ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UN SUBMODULO E-COMMERCE DE SUSCRIPCIONES PARA SERVICIOS O PRODUCTOS MEDIANTE GOOGLE CLOUD PLATFORM PARA MANTICORE-LABS

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

EDWIN PAUL GUAMUSHIG GUALOTUÑA

edwin.guamushig@epn.edu.ec

DIRECTOR: Ing. Adrián Egüez, MSc.

adrian.eguez@epn.edu.ec

CODIRECTOR: PhD. Edison Loza.

edison.loza@epn.edu.ec

QUITO, DICIEMBRE 2022

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Guamushig Gualotuña Edwin Paul bajo mi supervisión.

MSc. Adrián Egüez, MSc

DIRECTOR DE PROYECTO

Dr. Edison Loza

CODIRECTOR DE PROYECTO

DECLARACIÓN

Yo, Edwin Paul Guamushig Gualotuña, declaro bajo juramento que el trabajo aquí descrito es de nuestra autoría, que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Edwin Paul Guamushig Gualotuña

DEDICATORIA

A mi madre Luz María, quien con su apoyo incondicional me enseñó a luchar por mis objetivos, siempre supo motivarme a seguir adelante y nunca dejar las cosas a medias; pero sobre todo por darme su amor incondicional sin pedir nada a cambio.

A mi padre Galo, que nunca me dejo solo y siempre estuvo al pendiente de mi y supo apoyar me en los momentos de dificultad que tuve.

A mis hermanos Marisol y Eduardo, quienes siempre me han brindado su apoyo moral y han ayudado a que mi paso por la universidad no sea tan complejo como el que ellos pasaron.

AGRADECIMIENTO

En primer lugar, quiero expresar mi gratitud a Dios, por brindarme sabiduría y fortaleza en momentos difíciles, por guiar cada uno de mis pasos y sobre todo por permitirme compartir esta etapa de mi vida junto a mi familia.

A mi madre, quien es un pilar fundamental en mi vida, gracias por todo el esfuerzo que realizó para que no me faltara nada y pudiera llegar a culminar mi carrera universitaria. A mi padre por siempre acompañar me y a veces soportar malas noches por no dejar me solo. Gracias a ambos haberme apoyado incondicionalmente, pese a las adversidades e inconvenientes que se presentaron, siempre estaré eternamente agradecido.

A mis hermanos quienes desde el primer día me compartieron su experiencia y supieron bridar me consejos que me ayudaron durante mi vida estudiantil.

A mi novia Vanessa, quien siempre estuvo incondicionalmente a pesar de tener momentos difíciles, gracias por ayudar me a ser una mejor persona y sobre todo por formar parte de mi vida.

A mi director de tesis, MSc. Adrián Egüez, a quien considero un amigo, pero sobre todo un gran maestro, el cual supo compartir sus conocimientos y me ha ayudado a crecer tanto personal como profesionalmente. Dicho conocimiento ha sido el pilar principal en el desarrollo de este trabajo. Y a mi codirectora, PhD. Edison Loza cuyo apoyo en las revisiones e ideas compartidas fueron de gran ayuda para la presentación de este proyecto de titulación.

Finalmente, agradezco a la Escuela Politécnica Nacional y a la facultad de Ingeniería de Sistemas, por sus enseñanzas y porque gracias a ellas he podido conocer grandes amigos y profesiones con los cuales he establecido una gran y larga amistad.

CONTENIDO

FIGURAS

TABLAS

RESUMEN

Con la aparición del internet y su incorporación en la vida diaria, tanto de las organizaciones como de los hogares, se ha posibilitado la aparición de una nueva tipología de comercio, el comercio electrónico (E-Commerce). Sin embargo, las organizaciones que apuestan por este tipo de comercio enfrentan un importante riesgo: el manejo del stock de los productos que ofertan, ya que no siempre se va a tener una idea clara de cuántos y qué productos se debe disponer para la venta mensual.

Este trabajo se centra en el desarrollo de un submódulo e-commerce enfocado en la suscripción de servicios o productos con reportería en la nube mediante Google Cloud Platform, haciendo uso de la metodología ágil Scrum y herramientas DevOps, para la empresa Manticore-Labs. El submódulo va a permitir a Manticore-Labs ofertar a sus clientes un sistema e-commerce de compras recurrentes con una periodicidad que puede ser fijada entre el vendedor y comprador, permitiéndole al usuario final despreocuparse de las compras y al cliente realizar proyecciones claras de las ventas mensuales.

Además, gracias al uso de Google Cloud Platform no se deberá gestionar ninguna infraestructura física y se podrá hacer uso de herramientas como Big Query la cual va a permitir realizar el análisis de grandes volúmenes de datos en pocos minutos, ayudando al cliente a interpretar tendencias del mercado y dirigir su negocio de una manera más productiva.

Palabras clave: Suscripción, e-commerce, scrum, DevOps, reportería.

ABSTRACT

With the appearance of the internet and its incorporation into the daily life of both organizations and households, a new type of commerce has been born: electronic commerce (E-Commerce). However, organizations that are committed to this type of trade face a significant risk: the management of their stocks. This leads to the state where the company doesn´t have a clear idea of how many products and which of them should be available every monthly.

This work focuses on the development of an e-commerce sub-module for the subscription of services or products with reporting in the cloud through Google Cloud Platform. We used the agile Scrum methodology and DevOps tools, for the Manticore-Labs company. The sub-module will allow Manticore-Labs to offer its customers an e-commerce system of recurring purchases with a periodicity that can be set between the seller and their clients. It would allow end users to forget about purchases and the customer to make clear projections of the monthly sales.

In addition, thanks to the use of Google Cloud Platform, no physical infrastructure should be managed and tools such as Big Query can be used, which will allow us to perform analysis of large volumes of data in minutes. It would also help the client to identify trends in the market and run the business more productively.

Keywords: Subscription, e-commerce, scrum, DevOps, reporting.

1. INTRODUCCIÓN

1.1. **Planteamiento del Problema**

La aparición del internet, su posterior eclosión masiva e incorporación a la vida diaria tanto de las organizaciones como de los hogares, ha posibilitado la aparición de una nueva tipología de comercio, el comercio electrónico (E-Commerce) [1]. Dentro del comercio existe varias actividades, entre las cuales destaca las compras rutinarias que suelen hacerse cada día, semana, mes, año o en un determinado periodo de tiempo, por ejemplo, medicinas, artículos para limpieza, productos cosméticos, flores, etc. En este tipo de compras, precisamente al ser rutinarias, ya está decidido de antemano el producto, la cantidad y la marca que se va a adquirir [2].

Sin embargo, el peligro que enfrentan las empresas que incursionan en el E-Commerce, es el manejo del stock de sus productos, ya que no siempre se va a tener una idea clara de cuantos y que productos deben disponer para su venta.

Además, en la práctica, la recopilación y uso de la información de las ventas no siempre es eficiente; ya que, en la mayoría de los casos, para obtener análisis e informes se utilizan herramientas tradicionales de reportería como Microsoft Excel [3]. Estas herramientas tradicionales proporcionan un resumen de la información sin muchos detalles con informes estáticos que proporcionan solo la información que se solicita. La escasa información no permite formular un plan para aumentar las ventas de productos y por ende mejorar los ingresos [4]. Por otro lado, para el uso de dichas herramientas se necesita espacio físico, recursos, personal capacitado, invertir en mantenimiento y seguridad de toda la información, lo cual añade un coste adicional que muchas empresas no están dispuestas a asumir. Por esto, en la actualidad se ha popularizado el uso de toda la suite de Google Cloud Platform que está preparada para ser gestionada desde cualquier ordenador o dispositivo con conexión a Internet. Esto permite escenarios de trabajo descentralizados, en los que los empleados de una empresa pueden operar a distancia y de forma independiente, con las consiguientes mejoras en la productividad y agilidad del negocio que se derivan [5].

Con el fin de ofertar una solución a los problemas del e-commerce tradicional, el presente trabajo de titulación propone desarrollar un sistema e-commerce enfocado a la suscripción de servicios o producto haciendo uso de la metodología Scrum y herramientas DevOps, lo cual permitirá un desarrollo de software con una alta frecuencia de raleases. El sistema

se encontrará disponible para la empresa Manticore-Labs, la cual podrá ofrecer a sus clientes soluciones de E-Commerce de calidad.

1.2. **Objetivos**

1.2.1. Objetivo general

Desarrollar un submódulo Git E-Commerce para la suscripción de servicios o entrega de productos con reportería en la nube mediante Google Cloud Platform utilizando el marco de trabajo Scrum y DevOps para la empresa Manticore Labs.

1.2.2. Objetivos específicos

- Analizar los requerimientos y las necesidades de los sistemas de suscripción E-Commerce mediante un benchmarking y revisión de literatura.
- Diseñar una solución informática que permita a los usuarios suscribirse a productos o servicios.
- Implementar la solución por medio de SCRUM, utilizando sprints e historias de usuario.
- Aplicar DevOps para obtener escalabilidad, menores tasas de errores en implementación y rapidez en raleases.
- Probar la usabilidad y funcionalidad del sistema.

1.3. **Alcance**

El presente trabajo de titulación tiene como objetivo desarrollar un submódulo web e-commerce por suscripciones para el sistema principal de la empresa Manticore-Labs, el cual permitirá a la empresa ofertar soluciones a los inconvenientes de los sistemas e-commerce tradicionales.

El sistema web también permitirá realizar un análisis escalable de datos que se irán recolectando de los usuarios finales que hagan uso del sistema, haciendo uso de BigQuery el cual es un servicio que oferta Google Cloud Platform.

Finalmente, para tener una aplicación con una tasa de errores bajos en producción se liberarán raleases constantes por medio de herramientas DevOps.

1.4. **Marco Teórico**

En esta sección se detalla conceptos técnicos relacionados a la estructura de la aplicación.

1.4.1. E-Commerce

Se puede decir que el e-commerce consiste en la compra y venta de productos o de servicios a través de medios electrónicos, tales como internet y otras redes informáticas. Originalmente el término se aplicaba a la realización de transacciones mediante medios electrónicos tales como intercambio electrónico de datos; sin embargo, con el advenimiento de internet comenzó a referirse principalmente a la venta de bienes y servicios a través de internet, usando como forma de pago medios electrónicos, tales como las tarjetas de crédito [6].

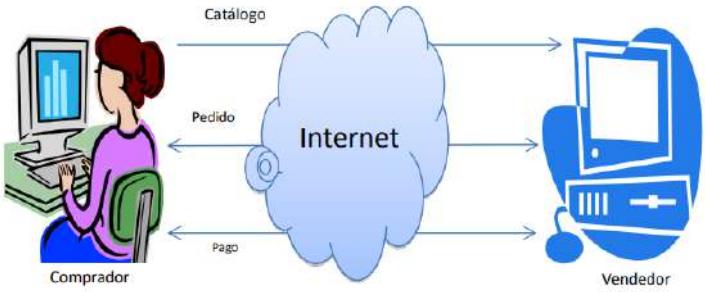


Figura 1: Esquema de e-commerce

Su objetivo es reducir costos en productos y servicios y mejorar la calidad y la respuesta al cliente, al facilitar el proceso de abastecimiento, contribuyendo con la reducción de los ciclos de producción [7].

El e-commerce cuenta con una serie de ventajas respecto al comercio tradicional [8]:

- Disponibilidad 24 horas durante los 365 días del año para el cliente.
- No existen barreras geográficas para el cliente.
- Posibilidad de segmentar a los clientes al trabajar online, mejorando la comunicación y lanzando campañas especializadas.
- Extender el alcance del negocio a nuevos usuarios, pero reducirlo respecto a otros.

1.4.2. E-Commerce por suscripciones

E-commerce por suscripciones es un modelo de negocio con automatización donde un cliente paga por suscribirse a contenidos digitales o a productos y servicios, con frecuencia de compra recurrente. Este permite recibir ingresos por adelantado; así como programar las ventas de forma periódica [9].

En este tipo de modelo el consumidor se despreocupa de realizar compras periódicas de productos que consume de manera recurrente, sean estos cepillos de dientes, cuchillas de afeitar, productos de cosmética, fruta, entre otros; mientras que, desde la perspectiva de la tienda online, permite una mayor fidelización de los usuarios, ya que en lugar de formalizar una compra única y esperar convencer al usuario para que vuelva, se formaliza una compra continuada en el tiempo. Con esto, se puede hacer una proyección más clara y ajustada de ventas, así como poder tener una mayor capacidad de negociación con los distribuidores [10].

1.4.3. Aplicación Web

Las aplicaciones Web no son más que las herramientas de ofimática de la Web 2.0 que se manejan a través de una conexión a Internet. Son aplicaciones de software que se codifican en un lenguaje soportado por los navegadores Web (HTML, JavaScript, Java, etc.) y que confían su ejecución a un navegador web [11].

En las aplicaciones web suelen distinguirse tres niveles:

- Nivel superior que se encarga de interactuar con el usuario.
- Nivel intermedio que se encarga de procesar los datos.
- Nivel inferior que proporciona los datos.

Por lo cual, una aplicación web es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (navegador) como el servidor (servidor web) y el protocolo (HTTP/HTTPS) se comunican entre sí [12], tal como se muestra en la Figura *14*.

1.4.4. Modelo Vista Controlador

Dentro del desarrollo de software por capas, encontramos diferentes modelos, uno de ellos es el MVC (Modelo Vista Controlador) [13]. Este modelo ha llegado a demostrar

su validez a lo lardo de los años en aplicaciones de todo tipo, y sobre todo se ha mostrado su madurez en una multitud de lenguajes y plataformas de desarrollo.

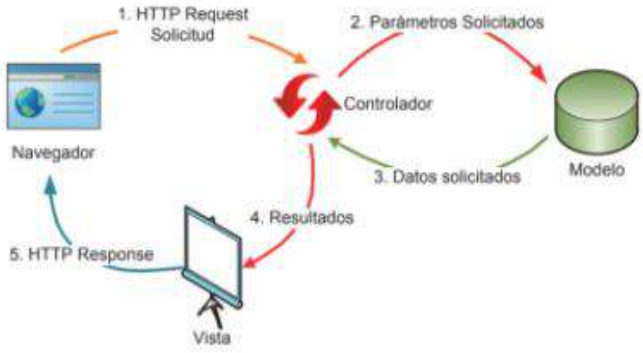


Figura 2: Modelo-Vista-Controlador

El MVC define tres componentes para las capas las cuales buscan separar una aplicación en tres componentes principales:

- Modelo: Accede a la capa de almacenamiento de datos y define la lógica de negocio.
- Vista: Recibe los datos de la capa del modelo y los muestra al usuario.
- Controlador: Realiza peticiones tanto a la capa de modelo como a la vista.

1.4.5. Benchmarking

Permite realizar la evaluación comparativa entre aquellos bienes, servicios y procesos de trabajo administrativos o de producción que pertenezcan a empresas u organizaciones que evidencien las mejores prácticas sobre un área de interés, mediante la recopilación de información [14].

Benchmarking busca identificar lo que otras empresas del mismo negocio hicieron, entender sus métodos y adaptarlos a la realidad de la empresa. A este tipo de técnicas se las llama también de *mejora continua* debido a que constantemente está buscando procesos, prácticas o formas nuevas a implementar.

En el caso de la industria del comercio electrónico, hay muchos aspectos que se podrían identificar para llevar el benchmarking a la práctica [15], entre los cuales se destaca:

- Tienda online: Al momento de crear o mejorar la tienda online, se debe tomar como referencia: ¿Cómo las tiendas exitosas atraen clientes?, ¿Cómo es su atención?, ¿Cuál es la manera de mostrar los productos que oferta?, ¿Cómo es el diseño de la aplicación?
- Estrategia de marketing digital: Observar y analizar el tipo de campañas que las otras marcas están realizando y les está dando éxito, para poder aplicarlo al negocio.
- Atención al cliente: Conocer el servicio Post-Venta de las principales marcas, además de logística de entrega y cómo manejar la garantía de entrega.

Para realizar un análisis se seleccionó los siguientes aplicativos: FastFarma, OpenCart y OsCommerce.

OpenCart

Acorde a su página de presentación, es un sistema de gestión de tienda en línea. Está basado en PHP, utilizando una base de datos MySQL y componentes HTML [16]. En la Figura 3 y Figura 4 se pude observar las pantallas de administración y cliente final que nos ofrece OpenCart.

En la parte administrativa presenta las siguientes funcionalidades:

- Dashboard de reportería
- Plugin para multilenguaje
- Catálogo de categorías, productos, imágenes y perfiles de usuario
- Historial de ventas y devoluciones

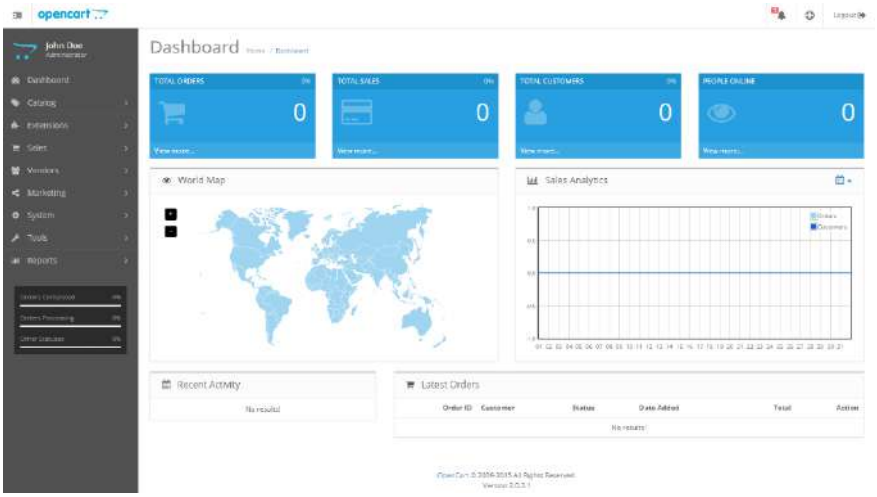


Figura 3: Interfaz de inicio sección administrativa OpenCart

- Listar los productos
- Filtrar productos por categoría
- Visualizar información detallada por cada producto
- Agregar un producto al carrito de compras
- Visualizar estado de carrito de compras

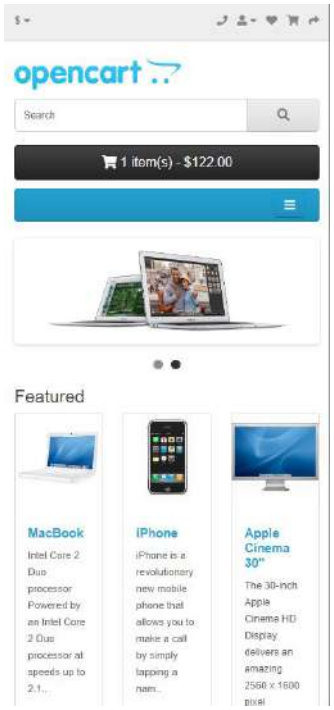


Figura 4: Página de inicio para cliente final, versión móvil

Este aplicativo ofrece a sus potenciales clientes un demo, donde se pude testear las funcionalidades básicas; además de poseer una comunidad que oferta por la compra un soporte de 1 año gratis ante cualquier eventualidad.

