjunto particular de herramientas; en su lugar, tenemos la capacidad y flexibilidad

para adaptarnos y trabajar con diferentes tecnologías según sea necesario en cada contexto dado.

Si nos concentramos en la etapa de Análisis de Requerimientos, podemos concluir que desde un principio se logró comprender de forma exitosa no solo las necesidades del adoptante sino también sus expectativas. Esto generó que el resto de las etapas se lleven a cabo de una forma más fluida, sin demasiados cambios a los que adaptarse. Obviamente que los requerimientos fueron cambiando a lo largo de todo el proceso de software y esto hizo que también nos adaptáramos a los cambios en cada una de las etapas, pero hoy se puede afirmar que estas variaciones no fueron significativas, dado que se había logrado una buena base confiable.

Por otra parte, la naturaleza de la problemática a resolver era compleja. El sistema que necesitaba El Paseo debía considerar distintos perfiles de usuarios (administradores, consumidores y productores con sus respectivos roles y permisos), manejo de su autenticación y autorización, envío de emails, operaciones CRUD para cada una de sus entidades, generación de reportes de negocio, publicación de noticias, manejo de stock, publicación de videos y fotos, entre otras cosas. Esto también desafió mis habilidades como desarrolladora, ya que las funcionalidades a resolver eran de naturaleza muy variada.

Además, tanto en las experiencias personales como desarrolladora como en los trabajos realizados en la facultad, estaba acostumbrada a trabajar con equipos de desarrollo de alrededor de tres integrantes. Pero para este proyecto, el equipo de desarrollo disminuyó a una sola persona, por lo que naturalmente hubo que adaptarse a realizar todo tipo de tareas.

Como desarrolladores sabemos la importancia de enfocarnos en no reinventar la rueda, es decir, no realizar una sobre-ingeniería para resolver funcionalidades que hoy en día ya están resueltas a partir del uso de una tecnología en particular, por lo que me enfoqué en hacer uso de estas tecnologías siempre que sea posible, recurriendo a la inmensa comunidad de desarrolladores y documentación online, lo que me permitió integrar una tecnología nueva sin mayores complicaciones.

Como estudiante universitaria, busqué que con este proyecto se siga fomentando la participación de estudiantes de Informática en distintas problemáticas sociales, entendiendo que con las herramientas aprendidas realmente podemos generar un gran impacto en la comunidad de la que somos parte.

Al finalizar el proyecto, fue posible entregar un sistema que permita la

comercialización online de productos y también mantenga los lazos sociales que tanto identifican a El Paseo. Se utilizaron herramientas y tecnologías actuales, abiertas, libres y competitivas, reforzando la importancia de promover el software libre y de código fuente abierto.

Concluyo que el producto de software final logró exitosamente resolver los problemas operativos que tenía la comercializadora, principalmente gracias a la capacidad de entender el problema, trabajar para solucionarlo, buscar exceder las expectativas del cliente, adaptarse a los cambios, desarrollar un producto de software de calidad, entregarlo en tiempo y forma, y atender a las tareas de mantenimiento que surgen luego de la puesta en producción.

## 8.2. Trabajo futuro

Del trabajo realizado se desprenden distintos trabajos futuros de distinta naturaleza, los cuales se detallarán a continuación.

En este informe mencionamos las tareas de mantenimiento de software y soporte técnico que se deben brindar como parte del proceso de desarrollo del producto de software. Una vez que se entregue el sistema a El Paseo que actualmente se encuentra finalizando la fase de pruebas, naturalmente surgirán distintas tareas de esta índole que seguramente se mantengan a lo largo del tiempo. Por esta razón, va a ser necesario organizar junto con el cliente cómo se llevaran a acabo estas tareas.

Algo similar ocurre con el mantenimiento de los recursos que requiere el sistema para funcionar. El sistema se configuró para que utilice una serie de recursos acorde a un tráfico inicial reducido, pero la realidad es que esto puede cambiar rápidamente a medida que El Paseo crezca, por lo que seguramente haya que readaptar estos recursos en un futuro, o hasta hacer modificaciones a la arquitectura actual.

Por otro lado, con la realización de este proyecto de tesina cumplimos con la necesidad de El paso de digitalizar sus procesos a partir de una aplicación web. En un futuro, habría que analizar si es necesario el desarrollo de una aplicación móvil, por ejemplo, a partir de detectar que los usuarios suelen acceder al sistema usando sus celulares, o si surgen funcionalidades específicas que serían más efectivas o solo son posibles de implementar en una aplicación móvil, como el acceso a cámara, GPS, notificaciones push, etc.

Con respecto a la deuda técnica, hubo distintas tareas que quedaron sin realizar en este proyecto de tesina para no dilatar los tiempos de entrega del

portal web al cliente. Entre ellos, podemos mencionar:

1. **Unit/Integration tests:** En este proyecto nos concentramos en las pruebas de aceptación del usuario, y dejamos por fuera del alcance el resto de los tests a nuestro código. Como desarrolladores sabemos que esto es una muy mala práctica y que siempre nuestro código debe estar testeado para alcanzar un nivel de confiabilidad del sistema para que el cliente pueda usarlo. Es por esto que sumar estas pruebas ayudaría a mejorar la calidad del sistema.

2. **Integración continua:** Se podría sumar esta herramienta para que cuando los cambios de código se integren automáticamente en un repositorio central se ejecuten pruebas automatizadas en un entorno controlado, para garantizar la calidad del software. Además, con esta tecnología se podría desplegar la aplicación automáticamente en un entorno de prueba para poder realizar pruebas manuales en un ambiente similar al de producción. Ejemplo de estas tecnologías pueden ser Jenkins[1], GitLab CI[2] o CircleCI[3].