Es una aplicación que ofrece grandes funcionalidades, pero una tienda que desee adquirir este producto debe contratar alguien con conocimientos en hosting, instalación de plugins, PHP, Apache y Base de datos ya que, al adquirir la licencia, se entrega un .zip con el código, mas no una URL de uso.

OsCommerce

Es un programa de comercio electrónico y administración en línea desarrollado en PHP por Harald Ponce de Leon y lanzado el 12 marzo de 2000. Requiere de una base de datos MySQL y un servidor Apache [17] . En la Figura 5 y Figura 6 se pueden observar las pantallas de inicio, tanto para la parte administrativa, como para la parte del cliente final.

El aplicativo de administración ofrece las siguientes funcionalidades:

- Listado de clientes
- Catálogo de productos, review y categorías
- Reporte de ventas realizadas
- Configuración de banners y cupones de descuento.
- Administración de usuarios y roles
- Dashboard de ventas por periodo

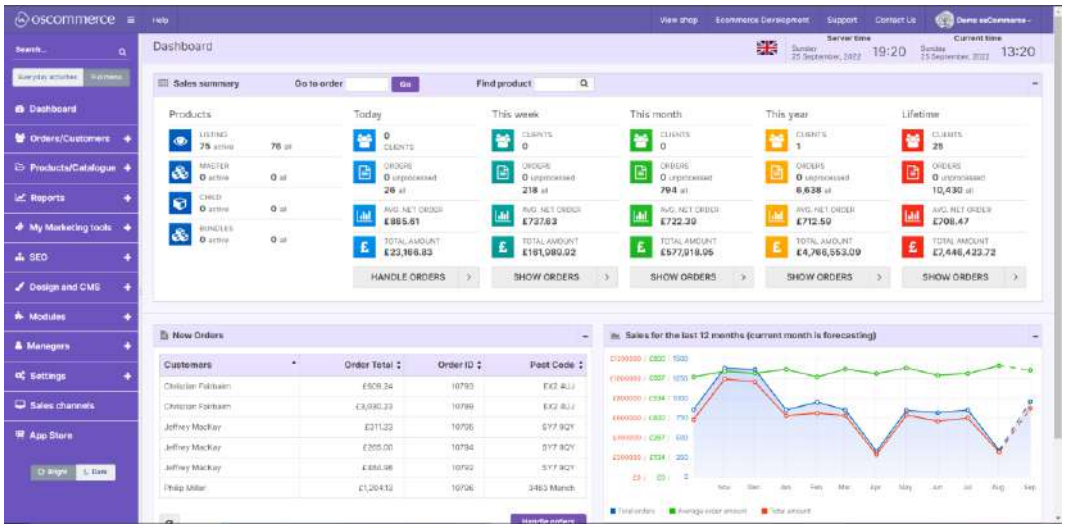


Figura 5: Pantalla administrativa de OsCommerce

En el aplicativo del cliente final se puede:

- Realizar pagos a través de PayPal y tarjetas de crédito
- Listar y filtrar productos
- Agregar al carrito de compras
- Ver detalle de producto
- Administrar los productos que se encuentran en el carrito de compras.
- Adjuntar reseñas por producto

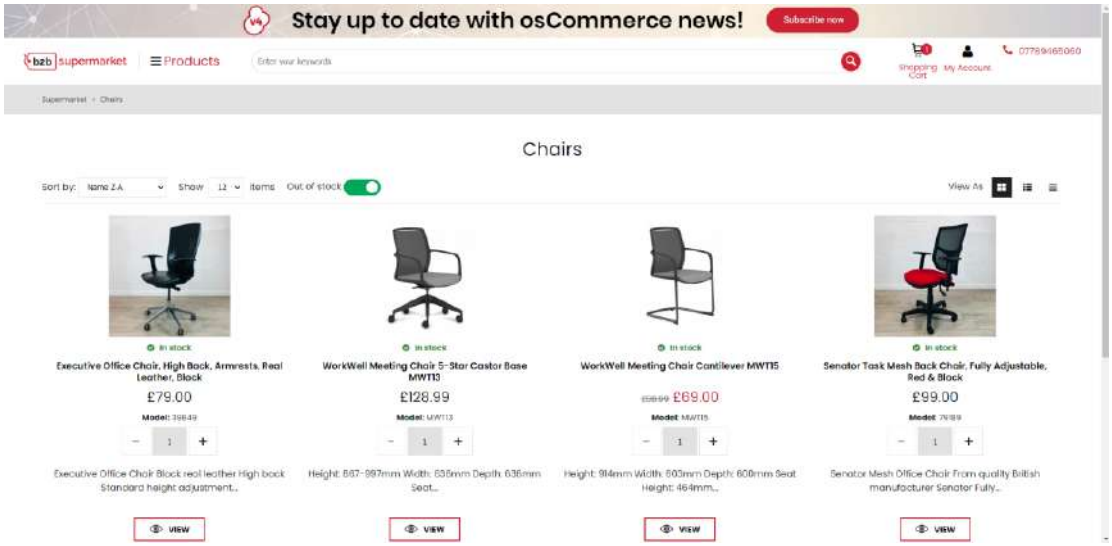


Figura 6: Pantalla de cliente final OsCommerce

Este aplicativo es OpenSource, por lo cual cualquier tienda que quiera implementarlo para vender sus productos deberá contratar una persona que tenga conocimientos en hosting, PHP, Apache y Base de datos; ya que, en la página oficial se puede descargar únicamente el código fuente, por lo que es necesario realizar un proceso para la instalación de dependencias y poder tener un aplicativo funcional. Además, se deben realizar pagos en caso de querer utilizar la pasarela de pagos que viene integrada.

FastFarma

Es una tienda perteneciente a la farmacia digital FastFarma, en donde se realiza la venta de productos médicos a través de suscripciones. Actualmente se encuentra disponible en Ecuador y México. Este sistema cuenta con las siguientes funcionalidades:

- Venta de productos de manera normal o por suscripciones
- Listado de productos médicos
- Filtros por categoría
- Agregar productos al carrito
- Administrar productos que se encuentran en el carrito
- Pagos a través de PayPhone
- Utilizar cupones para descuentos
- Envió de productos a ubicación donde se realizó la compra

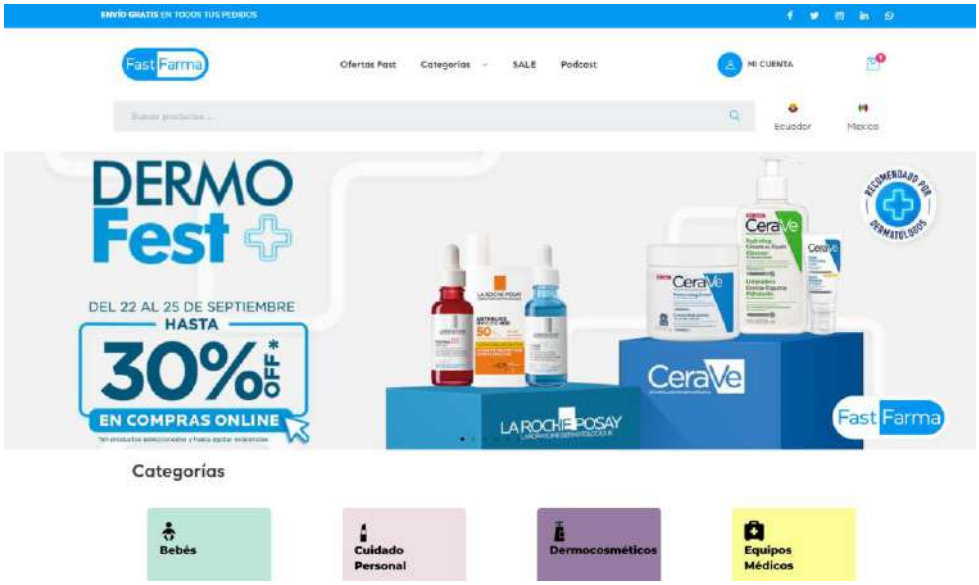


Figura 7: Pantalla de inicio aplicativo de FastFarma

1.4.6. Revisión de literatura

Se optó por la revisión de 3 papers, estos fueron elegidos ya que nos permite conocer como ha ido evolucionando la venta de productos por internet y cuales han sido los módulos de mayor impacto dentro de un aplicativo, además de permitirnos comparar los distintos cambios que ha sufrido la venta de productos por internet los cuales trataban temas referentes a aplicaciones e-commerce y el nuevo modelo de suscripciones a productos, el cual permite a los negocios que utilizan este nuevo modelo de negocio optimizar su stock y planificar sus ventas.

A continuación, en la Tabla 1 se muestra un breve resumen de las características a destacar de cada uno de los aplicativos junto a las funcionalidades sugeridas durante la revisión de literatura.

Tabla 1: Funcionalidad básica de aplicativos similares

Funcionalidad	OpenCart	OsCommerce	FastFarma	Revisión de literatura
Gestión de grupos	X	X		X
Gestión de precios	X	X		X
Gestión de productos	X	X		X
Gestión de imágenes por producto	X	X		X
Lista de reseñas		X		X
Gestión de usuarios	X	X		
Lista de ventas	X	X		X
Filtros dinámicos	A partir de la versión de pago	X		
Dashboard de ventas	X	X		X
Reporte de ventas		X		
Registro de cliente	X	X	X	X
Recuperar contraseña	X	X	X	X
Lista de productos	X	X	X	X
Detalle de productos	X	X	X	X
Dejar reseña de producto	X	X		X
Agregar producto como compra por suscripción			X	X
Agregar producto como compra normal	X	X	X	X
Pago online	A partir de la versión de pago	A partir de la versión de pago	X	
Estado de carrito de compras	X	X	X	X
Gestionar carrito de compras	X	X	X	X
Gestión de Banners			X	X
Manejo de Kardex		X		X

1.4.7. DevOps

El término DevOps, es una combinación de los términos ingleses development (desarrollo) y operations (operaciones), designa la unión de personas, procesos y tecnología para ofrecer valor a los clientes de forma constante [18].

DevOps como cultura trata de dar solución a los problemas que enfrenta el negocio, en especial el Time-to-Market que sin duda es un factor clave que permite a un producto ser competitivo [19]. Time-to-market promueve una comunicación continua más fluida, así como colaboración, integración, visibilidad y transparencia entre los equipos de desarrollo de aplicaciones (Dev) y sus homólogos en operaciones tecnológicas (Ops). Las herramientas DevOps permiten, a su vez, automatizar la entrega de software y la infraestructura necesaria para garantizar un proyecto de alta calidad.

1.4.8. Infraestructura como código (IaC)

La infraestructura como código o por sus siglas en ingles IaC permite a los equipos de DevOps ofrecer entornos estables y escalables de manera rápida [20], ya que permiten gestionar y preparar la infraestructura a través de líneas de código, en lugar de hacerlo mediante procesos manuales.

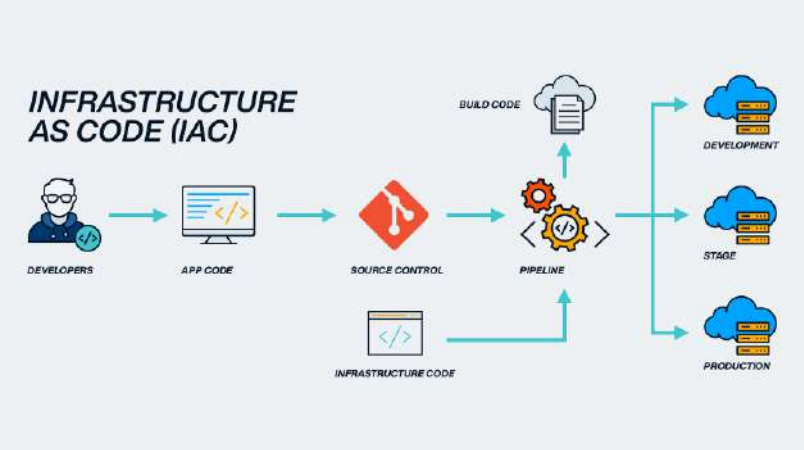


Figura 8: Esquema de infraestructura como código

Con este tipo de infraestructura, se crean archivos de configuración que contienen las especificaciones que esta necesita, lo cual facilita la edición y la distribución de las configuraciones entre los diferentes equipos.

1.4.9. Submódulo GIT

Es un mecanismo que ofrece GIT para clonar un repositorio secundario dentro de una carpeta del repositorio principal, de manera que se puedan manejar las distintas versiones de código de ambos repositorios por separado. Pero a la vez permite

referenciar desde el repositorio principal al secundario; esto permite clonar otro repositorio en su proyecto y mantener los commits separados [21].

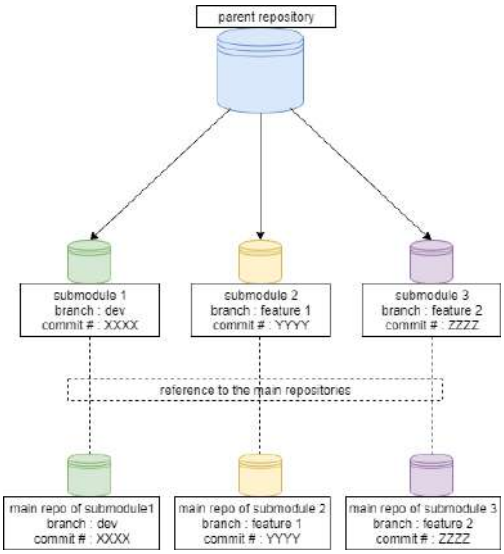


Figura 9: Esquema manejo de submódulos GIT

1.4.10. Google Cloud Platform

Gracias a la computación en la nube, hoy en día los productos de hardware y software coexisten de forma remota en centro de datos. Google Cloud consiste en un conjunto de recursos físicos, como computadoras y unidades de disco duro, y recursos virtuales, como máquinas virtuales (VM), que se encuentran en los centros de datos de Google en todo el mundo. Cada centro de datos está ubicado en una región, específicamente en Asia, Australia, Europa, América del Norte y América del Sur [22].

Las herramientas que ofrece Google Cloud Platform ayudan a proteger y controlar los datos en todas las etapas de su ciclo de vida, ya que estos se encriptan automáticamente [22]. No se tendrá que gestionar ninguna infraestructura, se podrá utilizar fácilmente productos como BigQuery para analizar grandes volúmenes de datos en cuestión de minutos, en lugar de meses [22]; y gracias a esto, se podrá interpretar las tendencias del mercado y dirigir el negocio de una manera más ágil.

1.4.10.1. BigQuery

BigQuery es el servicio empleado para hacer analítica de datos en Google Cloud. Con BigQuery se pueden realizar consultas tipo SQL 2011 standard y legacySQL sobre

tablas, sin importar su tamaño, y estas consultas se ejecutarán en el menor tiempo posible (mucho más rápido que en Cloud SQL). Hay que tener en cuenta que BigQuery no está diseñado para servir como una base de datos transaccional (responder a un gran número de consultas rápidamente) [23].

El almacenamiento de Google BigQuery es completamente administrado, sin servidores, con gran escalabilidad y con escalabilidad de hasta petabytes; además tiene integración con diversas herramientas tales como [24]:

- ETL (Informática y Talend).
- BI (Tableau, MicroStrategy, Looker y Data Studio).
- Cloud Storage (almacenamiento de objetos).
- Cloud Bigtable (bases de datos transaccionales).
- Hojas de cálculo en Drive para procesar fuentes de datos externas sin duplicar datos.
- Google Marketing Platform.
- Google Ads.

1.4.11. Metodología Ágil

Kent Beck, uno de los fundadores del desarrollo de software Agile, describe a las metodologías ágil como un grupo de métodos de desarrollo de software basados en un desarrollo iterativo e incremental en el que los requisitos y las soluciones evolucionan a través de la colaboración entre equipos autoorganizados y multifuncionales [25].

Figura 10: Proceso ágil de desarrollo de software [26]

1.4.11.1. SCRUM

Scrum es un método para solventar problemas complejos entregando productos que aporten el mayor valor posible [26]. Se basa en la teoría de control de procesos empírica o empirismo, el empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Esta metodología emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo, y realizar entregas incrementales del proyecto [27].

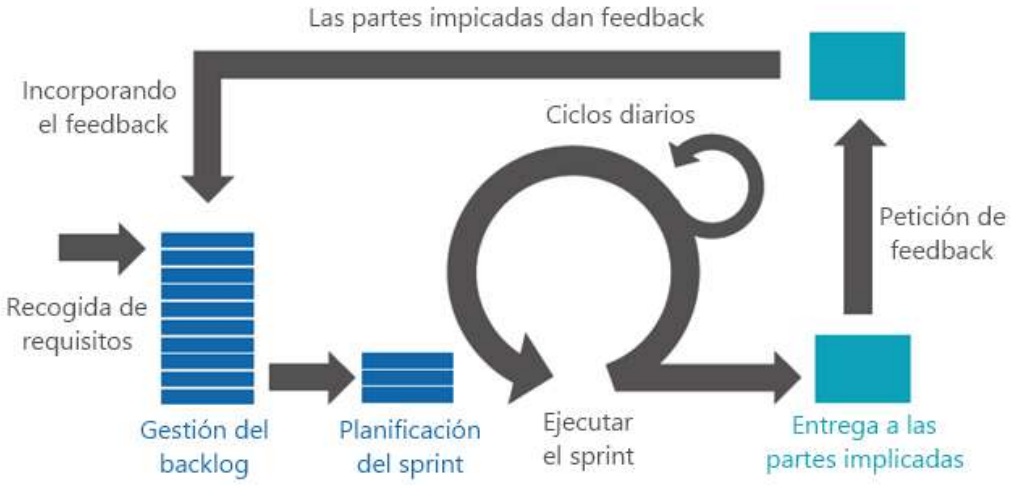


Figura 11: Esquema de proceso de desarrollo Scrum [27]

Scrum es una metodología [26]:

- Ligera: Scrum tiene poca teoría, únicamente define algunas reuniones o ceremonias, roles, y unos pocos principios básicos. El contenido teórico se lee en menos de 5 minutos.
- Fácil de entender: Es una metodología abierta, que no propone reglas complicadas ni demasiado específicas en función del proyecto.
- Difícil de dominar: La clave es adaptarla correctamente al entorno y al proyecto concreto. Por eso está definido el rol de Scrum Master, que es la figura que domina el método y ayuda a su aplicación y ajuste

EQUIPO

La metodología Scrum se construye alrededor de tres roles [28]:

- El Product Owner
- El Scrum Master
- El equipo de desarrollo

Product Owner (*comúnmente llamado PO*) el Propietario del Producto es responsable de maximizar el valor del producto resultante del trabajo del equipo de Scrum. También es responsable de la gestión eficaz de la pila del producto (Product Backlog), que incluye [29]:

- Desarrollar y comunicar explícitamente el objetivo del producto
- Creación y comunicación clara de elementos de trabajo pendiente del producto
- Pedido de artículos de trabajo pendiente del producto
- Asegurarse de que el trabajo pendiente del producto sea transparente, visible y comprendido

Scrum Master es el "facilitador", tiene la misión de eliminar los obstáculos que pueden aparecer a los miembros del equipo, garantizando que el método Scrum se aplica correctamente.

Equipo de desarrollo (*también llamado Scrum Team*) reúne funciones heterogéneas y desarrolla las historias de usuario contendidas en el Product Backlog, con el objetivo de ofrecer un entregable de calidad. Scrum no realiza diferencias entre los diferentes miembros del equipo por lo tanto el equipo de desarrollo se autoorganiza, es decir que ninguna persona tiene la autoridad sobre la manera en la que otra debe realizar su trabajo, ni sobre lo que la otra debe realizar [28].

EVENTOS DE SCRUM

A las diversas reuniones que conocemos propias de la gestión de un proyecto tradicional, en el método Scrum se añaden un determinado número de eventos específicos los cuales son [28]:

- Sprint
- Sprint planning meeting
- Daily scrum meeting
- Sprint review meeting
- Retrospectiva del sprint

Tabla 2: Eventos de SCRUM

Tarea	Descripción	Responsables
Sprint	Son ciclos iterativos en los cuales se desarrolla o mejora una funcionalidad para producir nuevos incrementos. [30], tienen una duración fija y determinada, entre 1 y 4 semanas.	
Sprint planning meeting	Se establece que ítems de la Product Backlog List van a ser realizados durante el Sprint. Esto se realiza a partir de lo que el Scrum Team considera que puede construir durante el Sprint [30]. También se decide como se van a alcanzar los objetivos del Sprint.	Product Owner Scrum Master Scrum Team
Daily scrum meeting	Tienen una duración de 15 minutos y los participantes se quedan parados. Estas reuniones no se utilizan para resolver problemas. En ellas se realizan tres preguntas [30]: · ¿Qué hiciste ayer? · ¿Qué harás hoy? · ¿Qué obstáculos ves en tu camino?	Scrum Team
Sprint review meeting	Reunión que se mantiene al final de cada Sprint para inspeccionar el Incremento de producto, y adaptar el Backlog del producto si es necesario [26]. Durante el Sprint Review, el equipo de trabajo muestra al resto de interesados qué se ha conseguido en el sprint.	Producto Owner Cliente Otros interesados
Retrospectiva del sprint	Inspecciona cómo ha ido el sprint, en lo referente a las personas, sus relaciones, el proceso y las herramientas. Se identifican y ordenan los asuntos más importantes, tanto los que fueron bien, como los que suponen una mejora potencial [26].	Product Owner Scrum Master

ARTEFACTOS DE SCRUM

Product Backlog consiste en la lista de requisitos de usuario, que a partir de la visión inicial del producto crece y evoluciona durante el desarrollo.

Sprint Backlog es la lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto.

Incremento representa el resultado obtenido de cada sprint.

Gráfico burn down o grafico de avance, el cual es actualizado a diario por el Scrum Team con el fin de comprobar los avances de las tareas durante el sprint.

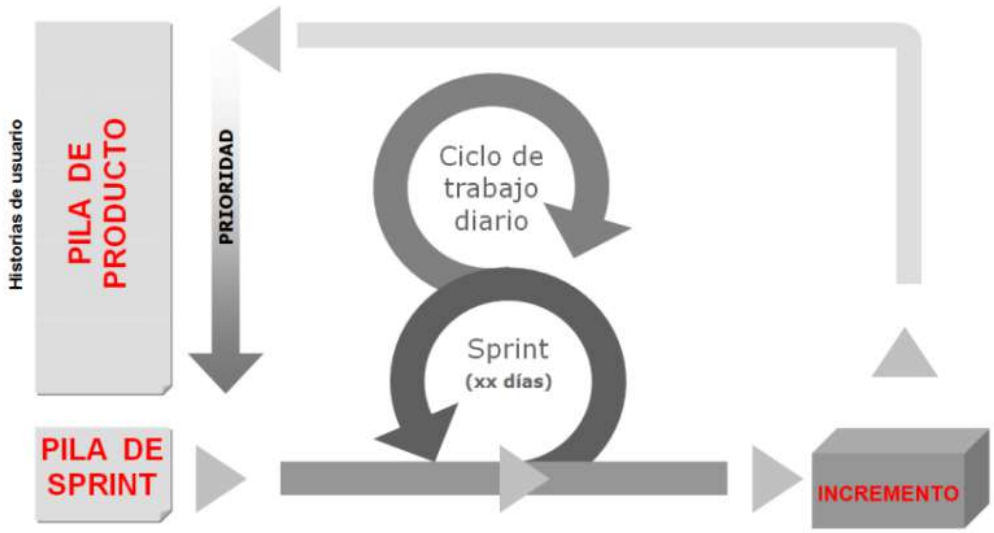


Figura 12: Diagrama del ciclo iterativo scrum

1.5. **Lenguajes, Herramientas, Frameworks y Librerías**

A continuación, se describe las herramientas, frameworks y librerías que serán utilizadas para el desarrollo del presente proyecto.

Tabla 3: Descripción de herramientas, frameworks y librerías a utilizar

Nombre	Descripción	Utilizado en
TypeScript	Es un lenguaje de programación de alto nivel que implementa muchos de los mecanismos más habituales de la programación orientada a objetos, pudiendo extraer grandes beneficios que serán especialmente deseables en aplicaciones grandes, capaces de escalar correctamente [31]. La característica fundamental de TypeScript es que compila en JavaScript nativo.	Aplicación web
NodeJs	NodeJs es un entorno de ejecución multiplataforma de código abierto para el lenguaje de programación JavaScript [32]. utiliza un modelo de entrada y salida sin bloqueo controlado por eventos que lo hace ligero y eficiente (con entrada se refiere a solicitudes y	Aplicación web

	con salida a respuestas). Puede referirse a cualquier operación, desde leer o escribir archivos hasta hacer una solicitud HTTP.	
 **NestJs**	Es un framework desarrollado por completo en TypeScript, con enfoque al lado del servidor [33]. Al ser desarrollado en TypeScript nos permite combinar elementos de POO (programación orientada a objetos) y PF (programación funcional).	Aplicación web
 **Bull Queue**	Bull es una biblioteca de nodos que implementa un sistema de cola rápido y robusto basado en redis.	Aplicación web
 **MySql**	Es un sistema de gestión de base de datos relacionales de código abierto, con un modelo cliente-servidor.	Aplicación web
 **Angular**	Es un framework opensource desarrollado por Google, cuya finalidad es facilitar el desarrollo de aplicaciones web SPA (Single Page Aplication) [34]. Hace uso de HTML, CSS y TS, además de clases o componentes que permiten realizar una aplicación más escalable y mantenible.	Aplicación web
 **Man-lab-nest y Man-lab-ng**	Paquetes para Angular y NestJs, desarrollados por el equipo de Manticore-Labs, disponible en el repositorio NPM; permite simplificar el trabajo y realizar gestiones de manera óptima y rápida.	Aplicación web

tailwindcss	Es un framework CSS el cual perite un desarrollo ágil, basado en clases que se pueden aplicar con facilidad en el código HTML; y unos flujos de desarrollo que permiten optimizar mucho el peso del código CSS [35].	Aplicación web, estilos CSS
GitLab	Es una potente solución de software libre que permite crear y gestionar repositorios de código y documentos [36]. Proporciona los usuarios un manejo de control de versiones de repositorios Git, ofrece herramientas de planificación, de gestión de wikis, de seguimiento de problemas (Issue) y DevOps.	Control de versiones y DevOps
LucidChart	Es una herramienta web, que permite trabajar de manera colaborativa y en tiempo real, en la creación de diagramas y diseño de interfaces de bajo nivel.	Diseño del modelo de base de datos y arquitectura de la aplicación.
Figma	Es una poderosa herramienta de diseño de interfaz colaborativa, utilizada por diseñadores de UX/UI [37].	Prototipado
WebStorm	Es un IDE desarrollado por JetBrains, tiene características excelentes, que incluyen edición en vivo y sugerencias de código, o Intellisense. La edición en vivo le permite mantener un navegador abierto que se actualizará automáticamente en función de los cambios en CSS, HTML y JavaScript. Las sugerencias de código, resaltara el código escrito y sugiere mejores formas de implementarlo [38].	IDE

2. METODOLOGÍA

Para el desarrollo del siguiente proyecto se hizo uso de la metodología SCRUM, ya que la misma nos permite realizar un desarrollo incremental y un enfoque ágil. Debido a esto se decidió, junto a el equipo implementar lo siguiente:

- Duración de sprint de 15 días
- Escribir historias de usuario sencillas, y con criterios de aceptación claros.
- Estimar historias de usuario haciendo uso de FirePoker
- Realizar un Aseguramiento de la Calidad (QA – Quality Assurance) de cada una de las tareas durante el Backlog Grooming para mantener un tablero lo más actual posible.
- Diseño simple de la aplicación.

2.1. **Equipo SCRUM**

Un equipo scrum trabaja de manera horizontal, por lo cual no existirá jerarquías dentro del mismo, sino que será una unidad cohesiva de profesionales enfocados en un objetivo a la vez [39]. Los roles dentro del equipo son:

- **Scrum Master**: Msc. Adrián Eguez; responsable de ayudar al equipo y organización (Manticore-Labs) a comprender la teoría y práctica de SCRUM.
- **Product Owner:** Ing. Crishtian Lara; responsable de maximizar el valor del producto como resultado del trabajo del equipo.
- **Desarrollador:** Edwin Guamushig; responsable de la creación del producto.

2.2. **Requerimientos**

Tomando como punto de referencia el benchmarking realizado a las tiendas e-commerce y la entrevista con el Product Owner, donde se socializo el estado actual del sistema realizado por Manticore-Labs y los procesos que se manejan internamente; se decidió crear un nuevo submódulo, que se integrará con las funcionalidades ya existentes y que será totalmente transparente para el equipo de desarrollo.

En consenso con el Product Owner, se decidió agrupar los siguientes módulos dentro del submódulo e-commerce:

Tabla 4: Requerimientos

Módulo	Requerimiento
Impuestos	Gestionar los impuestos a aplicar en las compras de manera dinámica.
	Acceso restringido al administrador.
	Habilitar y deshabilitar impuestos de manera dinámica.
	Los impuestos deshabilitados no deben mostrarse en el cálculo final de una compra.
	Los impuestos deben ser manejados de manera independiente por cada empresa.
Artículos por empresa	El administrador puede gestionar los artículos y el stock disponible, y el cual se mostrará en la plataforma e-commerce.
	Para cada artículo se debe registrar un precio base.
	Se debe visualizar el stock disponible de cada producto o servicio.
	Visualizar histórico de reseñas realizadas por clientes.
	Una empresa no debe tener dos artículos con el mismo código.
Precios	Cada artículo empresa puede tener asociado al menos un precio para poder ser visualizado en la página e-commerce.
	Debe existir dos tipos de precios: NORMAL y PROMOCIÓN.
	Solo debe existir un precio habilitado por cada producto.
	El acceso para la edición o creación de nuevos precios es visual solo para el administrador.
Pagos	Registrar los montos e información de pagos realizados en la plataforma e-commerce.
	Historial de pagos.
	Generar reportaría con BigQuery.
Suscripciones	Historial de suscripciones.
	Manejar estados para las suscripciones en la plataforma: FINALIZADO, EN PROCESO, CANCELADO.
	El administrador podrá cambiar el estado de una suscripción.
	Cada suscripción debe estar asociada a un cliente y a una empresa.
	Generar reportería con BigQuery.
Clientes	Información de clientes registrados en la plataforma.
	El administrador podrá recuperar la contraseña de un cliente.
	Histórico de pagos realizados por el cliente.
	Por cada cliente, visualizar el estado de sus suscripciones.
Plataforma e-commerce	Login y registro de usuario.
	Recuperar contraseña.
	Filtro por grupo y subgrupo.
	Banner personalizable desde BackOffice, según necesidades de la empresa.

	Paginación para la lista de productos disponibles.
	Información detallada de producto.
	Manejo de calificación y reseñas de cada producto.
	Manejar ventas de productos o servicios de manera normal o por suscripción.
	Carrito de compras, actualizado en tiempo real.
	Detalle de compra.
	Información de pago.
	Historial de compras.
	Historial de suscripciones.

Además de los requerimientos definidos para desarrollo en el sistema, se consideró la implementación de un marco de trabajo DevOps, lo cual nos permitirá tener un desarrollo del aplicativo más eficiente y realizar entregas continúas permitiendo al usuario final realizar pruebas continuas antes de la fase de despliegue a producción.

2.3. **Prototipado de media fidelidad**

Una vez finalizada las primeras reuniones con el Product Owner y ya plasmadas las necesidades y requerimientos mínimos del sistema, se procedió con la implementación de mockups haciendo uso de Figma, herramienta que permite diseñar interfaces amigables para el usuario.

Con el Product Owner se llegó al acuerdo de maquetar las interfaces de la plataforma e-commerce, las cuales se pueden observar en la Figura 8.

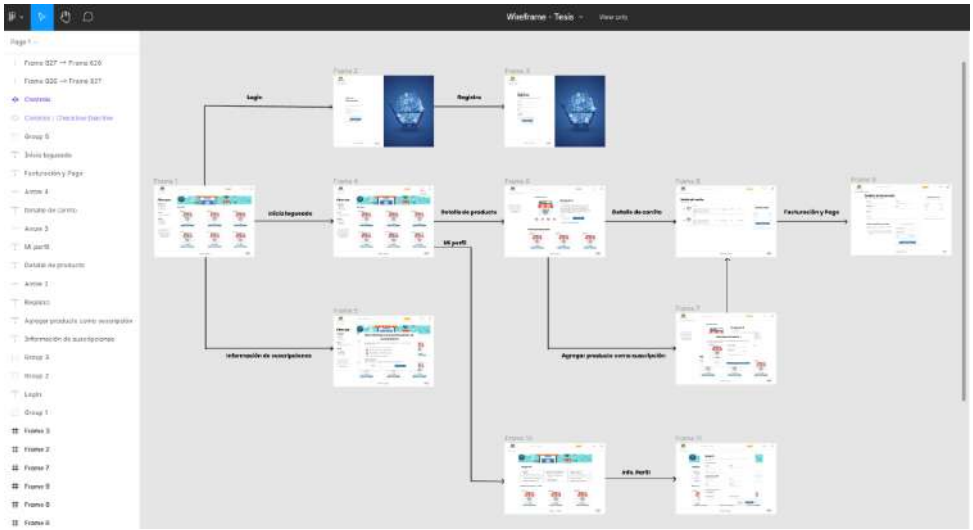


Figura 13: Mockups plataforma e-commerce

2.4. **Arquitectura de la aplicación**

Una vez definido los requerimientos y aprobados los mockups por parte del Product Owner, se decidió hacer uso del patrón de diseño Modelo-Vista-Controlador (MVC) y Atomic Design, lo que nos permitirá desarrollar una aplicación escalable y mantenible en el tiempo.

Para la generación de reportes se hará uso del Cloud Data Warehouse de Google / BigQuery, el cual nos permitirá administrar y analizar los datos recolectados por el aplicativo. Toda la información se almacenará en una base de datos MySQL.