3. **Entorno de pruebas separado al de producción:** Es necesario para poder evaluar cambios antes de desplegarlos en producción, aislar el riesgo de afectar la estabilidad y disponibilidad del sistema de producción, validar funcionalidades, correcciones de errores y mejoras de rendimiento antes de lanzarlas al público, entre otras cosas. En este proyecto, el hecho de que se desarrolló un sistema desde cero ayudó a que hagamos nuestras pruebas sin un ambiente dedicado. Pero una vez que se entrega el sistema al cliente, entendemos que es de vital importancia poder realizar distintas pruebas en un ambiente separado sin afectar el productivo.

4. **Agregar tecnologías para debugging y detección de errores:** Esto ayudaría principalmente a las tareas de mantenimiento, ya que favorece la identificación de problemas productivos y la reducción del tiempo de resolución de los mismos, mejorando también la experiencia del cliente y del usuario final. Como ejemplo de estas tecnologías

[1]Sitio oficial: `https://www.jenkins.io/`
[2]Sitio oficial: `https://docs.gitlab.com/ee/ci/`
[3]Sitio oficial: `https://circleci.com/`

podemos mencionar Kibana[4] o NewRelic[5].

5. **Despliegue a producción:** Una vez finalizada la etapa de pruebas y cuando el sistema reúna la confiabilidad necesaria para funcionar en producción, se deberá hacer el despliegue. Para esto, nos reuniremos con el cliente para acordar la contratación de los recursos necesarios para que el sistema funcione, y además se organizarán las tareas de mantenimiento y soporte técnico que naturalmente surgen al desplegar un sistema de software.

Por último, me gustaría hacer referencia a la seguridad de nuestra aplicación. Se podría robustecer el sistema en cuestiones de seguridad siguiendo los estándares y guías de OWASP[6], implementando mejoras en autenticación y autorización, protección contra vulnerabilidades comunes, seguridad en la comunicación, y monitoreo y respuesta a incidentes. Durante la carrera universitaria hay distintas materias optativas relacionadas con la seguridad informática, por lo que alumnos de estas materias podrían aplicar los conceptos aprendidos en un proyecto real, robusteciendo el sistema de El Paseo.

[4]Sitio oficial: `https://www.elastic.co/es/kibana`

[5]Sitio oficial: `https://newrelic.com/es`

[6]OWASP es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro. Sitio oficial: `https://owasp.org/`

# Bibliografía


[1] S. Rial, "Integrando saberes y construyendo redes: la experiencia del paseo de la economía social y solidaria del consejo social de la unlp," 2020.

[2] J. V. Suárez, "Economía social: la experiencia del paseo de la economía social y solidaria," 2019.

[3] L. Barrera, M. Bollini, and M. Samudio, "Proyecto de Identidad: Paseo de la Economía Social y Solidaria."

[4] J. F. Blanco and I. G. Gorosito, "Portal de comercialización online para la Economía Social y Solidaria Local," 2020.

[5] J. L. Coraggio, "Economía social y economía popular: Conceptos básicos," 2020.

[6] L. Fingermann and N. Drago, "El rol de la Universidad en la construcción de economía social y solidaria: los mercados solidarios de la UNLP," 2016.

[7] M. Caracciolo, "Construccion de Tramas de Valor y Mercados Solidarios," 2014.

[8] P. Alvarez, J. I. Fariña, S. Rial, and T. Roque, "Fortaleciendo El Paseo: Compartiendo saberes y afianzando identidad junto a los productores de la Economía Social y Solidaria."

[9] I. Sommerville, *Ingeniería del Software Séptima Edición*. Pearson, 2006.

[10] M. d. C. Gómez Fuentes, J. Cervantes Ojeda, and P. P. González Pérez, *Fundamentos de Ingeniería de Software*. 2019.

[11] M. Arias Chaves, "La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software," *Revista de las Sedes Regionales, Universidad de Costa Rica*, 2005.

[12] M. P. Izaurralde, "Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario," 2013.

[13] S. Kumar, "A REVIEW ON CLIENT-SERVER BASED APPLICATIONS AND RESEARCH OPPORTUNITY," *International Journal of Recent Scientific Research*, 2019.

[14] M. Massé, *REST API Design Rulebook*. O'Reilly Media, 2011.

[15] D. V. Kornienko, S. V. Mishina, and M. O. Melnikov, "The Single Page Application architecture when developing secure Web services," *Journal of Physics: Conference Series*, 2021.

[16] J. F. P. González, F. J. D. Mayo, J. J. G. Rodríguez, and M. J. E. Cuaresma, "Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental," *Revista De Sistemas De información Y documentación*, 2014.

[17] R. M. Stallman, *Software libre para una sociedad libre*. Traficantes de Sueños, 2004.

[18] A. Navarro Cadavid, J. D. Fernández Martínez, and J. Morales Vélez, "Revisión de metodologías ágiles para el desarrollo de software," *PROSPECTIVA, Universidad Autónoma del Caribe*, 2013.

[19] S. D. Amaro Calderón and J. C. Valverde Rebaza, *Metodologías Ágiles*. 2007.

[20] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, 2001. Manifesto for Software Agile Development [Internet].

[21] H. Kniberg and M. Skarin, *Kanban y Scrum – obteniendo lo mejor de ambos*. C4Media Inc, 2010.

[22] M. Bermejo, *El Kanban*. Universitat Oberta de Catalunya, 2011.

[23] S. Chacon and B. Straub, *Pro Git*. Apress Open, 2009.

[24] Flujo de trabajo de Gitflow.

[25] Documentación oficial Versionado Semántico.

[26] N. Jatana, S. Puri, M. Ahuja, and I. K. andDishant Gosain, "A survey and comparison of relational and non-relational database," *International Journal of Engineering Research & Technology (IJERT)*, 2012.