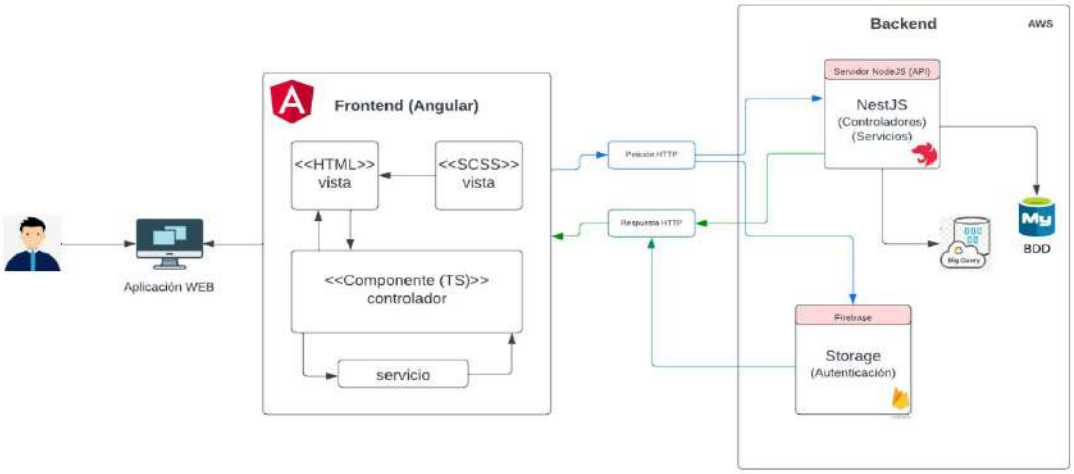


Figura 14: Arquitectura del sistema

Finalmente, para la parte de DevOps se hará uso de un nuevo marco de trabajo llamado Infraestructura como Código (IaC), el cual nos permitirá orquestar la infraestructura de los servidores donde se realizará el despliegue a producción de la aplicación a través de código.

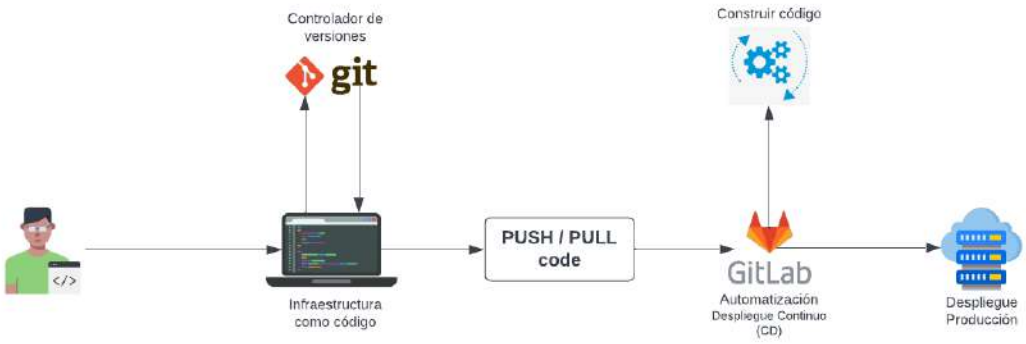


Figura 15: Arquitectura Infraestructura como código

2.5. **Modelo de base de datos**

El diseño de la base de datos SQL para el submódulo e-commerce se observa en la Figura *17*. Este diseño siguió estándares internos de la empresa, como se describe en la Figura *16*; la cual está compuesta por 5 entidades que serán implementadas en la base de datos MySQL para su integración con el sistema base de Manticore Labs.

Estas entidades serán encargadas de almacenar la información de historial de compras, suscripciones a servicios o productos, reseñas de productos, preguntas frecuentes, impuestos para cada producto, información de usuario, facturación e información de pago.

Obligatorios, Indices, Posibles valores
***C** Obligatorio Crear **C** Puede Crearse **U** Actualizarse **I** Indice (puede usarse para búsqueda) **(valor1, valor2, valor3, valor4)** Valores posibles (1, 0)

Figura 16: Estándares Manticore-Labs para diagramas entidad relación

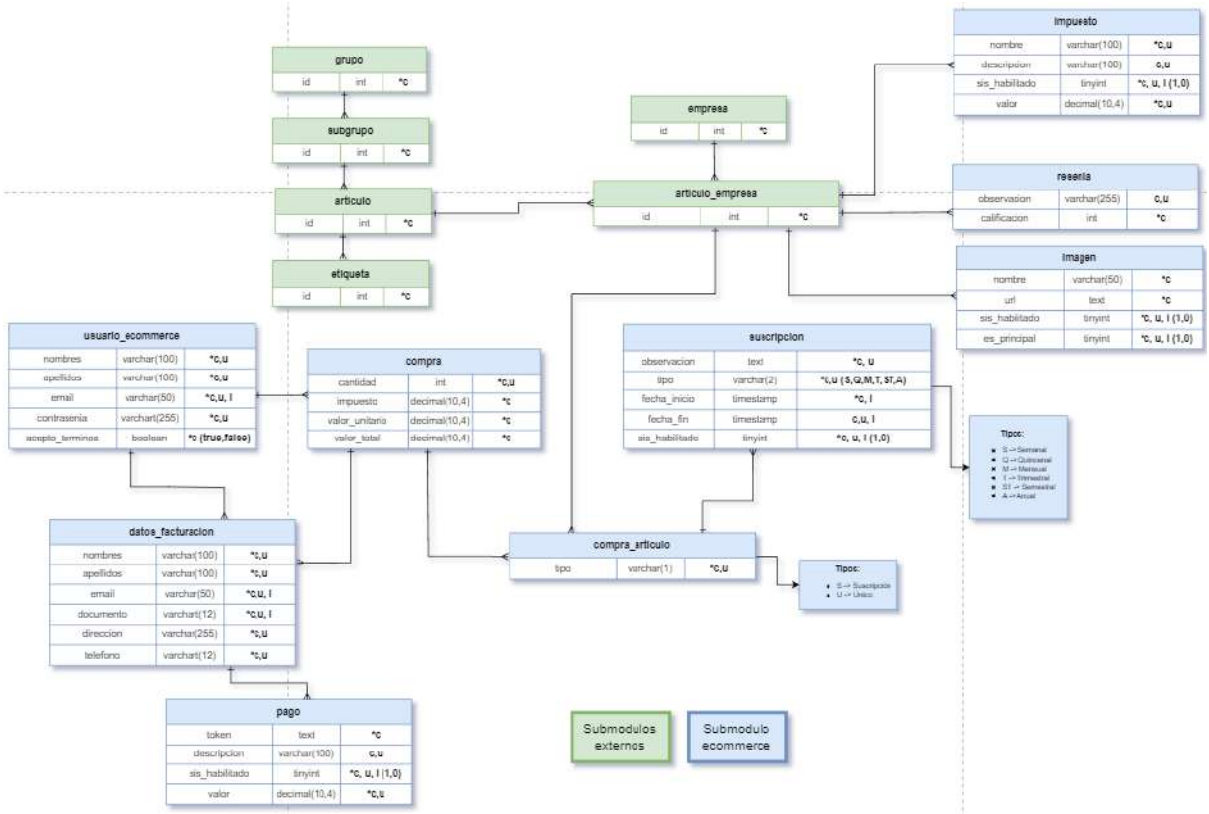


Figura 17: Modelo base de datos

2.6. **Diagrama de navegación**

En la Figura 18 se muestra las rutas y módulos que tendrá el sistema para la parte de backoffice, mientras que en la Figura 19 se muestran las rutas que tendrá el sistema E-Commerce, las cuales serán visualizadas por el cliente final:

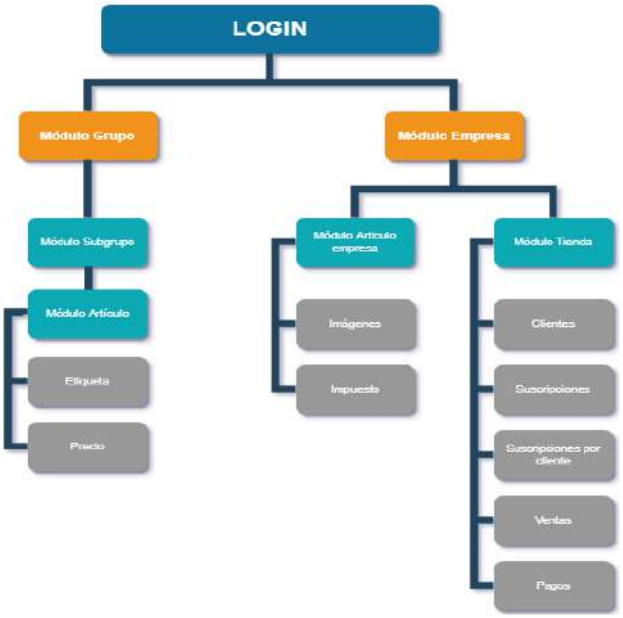


Figura 18: Diagrama de navegación BackOffice

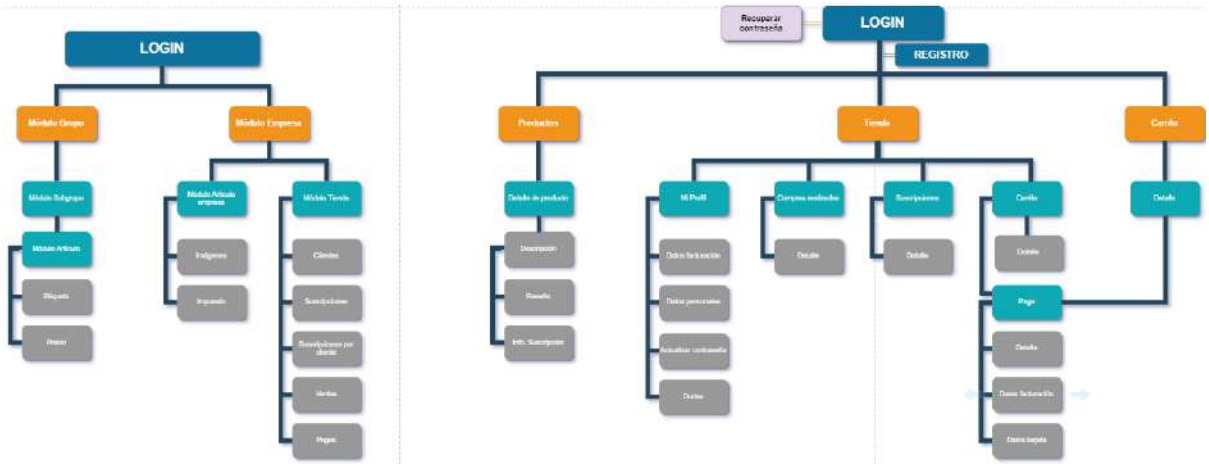


Figura 19: Diagrama de navegación sistema e-commerce

2.7. Desarrollo de la aplicación - Product Backlog

En la Tabla *5*, se detalla el Product Backlog inicial del proyecto. Se debe tener en cuenta que a medida que el proyecto avance se pueden ir aumentando nuevas historias a esta lista de requerimientos, ya sean estas para agregar nuevas características, mejoras o corrección de bugs.

Tabla 5: Product backlog inicial

Product Backlog			
Código	**Nombre**	**Peso**	**Prioridad**
EC-01	Seteo tecnológico	13	Alta
EC-02	Configuración CI	2	Alta
EC-03	Gestión de etiquetas por artículo	5	Media
EC-04	Gestión de artículos por empresa	5	Alta
EC-05	Gestión de imágenes por artículo empresa	13	Baja
EC-06	Gestión de impuestos por artículo empresa	3	Media
EC-07	Gestión de precios por artículo empresa	3	Media
EC-08	Registro de pagos	3	Baja
EC-09	Gestión de suscripciones	8	Media
EC-10	Gestión de clientes	8	Media
EC-11	Generar reportaría con BigQuery	20	Media

EC-12	Listado reseñas por artículo empresa	3	Baja
EC-13	Login de usuario	8	Alta
EC-14	Registro de usuario	8	Alta
EC-15	Recuperar contraseña	5	Baja
EC-16	Filtros por grupo y subgrupo para e-commerce	8	Alta
EC-17	Listar artículos por empresa	5	Alta
EC-18	Maquetar página inicial de tienda e-commerce	20	Alta
EC-19	Modal con información de suscripción y opción de enviar pregunta	3	Baja
EC-20	Detalle de producto seleccionado	20	Alta
EC-21	Modal para cambio de tipo de compra a suscripción	8	Media
EC-22	Agregar reseña y calificación a un producto	5	Alta
EC-23	Carrito de compras	20	Alta
EC-24	Detalle de carrito	13	Alta
EC-25	Información de pago y facturación	13	Media
EC-26	Perfil de usuario	8	Media
EC-27	Gestión de etiquetas	3	Alta
EC-28	Configuración de colas con Redis	13	Alta
EC-29	Generar suscripción a productos por medio de colas	20	Alta
EC-30	Listado de productos vendidos	3	Baja
EC-31	Configuración de CD	13	Media
EC-32	Configuración de cuenta para envío de correos	5	Baja

2.8. Sprint 0

Objetivos:

- Configurar los repositorios, frameworks y librerías, para tener un ambiente de desarrollo estable.
- Agregar los submódulos e-commerce al proyecto principal tanto a nivel de backend como de frontend.

2.8.1. Sprint Planning

En este sprint inicial se realizará el *seteo* tecnológico, donde se realizará la creación del submódulo e-commerce dentro de los repositorios de GitLab de la empresa Manticore-Labs. Se configurará el ambiente de desarrollo para el frontend y backend utilizando Angular y NestJs respectivamente, además de configurar las librerías visuales como Tailwind, PrimeNG y herramientas de análisis de código como ESLint.

En la Tabla *6*, se lista las historias de usuario sobre las que se trabajará durante el sprint; mientras que en el Anexo 6.6 se encuentra el detalle de cada una de ellas.

Tabla 6: Sprint planning inicial

Sprint 0			
Código	**Nombre**	**Peso**	**Prioridad**
EC-01	Seteo tecnológico	13	Alta
EC-02	Configuración CI	2	Alta

2.8.2. Implementación

Para iniciar el sprint se crearon tres repositorios (Figura *20*), dentro de la cuenta de GitLab de Manticore-Labs.

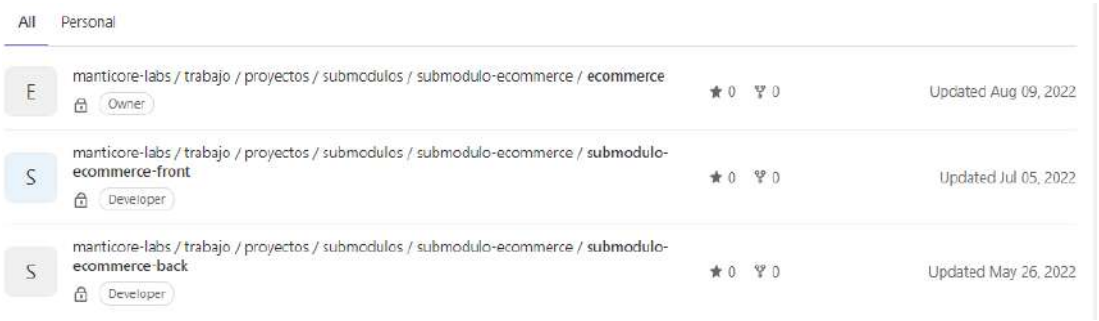


Figura 20: Creación de repositorios

Posteriormente, se creó una nueva rama llamada “ecommerce” (Figura 21) dentro de los proyectos principales de backend y frontend, la cual será la rama principal para el desarrollo del aplicativo para la parte de BackOffice.

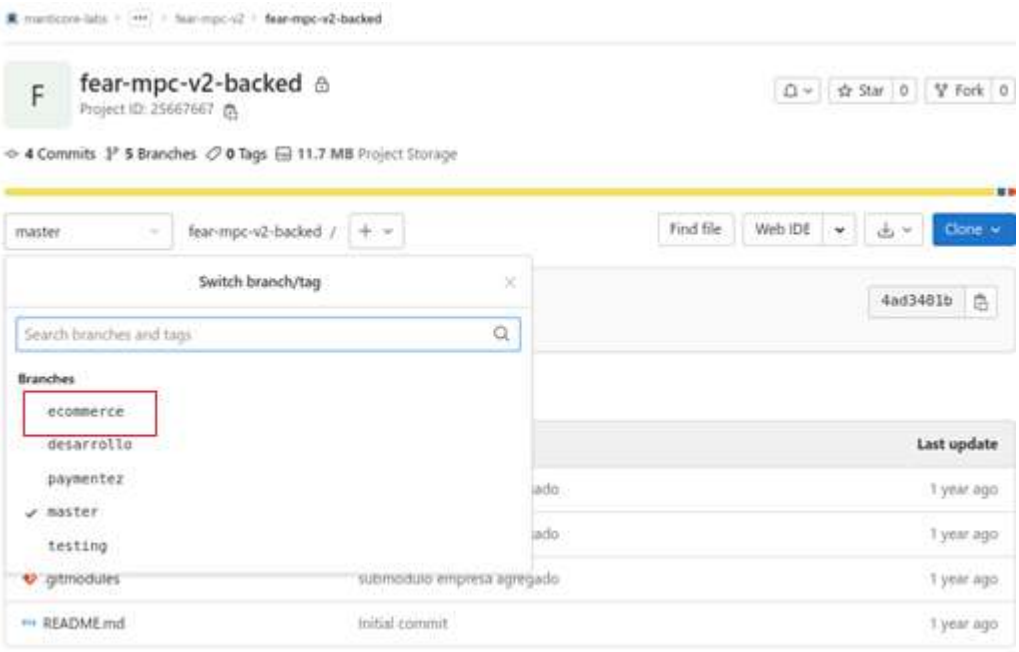


Figura 21: Configuración de ramas

En esta rama, dentro del archivo ".gitmodules" se agregó el nuevo repositorio con el nombre "submodulo-ecommerce", como se muestra en la Figura *22*.

Figura 22: Contenido de archivo ".gitmodules"

Posteriormente en la Figura 23, se muestra el repositorio el que contendrá el código del aplicativo a mostrarse al cliente final, donde se configuró: Tailwind como librería de estilos y PrimeNG con Angular material como librería de componentes, para una mejor experiencia visual del cliente.

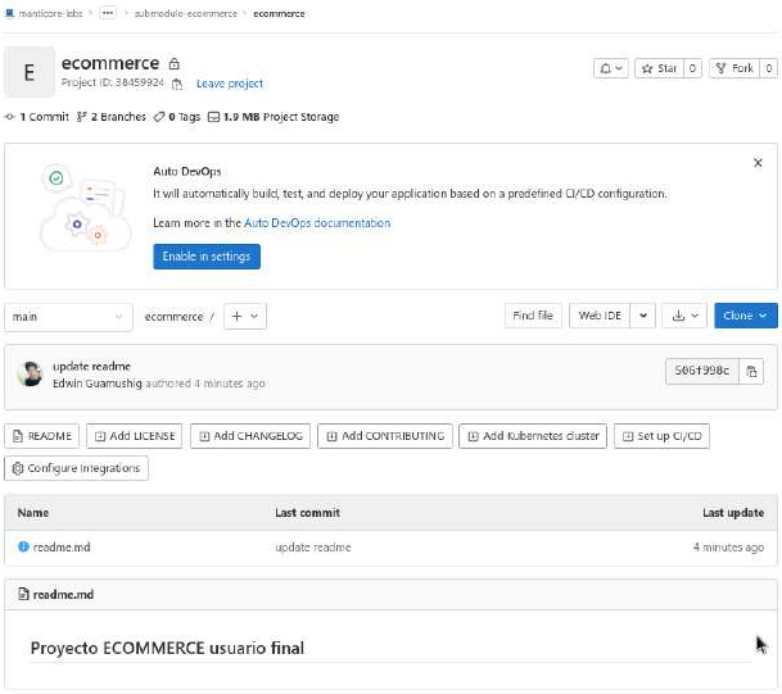


Figura 23: Repositorio aplicación e-commerce frontend

Como última tarea del sprint, se creó un archivo “yml”, sobre el cual se configura la parte de Continious Integration (CI), con el objetivo de tener un código legible y ordenado, utilizando ESLint y runners de GitLab, la configuración se lo puede observar en la Figura *24* y la ejecución en los pipelines de GitLab en la Figura *25*.

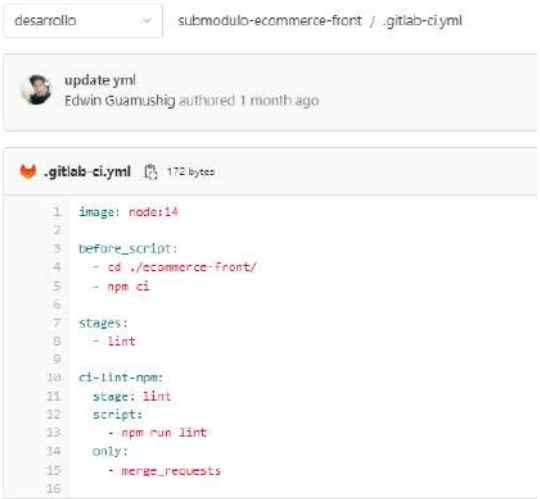


Figura 24: Configuración de CI

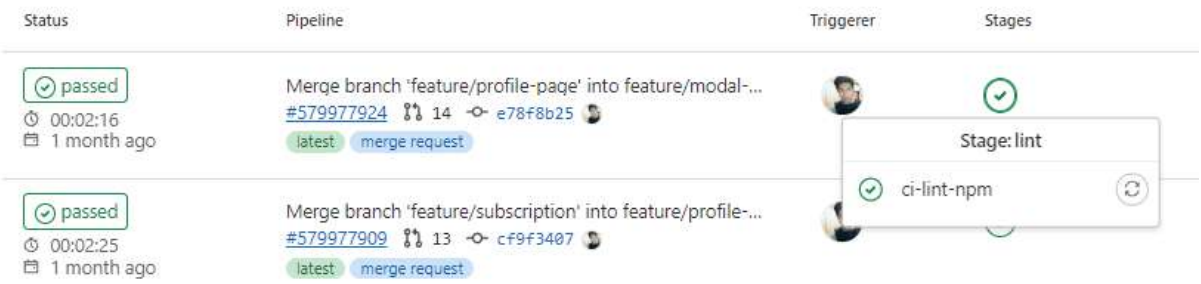


Figura 25: Pipelines CI en GitLab

2.8.3. Sprint Review

Para el sprint inicial, se cumplieron los objetivos sin mayor inconveniente, ya que se logró levantar un ambiente de desarrollo totalmente estable para iniciar con la implementación del aplicativo en futuros sprints.

Además, se tuvo una comunicación fluida con el PO, lo que nos facilitó integrar los nuevos submódulos a la aplicación principal.

A continuación, en la Tabla 7 se muestra el cumplimiento de las historias de usuario implementadas.

Tabla 7: Historias de usuario Sprint 0

Product Backlog				
Código	**Nombre**	**Peso**	**Prioridad**	**Cumplido**
EC-01	Seteo tecnológico	13	Alta	Si
EC-02	Configuración CI	2	Alta	Si

2.8.4. Sprint Retrospective

El sprint tuvo una duración de dos semanas y se llegó a cumplir con los objetivos planteados al inicio de este, además de obtener las siguientes conclusiones:

¿Qué estuvo bien?

- Se crearon los nuevos repositorios que serán parte del proyecto principal de Manticore-Labs como submódulos.
- Se dejó un ambiente estable para iniciar el desarrollo del aplicativo.
- Se configuró la parte de CI, con la finalidad de tener un código más limpio y ordenado.
- Se tiene prototipos de fidelidad media para la parte e-commerce del cliente.

¿Qué estuvo mal?

- Existieron errores en los tokens para el uso de la librería interna de Manticore-Labs.

¿Qué se debe empezar a hacer?

- Junto al PO, pensar en el alcance de cada tarea antes de empezarla.

¿Qué se debe parar de hacer?

- NA

2.9. **Sprint 1**

Objetivos:

- Implementar los catálogos en el BackOffice, para gestionar la información respecto a productos dentro del e-commerce.
- Integrarse con los submódulos de empresa, articulo y artículo empresa, existentes en la aplicación de Manticore-Labs.
- Iniciar con el maquetado del aplicativo e-commerce.

2.9.1. Sprint Planning

En este sprint se trabajará en los catálogos referente a artículos, junto con su integración al nuevo submódulo, estas tareas incluyen:

- *Artículo*: catálogo de etiquetas, etiquetas por artículo.
- *Artículo empresa*: catálogo de precios, impuestos e imágenes.

En la Tabla 8, se muestra las historias de usuario sobre las cuales se trabajará durante el sprint, el detalle completo de cada una se encuentra en el Anexo 6.6.

Tabla 8: Backlog sprint 1

Product Backlog			
Código	**Nombre**	**Peso**	**Prioridad**
EC-27	Gestión de etiquetas	3	Alta
EC-03	Gestión de etiquetas por artículo	5	Media
EC-04	Gestión de artículos por empresa	5	Alta
EC-05	Gestión de imágenes por artículo empresa	13	Baja
EC-06	Gestión de impuestos por artículo empresa	3	Media

EC-07	Gestión de precios por artículo empresa	3	Media

A continuación, en la Tabla 9, se muestran las subtareas detalladas de manera técnica en las que se dividió cada una de las historias de usuario.

Tabla 9: Resumen de subtareas por historia de usuario, Sprint 1

Product Backlog		
HU	**Código**	**Descripción**
EC-27	EC-27.1	Sincronizar base de datos y crear la entidad en la base de datos.
	EC-27.2	Crear servicio REST, controlador y dto's en el backend.
	EC-27.3	Crear la ruta "etiquetas" en el frontend.
	EC-27.4	Crear formulario y modal para crear/editar.
	EC-27.5	Crear filtros de búsqueda por nombre y estado.
	EC-27.6	Crear método para habilitar/deshabilitar un registro.
EC-03	EC-03.1	Crear entidad en la base de datos.
	EC-03.2	Crear servicio REST, controlador y dto's en el backend.
	EC-03.3	Crear la ruta "etiquetas por artículo" en el frontend.
	EC-03.4	Crear modal y tabla con etiquetas disponibles.
	EC-03.5	Crear filtros de búsqueda por nombre y estado.
	EC-03.6	Crear método para habilitar/deshabilitar un registro.
EC-04	EC-04.1	Crear entidad en la base de datos.
	EC-04.2	Crear servicio REST, controlador y dto's en el backend.
	EC-04.3	Crear la ruta "artículos por empresa" en el frontend.
	EC-04.4	Crear modal y formulario con autocomplete de artículos.
	EC-04.5	Crear filtros de búsqueda por nombre.
	EC-04.6	Crear método para habilitar/deshabilitar un registro.
EC-05	EC-05.1	Crear entidad en la base de datos.
	EC-05.2	Crear servicio REST, controlador y dto's en el backend.
	EC-05.3	Crear la ruta "imágenes por artículo empresa" en el frontend.
	EC-05.4	Crear modal y formulario custom para cargar imágenes.
	EC-05.5	Crear método para definir una imagen como principal.
	EC-05.6	Crear método para habilitar/deshabilitar un registro.

	EC-05.7	Crear servicio en el back para subir imágenes a un servidor en la nube.
EC-06	EC-06.1	Crear entidad en la base de datos.
	EC-06.2	Crear servicio REST, controlador y dto's en el backend.
	EC-06.3	Crear la ruta "impuesto para artículo empresa" en el frontend.
	EC-06.4	Crear modal y formulario.
	EC-06.5	Crear filtros de búsqueda por nombre.
	EC-06.6	Crear método para habilitar/deshabilitar un registro.
EC-07	EC-07.1	Crear entidad en la base de datos.
	EC-07.2	Crear servicio REST, controlador y dto's en el backend.
	EC-07.3	Crear la ruta "precio para artículo empresa" en el frontend.
	EC-07.4	Crear modal y formulario.
	EC-07.5	Crear filtros de búsqueda por nombre.
	EC-07.6	Crear método para habilitar/deshabilitar un registro.
	EC-07.7	Crear método para definir un único precio como principal.

2.9.2. Implementación

Para iniciar la integración del nuevo submódulo y el desarrollo de los catálogos, en primer lugar, se levantó las tres bases utilizadas por el sistema (MySQL, Redis, MongoDB) utilizando la herramienta Docker junto con Docker Compose, el cual es una nos permitirá orquestar los contenedores de manera local, como se muestra en la Figura 26.

```
~ » docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
4a79c9baf507   redis:latest   "docker-entrypoint.s…"   2 months ago   Up 52 minutes
r
07f8abc1739c   mongo:latest   "docker-entrypoint.s…"   2 months ago   Up 52 minutes
r
a6d05314570d   mysql:5.7      "docker-entrypoint.s…"   2 months ago   Up 52 minutes
```

Figura 26: Contenedores de bases de datos

En la Figura 27, se muestra la ruta para gestionar los catálogos de “artículos por empresa” (EC-04). Para crear esta pantalla se realizó lo siguiente:

- A nivel de backend dentro del submódulo articulo empresa se creó un nuevo módulo llamado “articulo empresa”, el cual será el encargado de gestionar los artículos por empresa de manera independiente. En la Figura 28 se muestra la estructura del módulo.

- También a nivel de base de datos se creó la nueva entidad “articulo empresa”, con sus respectivas relaciones y tipos de datos.

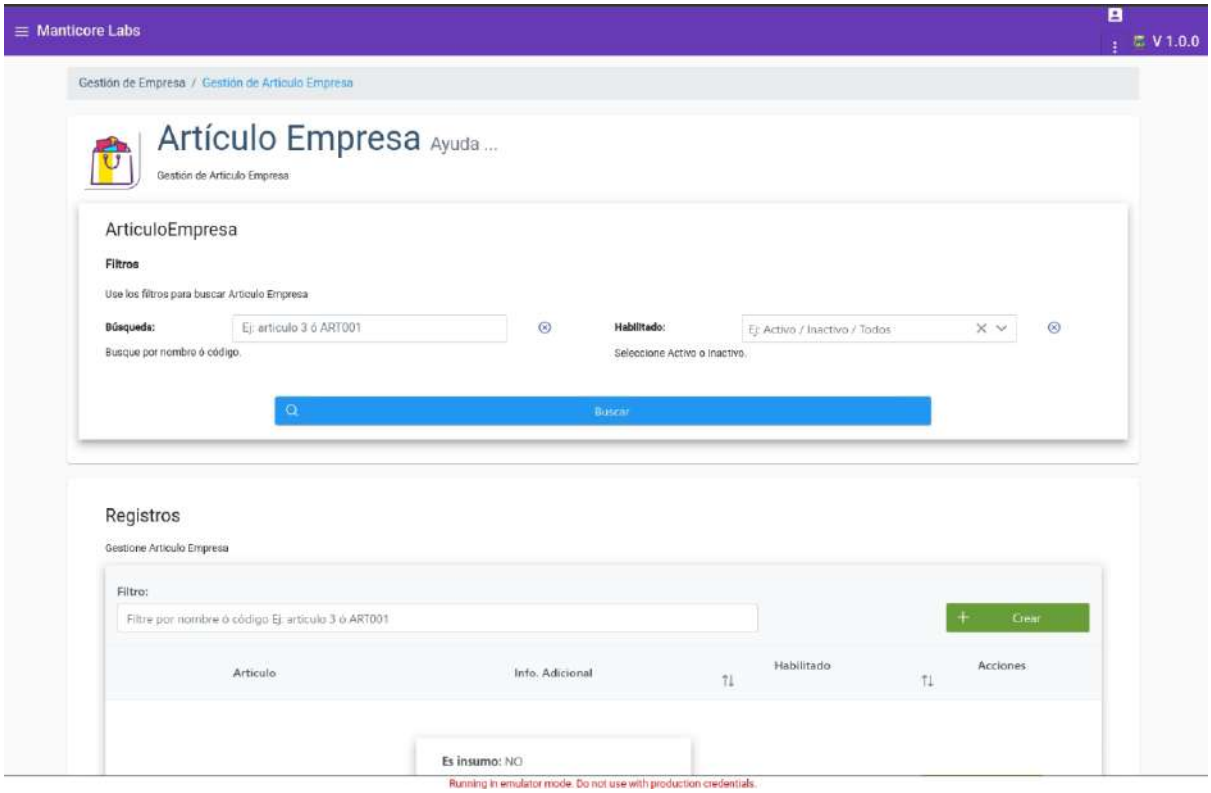


Figura 27: Ruta articulo empresa

Figura 28: Estructura de archivos en submódulo artículo empresa

Adicionalmente, junto a la ruta se creó el formulario para la creación y edición de un registro. El formulario consta de 4 campos, en el que cada campo cuenta con sus respectivas validaciones, tal como se muestra en la Figura *29*.

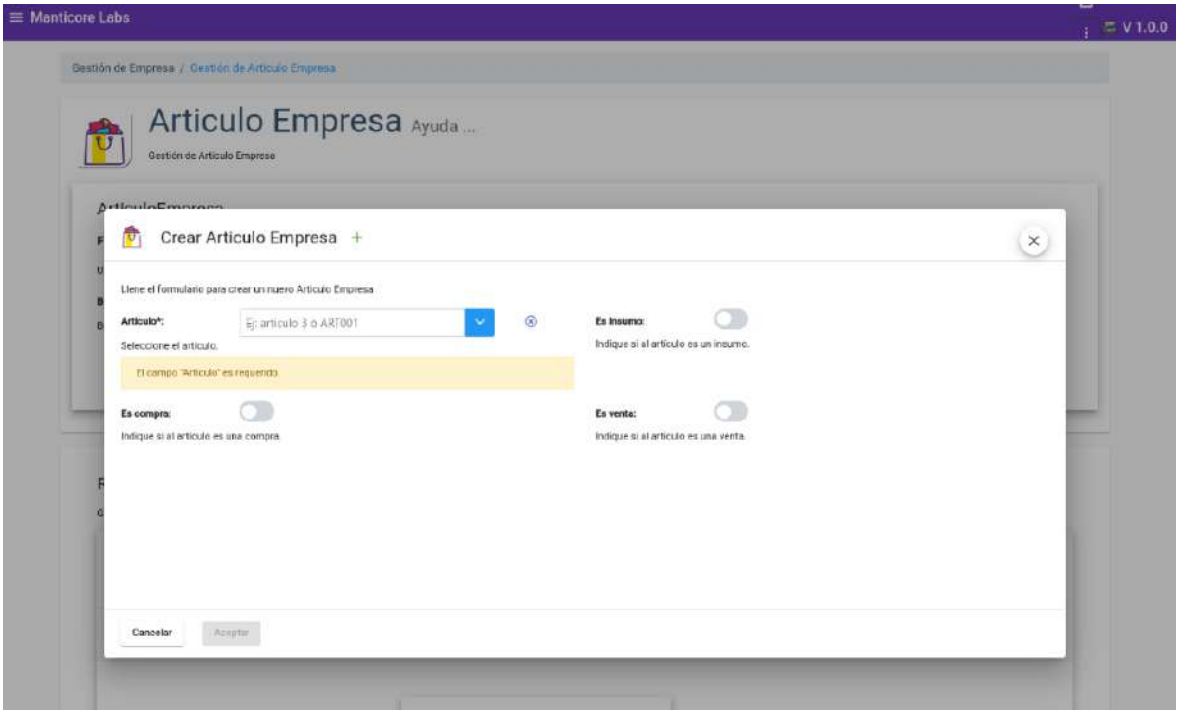


Figura 29: Formulario para creación/edición de articulo empresa

2.9.3. Sprint Review

Al finalizar el sprint 1, se procedió a revisar el estado de cumplimiento de cada una de las historias de usuario que se encuentran en el backlog, con el fin de obtener retroalimentación por parte del PO y obtener mejoras para implementaciones de los siguientes sprints. Los resultados obtenidos se encuentran en la Tabla 10:

Tabla 10: Resultados del sprint 1

Resultados Sprint 1				
HU	**Código**	**Descripción**	**Cumplimiento**	**Observación**
EC-27	EC-27.1	Sincronizar base de datos y crear la entidad en la base de datos.	SI	NA
	EC-27.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-27.3	Crear la ruta "etiquetas" en el frontend.	SI	NA
	EC-27.4	Crear formulario y modal para crear/editar.	SI	NA
	EC-27.5	Crear filtros de búsqueda por nombre y estado.	SI	NA

	EC-27.6	Crear método para habilitar/deshabilitar un registro.	SI	NA
EC-03	EC-03.1	Crear entidad en la base de datos.	SI	NA
	EC-03.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-03.3	Crear la ruta "etiquetas por artículo" en el frontend.	SI	NA
	EC-03.4	Crear modal y tabla con etiquetas disponibles.	SI	NA
	EC-03.5	Crear filtros de búsqueda por nombre y estado.	SI	NA
	EC-03.6	Crear método para habilitar/deshabilitar un registro.	SI	NA
EC-04	EC-04.1	Crear entidad en la base de datos.	SI	NA
	EC-04.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-04.3	Crear la ruta "artículos por empresa" en el frontend.	SI	NA
	EC-04.4	Crear modal y formulario con autocomplete de artículos.	SI	NA
	EC-04.5	Crear filtros de búsqueda por nombre.	SI	NA
	EC-04.6	Crear método para habilitar/deshabilitar un registro.	SI	NA
EC-05	EC-05.1	Crear entidad en la base de datos.	SI	NA
	EC-05.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-05.3	Crear la ruta "imágenes por artículo empresa" en el frontend.	SI	NA
	EC-05.4	Crear modal y formulario custom para cargar imágenes	NO	Debido a que aún no se define que servicio en la nube se usará para guardar las imágenes, no se pudo avanzar con el formulario de carga.

	EC-05.5	Crear método para definir una imagen como principal.	Parcialmente	Al no tener donde cargar las imágenes, aún no se tienen registros de estas.
	EC-05.6	Crear método para habilitar/deshabilitar un registro.	Parcialmente	Al no tener donde cargar las imágenes, aún no se tienen registros de estas.
	EC-05.7	Crear servicio en el back para subir imágenes a un servidor en la nube.	NO	Junto al PO aún no se ha decidido donde subir las imágenes, por lo cual esto se realizará en el último sprint, ya que las imágenes no son algo indispensable por el momento.
EC-06	EC-06.1	Crear entidad en la base de datos.	SI	NA
	EC-06.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-06.3	Crear la ruta "impuesto para artículo empresa" en el frontend.	SI	NA
	EC-06.4	Crear modal y formulario.	SI	NA
	EC-06.5	Crear filtros de búsqueda por nombre.	SI	NA
	EC-06.6	Crear método para habilitar/deshabilitar un registro.	SI	NA
EC-07	EC-07.1	Crear entidad en la base de datos.	SI	NA
	EC-07.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-07.3	Crear la ruta "precio para artículo empresa" en el frontend.	SI	NA
	EC-07.4	Crear modal y formulario.	SI	NA
	EC-07.5	Crear filtros de búsqueda por nombre.	SI	NA
	EC-07.6	Crear método para habilitar/deshabilitar un registro.	SI	NA
	EC-07.7	Crear método para definir un único precio como principal.	SI	NA

2.9.4. Sprint Retrospective

El sprint tuvo una duración de dos semanas, se llegó a cumplir parcialmente con los objetivos planteados, ya que existió problemas para definir donde se almacenarán las imágenes de cada uno de los productos.

¿Qué estuvo bien?

- Se logró cumplir con los principales objetivos planteados durante el sprint planning.
- El nuevo submódulo e-commerce se pudo integrar sin mayor problema con los otros submódulos del sistema (artículo empresa).
- Se utilizó Docker para levantar contenedores que nos permita manejar las bases de datos que usa el sistema de Manticore-Labs.

¿Qué estuvo mal?

- No se definió donde se almacenará las imágenes de cada uno de los productos.
- La historia EC-05 se realizó de manera parcial.

¿Qué se debe empezar a hacer?

- Implementar un linteo de código para mantener el proyecto lo más legible posible y además evitar el uso de console.log en el producto final.

¿Qué se debe parar de hacer?

- NA

Como se muestra en el gráfico burndown de la Figura *30*, no se logró completar la totalidad de los puntos previstos para el sprint, debido a los problemas comentados previamente sobre la carga de imágenes para los productos.

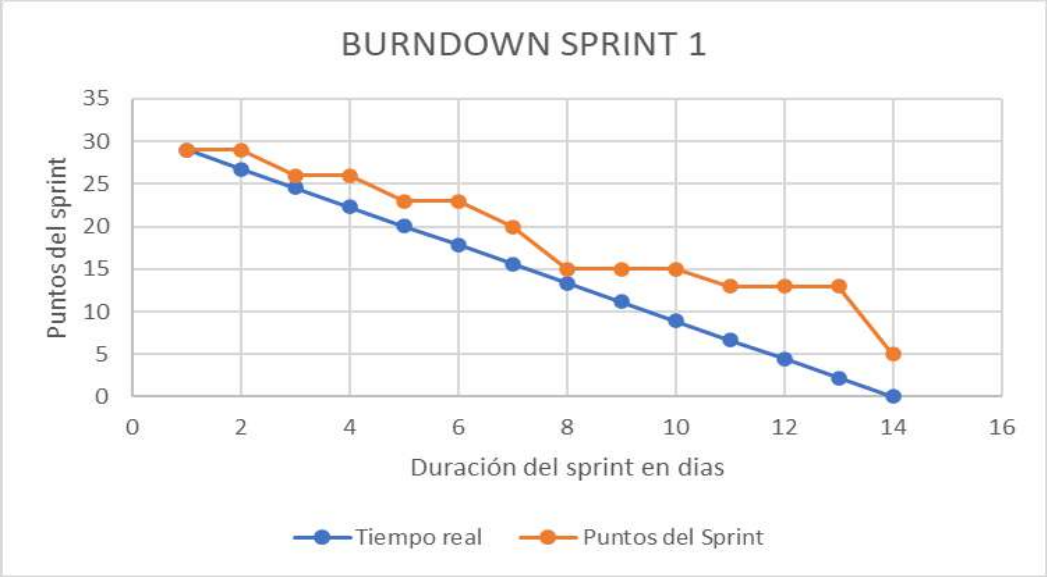


Figura 30: Burndown sprint 1

2.10. **Sprint 2**

Objetivos:

- Implementar en el BackOffice, rutas y catálogos para gestionar la información referente a compra y suscripción de productos.
- Configurar una cuenta en SendGrid para el envío de correos.
- Crear una nueva ruta para listar las reseñas de cada producto perteneciente a una empresa.

2.10.1. Sprint Planning

En este sprint se trabajará en los listados referentes a pagos, productos vendidos y reseñas, además de la gestión de clientes y suscripciones, y como paso final se configurará una cuenta en SendGrid para las notificaciones por medio de correo electrónico.

En la Tabla 11, se muestra las historias de usuario sobre las cuales se trabajará durante el sprint, el detalle completo de cada una se encuentra en el Anexo 6.6.

Tabla 11: Backlog sprint 2

Product Backlog			
Código	**Nombre**	**Peso**	**Prioridad**
EC-08	Listado de pagos	3	Baja
EC-09	Gestión de suscripciones	8	Media

EC-10	Gestión de clientes	8	Media
EC-12	Listado reseñas por artículo empresa	3	Baja
EC-32	Configuración de cuenta para envío de correos	3	Media
EC-30	Listado de productos vendidos	3	Baja

Para cada una de las historias de usuario que se encuentran en el backlog, se determinó un listado de tareas técnicas a realizar, las cuales se muestran de manera detallada en la Tabla 12.

Tabla 12: Resumen de tareas técnicas para sprint 2

Product Backlog		
HU	**Código**	**Descripción**
EC-08	EC-08.1	Sincronizar base de datos y crear la entidad en la base de datos.
	EC-08.2	Crear servicio REST, controlador y dto's en el backend.
	EC-08.3	Crear la ruta "pagos" en el frontend.
	EC-08.4	Crear formulario y modal para visualizar información.
	EC-08.5	Crear filtros de búsqueda por nombre y estado.
EC-30	EC-30.1	Crear servicio REST, controlador y dto's en el backend.
	EC-30.2	Sincronizar base de datos y crear la entidad en la base de datos.
	EC-30.3	Crear la ruta "productos vendidos" en el frontend.
	EC-30.4	Crear formulario y modal para visualizar información.
	EC-30.5	Crear filtros de búsqueda por nombre y estado.
EC-09	EC-09.1	Sincronizar base de datos y crear la entidad en la base de datos.
	EC-09.2	Crear servicio REST, controlador y dto's en el backend.
	EC-09.3	Crear la ruta "suscripciones" en el frontend.
	EC-09.4	Crear formulario y modal para visualizar información.
	EC-09.5	Crear filtros de búsqueda por nombre y estado.
	EC-09.6	Agregar opción para cambiar estado de suscripción.
EC-10	EC-10.1	Sincronizar base de datos y crear la entidad en la base de datos.
	EC-10.2	Crear servicio REST, controlador y dto's en el backend.
	EC-10.3	Crear la ruta "clientes" en el frontend.
	EC-10.4	Crear formulario y modal para visualizar información.
	EC-10.5	Crear filtros de búsqueda por nombre, documento y estado.
	EC-10.6	Agregar opción para dar de baja a un cliente.

	EC-10.7	Crear formulario para recuperar contraseña de cliente.
EC-12	EC-12.1	Crear servicio REST, controlador y dto's en el backend.
	EC-12.2	Sincronizar base de datos y crear la entidad en la base de datos.
	EC-12.3	Crear la ruta "reseña de producto" en el frontend.
	EC-12.4	Crear filtros de búsqueda por rating.
EC-32	EC-32.1	Crear una cuenta en SendGrid con credenciales propias.
	EC-32.2	Leer la documentación y entender la integración con sistemas NodeJS.

2.10.2. Implementación

Durante el sprint se continuo con la implementación de los catálogos para el BackOffice de la aplicación e-commerce.

- En el backend, dentro del submódulo e-commerce se creó los módulos “pago”, “suscripcion”, “cliente”, “reseña” y “venta”, cada uno con su respectivo servicio, controlador y DTO.
- Para cada módulo descrito en el anterior ítem, se creó una entidad en la base de datos, con sus respectivos campos y validaciones.
- En el frontend, se creó la ruta para los módulos “pago”, “suscripcion”, “cliente”, “reseña” y “venta”.
- Se creó los respectivos formularios para mostrar, crear y editar la información de los módulos mencionados en el ítem anterior.

Se inició con la implementación del módulo “pago”, tanto a nivel de frontend como de backend. Este módulo nos permitirá visualizar un listado de los pagos realizados en nuestra plataforma e-commerce; en la Figura 31 se puede observar la ruta pagos con sus respectivos filtros.

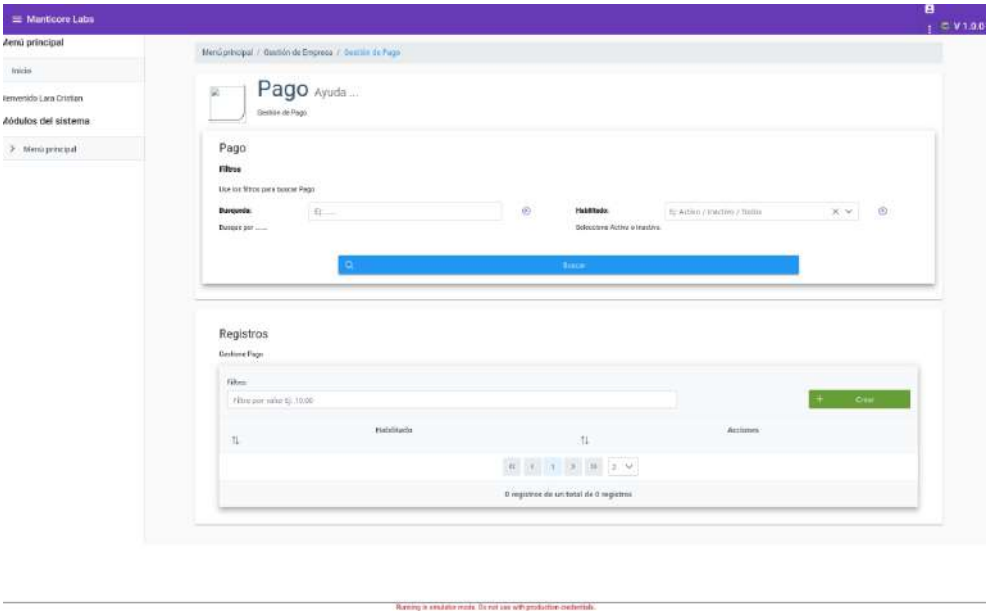


Figura 31: Ruta de pagos

Debido a que un requisito es poder observar y actualizar la información de un cliente, se implementó unca nueva ruta para el módulo “cliente” (Figura *32*), con su respectivo formulario dándole la opción al administrador de poder actualizar de manera directa la información errónea del cliente y recuperar su contraseña en caso de ser necesario.

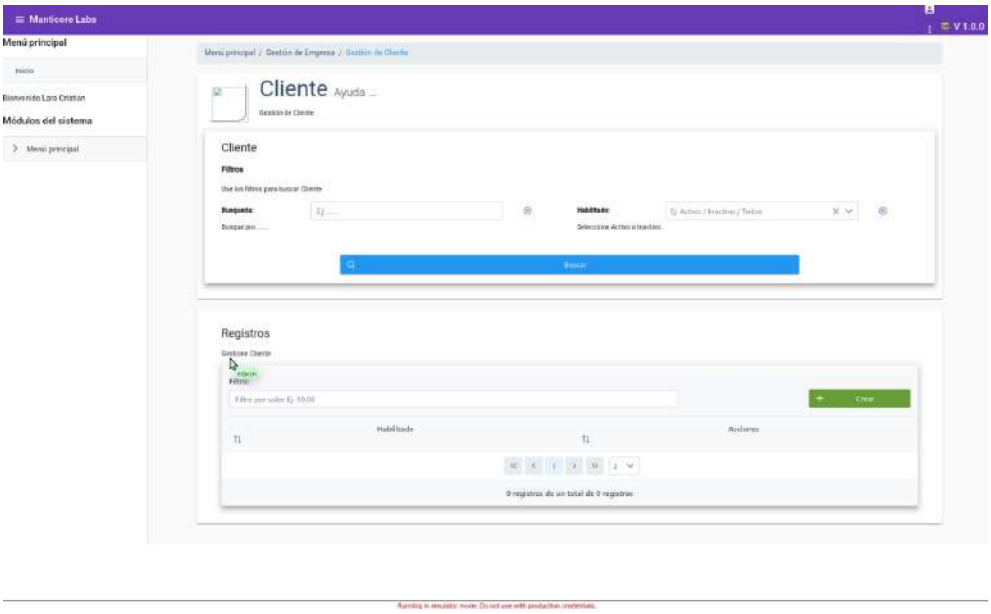


Figura 32: Ruta cliente

Finalmente, en la Figura 33 se muestra el listado de suscripciones que tiene la tienda e-commerce de la empresa, desde esta pantalla el administrador podrá cancelar una suscripción si el cliente así lo requiere.

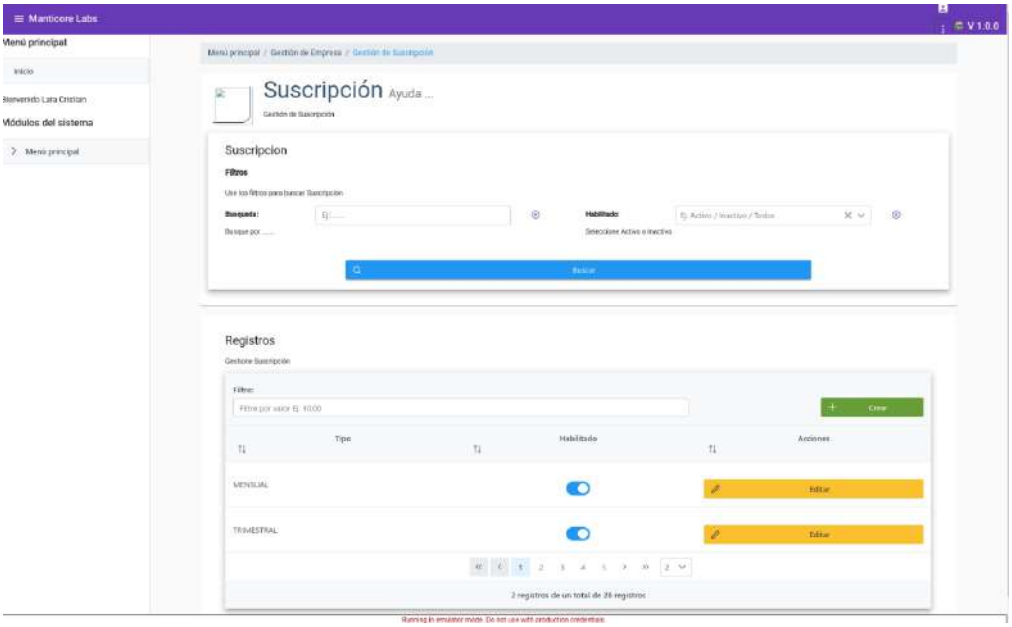


Figura 33: Ruta suscripción

2.10.3. Sprint Review

Tras finalizar el sprint 2, se realizó una revisión del estado de cada una de las tareas que se encontraban en el backlog, con el fin de obtener información y sacar conclusiones del avance del sprint. Los resultados se pueden observar en la Tabla 13:

Tabla 13:Resultados del sprint 2

Sprint Review				
HU	**Código**	**Descripción**	**Cumplimiento**	**Observación**
EC-08	EC-08.1	Sincronizar base de datos y crear la entidad en la base de datos.	SI	NA
	EC-08.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-08.3	Crear la ruta "pagos" en el frontend.	SI	NA
	EC-08.4	Crear formulario y modal para visualizar información.	SI	NA
	EC-08.5	Crear filtros de búsqueda por nombre y estado.	SI	NA
EC-30	EC-30.1	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-30.2	Sincronizar base de datos y crear la entidad en la base de datos.	SI	NA

	EC-30.3	Crear la ruta "productos vendidos" en el frontend.	SI	Los productos vendidos en un inicio estaban planificados para ser ruta hija de empresa. Pero para facilidad de uso y entendimiento se decidió que sea ruta hija de cliente.
	EC-30.4	Crear formulario y modal para visualizar información.	SI	NA
	EC-30.5	Crear filtros de búsqueda por nombre y estado.	SI	NA
EC-09	EC-09.1	Sincronizar base de datos y crear la entidad en la base de datos.	SI	NA
	EC-09.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-09.3	Crear la ruta "suscripciones" en el frontend.	SI	NA
	EC-09.4	Crear formulario y modal para visualizar información.	SI	NA
	EC-09.5	Crear filtros de búsqueda por nombre y estado.	SI	NA
	EC-09.6	Agregar opción para cambiar estado de suscripción.	SI	Se puede cambiar únicamente entre dos estados (activo, deshabilitado).
EC-10	EC-10.1	Sincronizar base de datos y crear la entidad en la base de datos.	SI	NA
	EC-10.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-10.3	Crear la ruta "clientes" en el frontend.	SI	NA
	EC-10.4	Crear formulario y modal para visualizar información.	SI	NA
	EC-10.5	Crear filtros de búsqueda por nombre, documento y estado.	SI	NA
	EC-10.6	Agregar opción para dar de baja a un cliente.	SI	NA
	EC-10.7	Crear formulario para recuperar contraseña de cliente.	SI	NA
EC-12	EC-12.1	Crear servicio REST, controlador y dto's en el backend.	SI	NA

	EC-12.2	Sincronizar base de datos y crear la entidad en la base de datos.	SI	NA
	EC-12.3	Crear la ruta "reseña de producto" en el frontend.	SI	NA
	EC-12.4	Crear filtros de búsqueda por rating.	SI	NA
EC-32	EC-32.1	Crear una cuenta en SendGrid con credenciales propias.	SI	Se creó una cuenta en SendGrid, ya que este servicio nos permitirá trackear el estado de cada correo enviado.
	EC-32.2	Leer la documentación y entender la integración con sistemas NodeJs.	SI	NA

2.10.4. Sprint Retrospective

En este sprint se logró cumplir con todos los objetivos planteados al inicio de este.

¿Qué estuvo bien?

- Se logró realizar todas las tareas del backlog en el tiempo esperado.
- Debido a que en el sprint 1 se adquirió experiencia trabajando con la librería de Manticore-Labs, en este sprint se pudo realizar las tareas de una manera más ágil.
- Se tiene configurado el servicio de SendGrid para el envío de los correos.

¿Qué estuvo mal?

- Se utilizó un correo personal al activar la cuenta de SendGrid.

¿Qué se debe empezar a hacer?

- Pedir a Manticore-Labs correos para realizar configuraciones en posibles servicios externos a usar.

¿Qué se debe parar de hacer?

- Tener reuniones demasiado largas con el PO.

El esfuerzo realizado durante el transcurso del sprint, se lo puede observar en el gráfico burndown de la Figura 34. En esta gráfica se puede ver que las tareas se realizaron de manera más ágil, ya que el anterior sprint nos dejó grandes conocimientos sobre el funcionamiento y uso de la librería interna de Manticore-Labs.

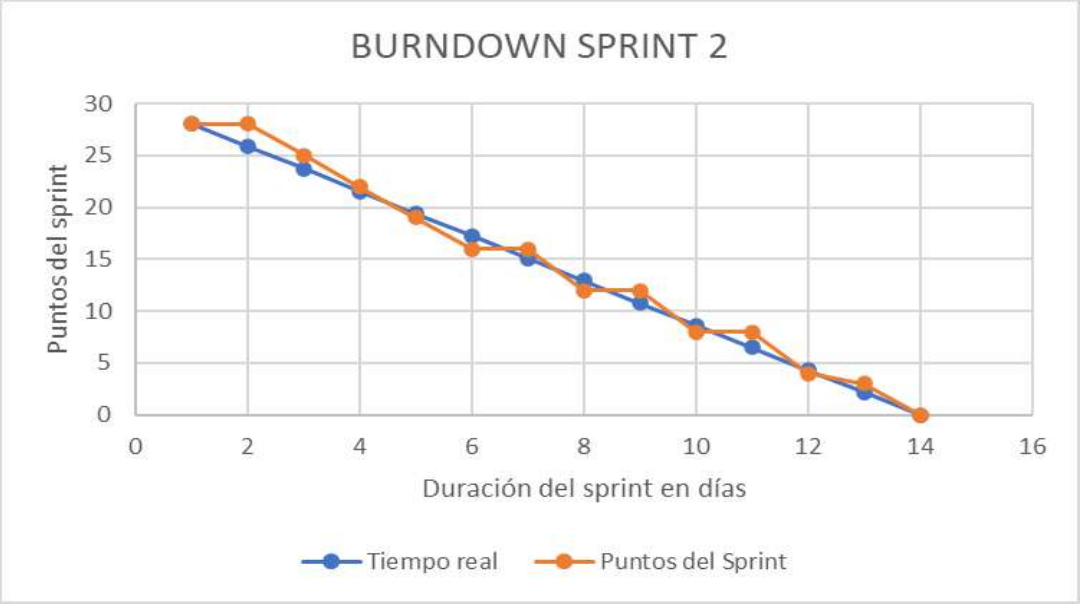


Figura 34: Burndown sprint 2

2.11. **Sprint 3**

Objetivos:

- Crear las rutas y formularios para registro y autenticación de clientes en el e-commerce.
- Maquetar pantalla de inicio del e-commerce utilizando tailwind.
- Conectar la aplicación frontend del cliente con el backend de Manticore-Labs para cargar los productos gestionados por empresa.
- Crear filtros dinámicos en base a los grupos y subgrupos de cada uno de los productos cargados en la pantalla inicial.

2.11.1. Sprint Planning

En este sprint se trabajará en la aplicación e-commerce que visualizará el cliente final y su interacción con el submódulo e-commerce y el resto de los submódulos del backend de Manticore-Labs.

En la Tabla 14, se muestra las historias de usuario sobre las cuales se trabajará durante el sprint, el detalle completo de cada una se encuentra en el Anexo 6.6.

Tabla 14: Backlog sprint 3

Product Backlog			
Código	**Nombre**	**Peso**	**Prioridad**
EC-13	Login de usuario	8	Alta
EC-14	Registro de usuario	8	Alta
EC-16	Filtros por grupo y subgrupo para e-commerce	8	Alta
EC-18	Maquetar página inicial de tienda e-commerce	20	Alta

En la Tabla 15, se muestra las tareas técnicas a realizar para cada una de las historias de usuario que se encuentra en el backlog.

Tabla 15: Resumen de tareas técnicas del sprint 3

Product Backlog		
HU	**Código**	**Descripción**
EC-13	EC-13.1	Crear una estructura escalable para el nuevo proyecto.
	EC-13.2	Crear componentes reutilizables.
	EC-13.3	Sincronizar base de datos y crear la entidad en la base de datos.
	EC-13.4	Crear servicio REST, controlador y dto's en el backend.
	EC-13.5	Crear ruta pública para el Login de usuarios.
	EC-13.6	Crear formulario de Login con sus respectivas validaciones, acorde a los mockups.
EC-14	EC-14.1	Sincronizar base de datos y crear la entidad en la base de datos.
	EC-14.2	Crear servicio REST, controlador y dto's en el backend.
	EC-14.3	Crear ruta pública para registro de usuarios.
	EC-14.4	Crear formulario de Login con sus respectivas validaciones, acorde a los mockups.
EC-18	EC-18.1	Configurar tipografía y estilos base para maquetar pantalla inicial.
	EC-18.2	Crear ruta de inicio de la tienda e-commerce.
	EC-18.3	Diseñar la pantalla de inicio acorde a los mockups.
	EC-18.4	Cargar productos por identificador de empresa.
	EC-18.5	Agregar paginación para listar los productos.
EC-16	EC-16.1	Cargar filtros en base a catálogos de grupos y subgrupos.
	EC-16.2	Crear persistencia de filtros.
	EC-16.3	Conectar filtros avanzados de grupo y subgrupo.

2.11.2. Implementación

Antes de iniciar con la implementación de las pantallas; se decidió, junto al PO, crear dentro del proyecto una estructura de carpetas que permita tener una aplicación escalable. Para lo cual se utilizó la estructura “Folders by feature” (Figura 35) de la guía de estilo angular, ya que esta nos permite desarrollar una aplicación modularizada. Para esta estructura se utiliza los siguientes fundamentos:

- Root: Punto de entrada de la app.
- Services: Responsables de almacenar los servicios globales.
- Shared: Debe contener los componentes que se reutilizaran en otros módulos.
- Module: Debe contener los módulos de la aplicación.

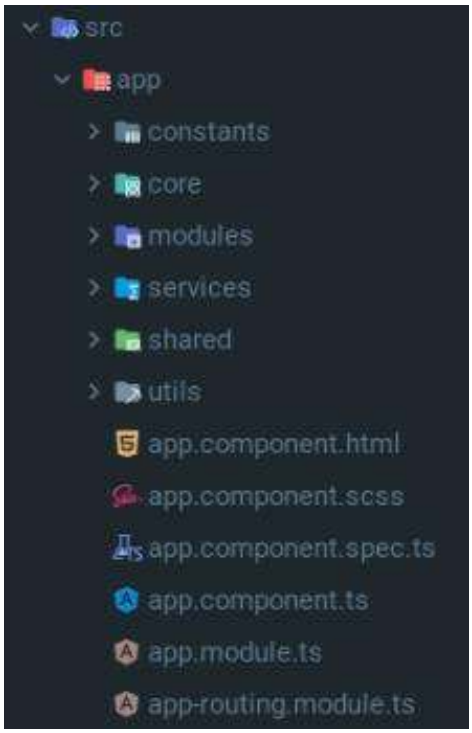


Figura 35: Estructura Folders by feature para carpetas

En la Figura 36, se muestra los componentes creados basándose en Atomic Design, los cuales tienen el objetivo de facilitar los posibles cambios existentes, permitir realizar los prototipos en un tiempo óptimo y sobre todo reutilizarlos en cualquier parte de la aplicación.

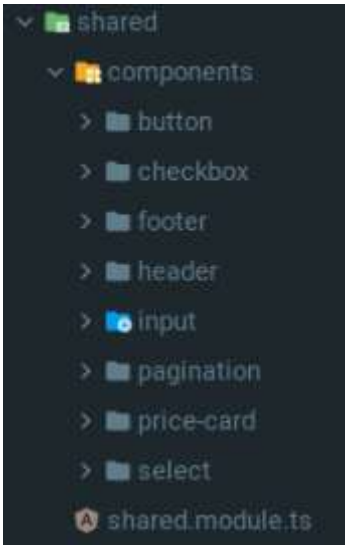


Figura 36: Directorio de componentes

Una vez terminado los pasos anteriores, se procedió a crear la ruta de Login con su respectivo formulario y validaciones, en el cual se utilizó componentes como “input” y “button” creados previamente con estilos ya definidos.

En la Figura 37 y Figura 38, se muestra la ruta y formulario con sus respectivas validaciones para el login y registro de usuarios.

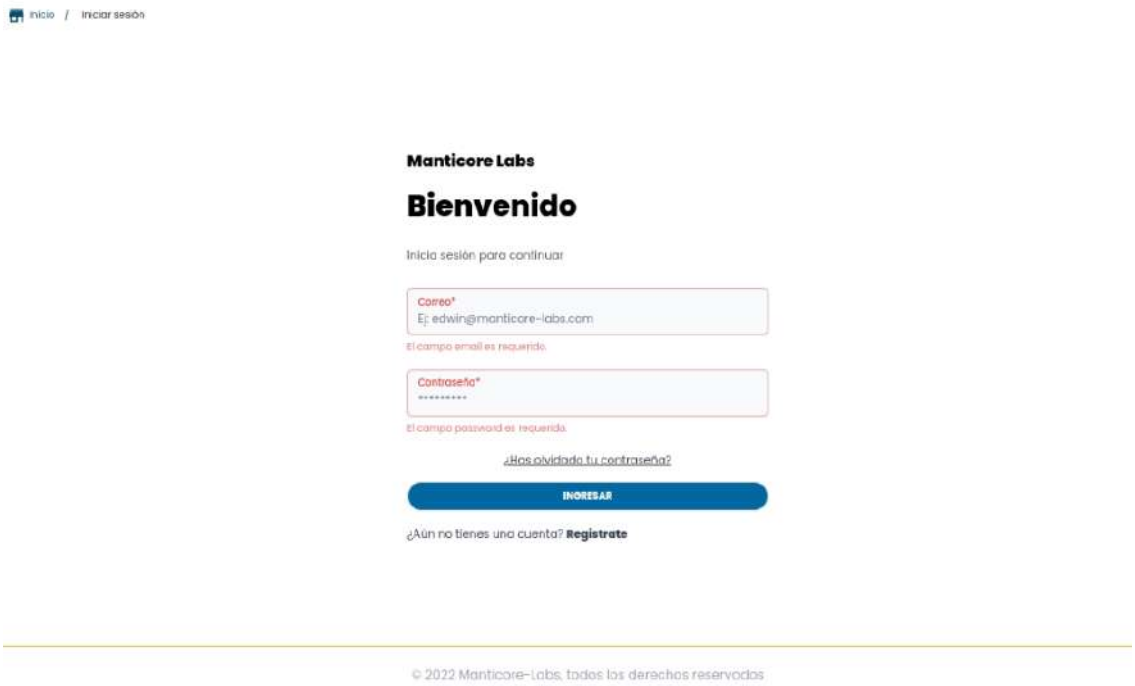


Figura 37: Ruta y formulario de Login

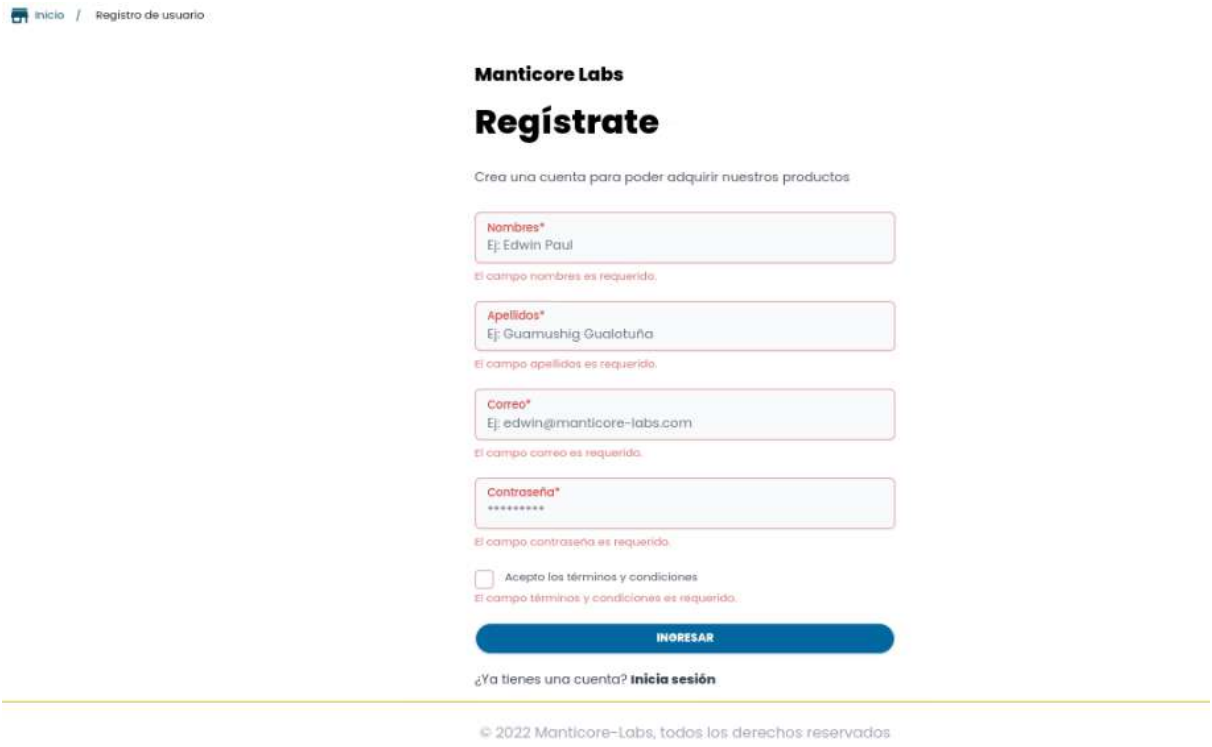


Figura 38: Ruta y formulario de registro

Tras la creación de las dos primeras pantallas del aplicativo, se procedió a crear la pantalla de inicio de la tienda. Para ésto se utilizó Tailwind, ya que este framework al ser altamente personalizable nos permitió realizar un desarrollo ágil, el cual se basó en la utilización de clases dentro del código HTML, permitiendo optimizar y minimizar al máximo el uso de código CSS.

Por ejemplo, en la Figura 39 se muestra el título “Búsqueda” estilizado con Tailwind.

```
<h2 class="text-lg font-semibold text-gray-900">Búsqueda</h2>
```

Figura 39: Clases Tailwind para estilos

En la

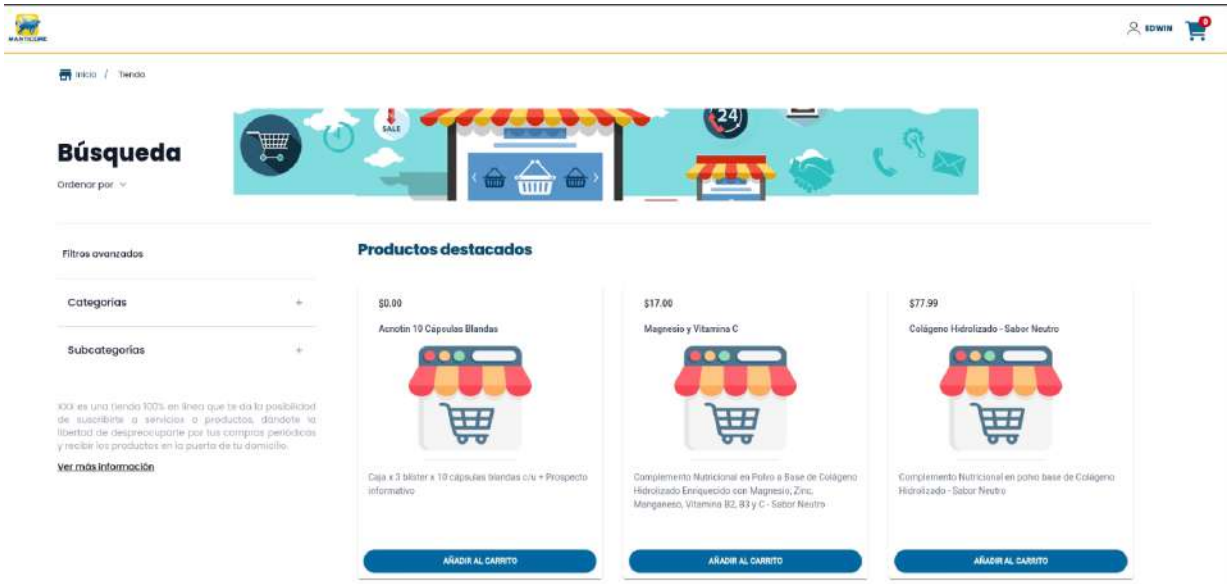


Figura *40*, se muestra el resultado final para la pantalla de inicio donde se puede observar la sección de filtros, productos e información sobre suscripciones. Esta pantalla se implementó lo más similar posible a los mockups, por lo cual Tailwind fue una gran ayuda al momento de maquetar y trabajar con grillas dentro de la pantalla.

Una vez finalizado la maquetación se realizó el servicio para cargar por lazy desde el backend de Manticore-Labs cada uno de los productos pertenecientes a una empresa en concreto.

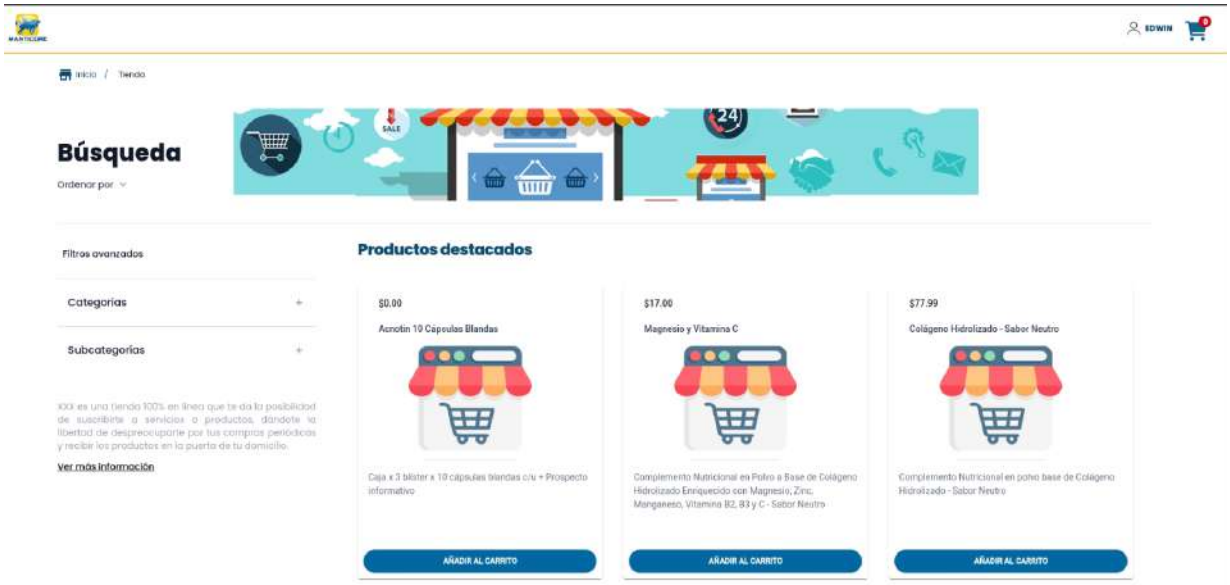


Figura 40: Pantalla inicial aplicativo e-commerce

Para terminar el sprint, se realizó el servicio para la carga dinámica de filtros en base a los grupos y subgrupos a los que pertenece cada producto.

En la Figura 41, se puede observar que los filtros categoría y subcategoría se han cargado en base a los grupos y subgrupos que maneja el sistema de Manticore-Labs, los cuales se pueden visualizar en la Figura 42.

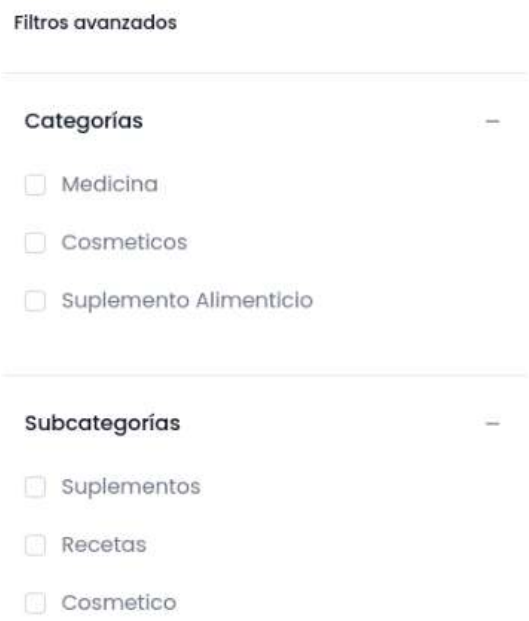


Figura 41: Filtros avanzados

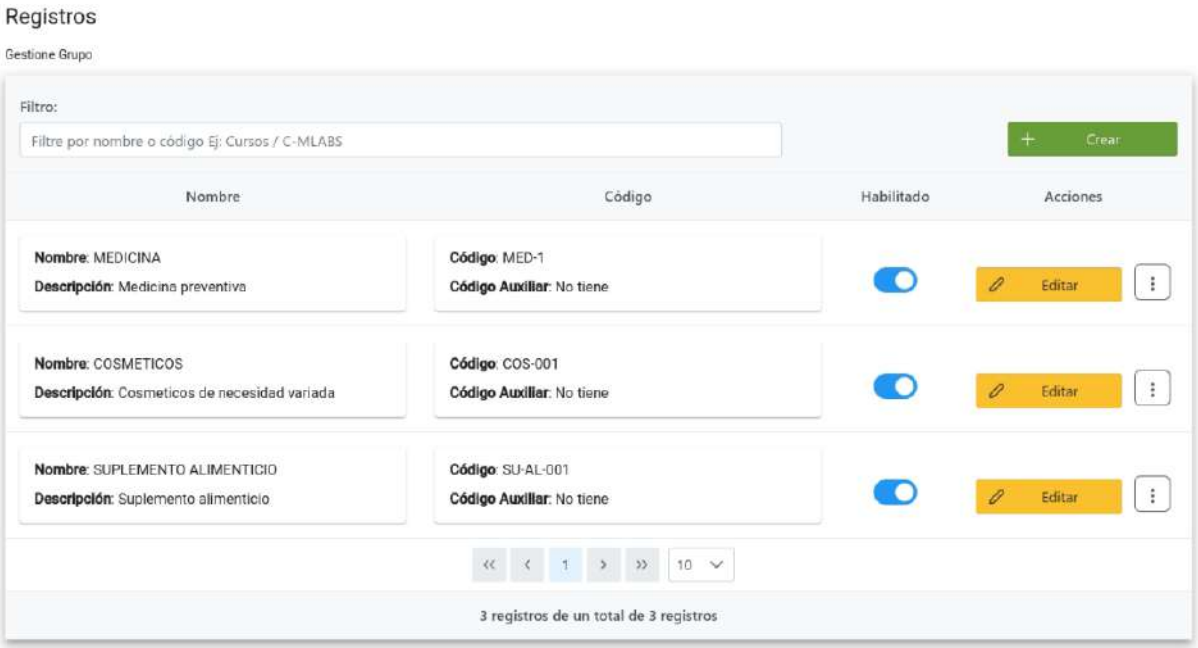


Figura 42: Listado de grupos existentes en el sistema

Finalmente, para mantener los filtros seleccionas por el usuario en caso de un reinicio de la página, se utilizó el servicio “Router” que ofrece angular, ya que éste nos permitió

guardar en las query params de la URL los identificadores de cada grupo y/o subgrupo seleccionado por el usuario de una manera sencilla; la URL final se la puede observar en la imagen a continuación.

Figura 43: URL con identificador de filtros seleccionados

2.11.1. Sprint Review

Tras finalizar el sprint 3, se realizó una revisión del estado de cada una de las tareas que se encontraban en el backlog, con el fin de obtener información y sacar conclusiones del avance del sprint. Los resultados se pueden observar en la Tabla 16:

Tabla 16: Resultados del sprint 3

Sprint Review				
HU	**Código**	**Descripción**	**Cumplimiento**	**Observación**
EC-13	EC-13.1	Crear una estructura escalable para el nuevo proyecto.	SI	NA
	EC-13.2	Crear componentes reutilizables.	SI	Se utilizó atomic design con el fin de tener una aplicación mantenible y escalable.
	EC-13.3	Sincronizar base de datos y crear la entidad en la base de datos.	SI	NA
	EC-13.4	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-13.5	Crear ruta pública para el login de usuarios.	SI	NA
	EC-13.6	Crear formulario de login con sus respectivas validaciones, acorde a los mockups.	SI	NA
EC-14	EC-14.1	Sincronizar base de datos y crear la entidad en la base de datos.	SI	NA

	EC-14.2	Crear servicio REST, controlador y dto's en el backend.	SI	NA
	EC-14.3	Crear ruta pública para registro de usuarios.	SI	NA
	EC-14.4	Crear formulario de login con sus respectivas validaciones, acorde a los mockups.	SI	NA
EC-18	EC-18.1	Configurar tipografía y estilos base para maquetar pantalla inicial.	SI	NA
	EC-18.2	Crear ruta de inicio de la tienda e-commerce.	SI	NA
	EC-18.3	Diseñar la pantalla de inicio acorde a los mockups.	SI	NA
	EC-18.4	Cargar productos por identificador de empresa.	SI	NA
	EC-18.5	Agregar paginación para listar los productos.	SI	NA
EC-16	EC-16.1	Cargar filtros en base a catálogos de grupos y subgrupos.	SI	NA
	EC-16.2	Crear persistencia de filtros.	SI	Se utilizó query params.
	EC-16.3	Conectar filtros avanzados de grupo y subgrupo.	SI	NA

2.11.2. Sprint Retrospective

Se cumplió con el objetivo propuesto al inicio de este sprint.

¿Qué estuvo bien?

- Se crearon componentes que permitirán tener una aplicación escalable y fácil de mantener.
- El uso de Tailwind facilito la implementación y customización de cada uno de los componentes y pantallas.
- La conexión con el backend de Manticore-Labs se la pudo realizar sin mayores inconvenientes.
- Los datos que se muestran por el momento en el e-commerce ya son datos administrados desde el BackOffice.

¿Qué estuvo mal?

- Como existió un mayor esfuerzo en las tareas, durante este sprint se pasó de trabajar de 6 horas a 8 horas para poder cumplir el objetivo.

¿Qué se debe empezar a hacer?

- NA

¿Qué se debe parar de hacer?

- NA.

El esfuerzo tomado para cumplir el objetivo del sprint se lo puede observar en el gráfico burndown de la Figura 44. En esta gráfica se puede ver que la tarea de creación de componentes tomó algo de tiempo más de lo previsto, pero se pudo compensar en las tareas respecto a formularios de login y registro, ya que los componentes facilitaron la implementación de éstas.

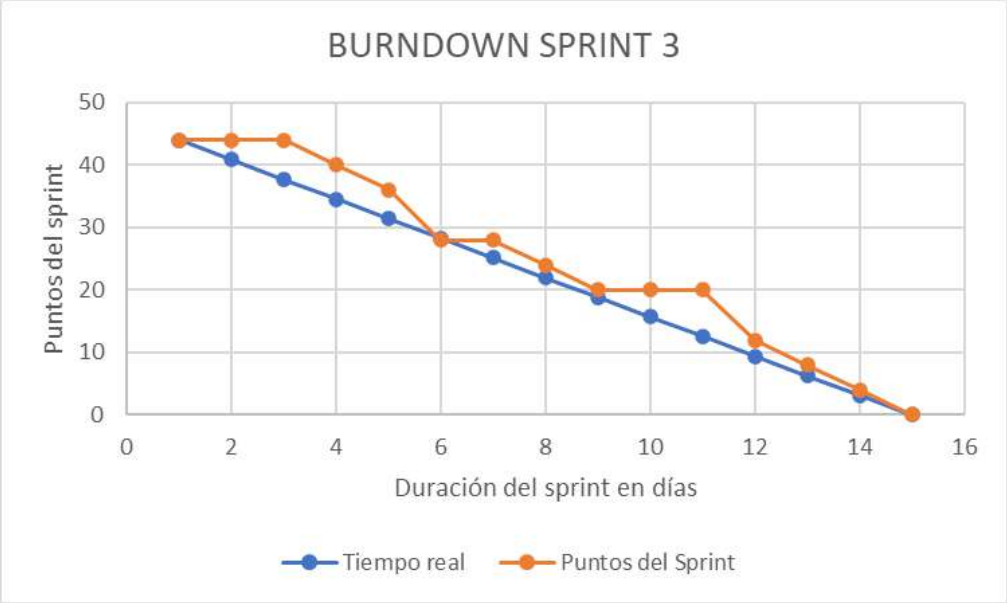


Figura 44: Burndown sprint 3

2.12. **Sprint 4**

Objetivos:

- Crear ruta con información completa e imágenes del producto seleccionado por el usuario.

- Crear el modal para agregar una compra de tipo suscripción.
- Crear la ruta para mostrar al cliente el detalle de su compra, junto al precio final con el desglose de los impuestos.
- Agregar la opción para que el cliente pueda dejar su reseña respecto a algún producto.

2.12.1. Sprint Planning

En este sprint se continuará trabajando en la aplicación e-commerce que visualizará el cliente final, concretamente en la visualización de detalle de producto y compra.

En la Tabla 17, se muestra las tareas sobre las que se trabajará este sprint.

Tabla 17: Backlog sprint 4

Product Backlog			
Código	**Nombre**	**Peso**	**Prioridad**
EC-20	Detalle de producto seleccionado	20	Alta
EC-21	Modal para cambio de tipo de compra a suscripción	8	Media
EC-22	Agregar reseña y calificación a un producto	5	Alta
EC-24	Detalle de carrito	13	Alta

Mientras que en la Tabla 18 se muestran las tareas técnicas a realizar dentro cada tarea que se encuentra en el backlog.

Tabla 18: Detalle técnico de HU de sprint 3

Product Backlog		
HU	**Código**	**Descripción**
EC-20	EC-20.1	Servicio REST para obtener detalle completo de producto por id.
	EC-20.2	Mostrar imagen principal e imágenes secundarias en un carrusel.
	EC-20.3	Deshabilitar botón de agregar al carrito mientras no haya seleccionado una cantidad.
	EC-20.4	Cargar productos relacionados en la parte inferior de la pantalla.

	EC-20.5	Mostrar precio principal del producto, junto a su información.
	EC-20.6	Maquetar interfaz acorde a mockups aprobados.
EC-21	EC-21.1	Cargar información completa del producto en un modal.
	EC-21.2	Deshabilitar botón de agregar al carrito mientras no haya seleccionado una cantidad.
	EC-21.3	Crear formulario para generar una compra de tipo suscripción.
	EC-21.4	Mostrar únicamente imagen principal del producto.
EC-22	EC-22.1	Crear servicio REST y Dto's para crear una reseña.
	EC-22.2	Crear formulario con validaciones dentro del detalle de producto.
EC-24	EC-24.1	Crear ruta para detalle de carrito.
	EC-24.2	Crear un servicio REST para obtener los impuestos de la tienda.
	EC-24.3	Calcular el precio final y mostrar al cliente.
	EC-24.4	Permitir editar el tipo de compra que se encuentra en la lista.
	EC-24.5	Permitir eliminar un producto de la lista.

2.12.2. Implementación

En la Figura 45, se muestra la pantalla de detalle del producto seleccionado por el cliente. Para dejar la pantalla lo más similar posible a los mockups, se utilizó la librería de componente PrimeNG.

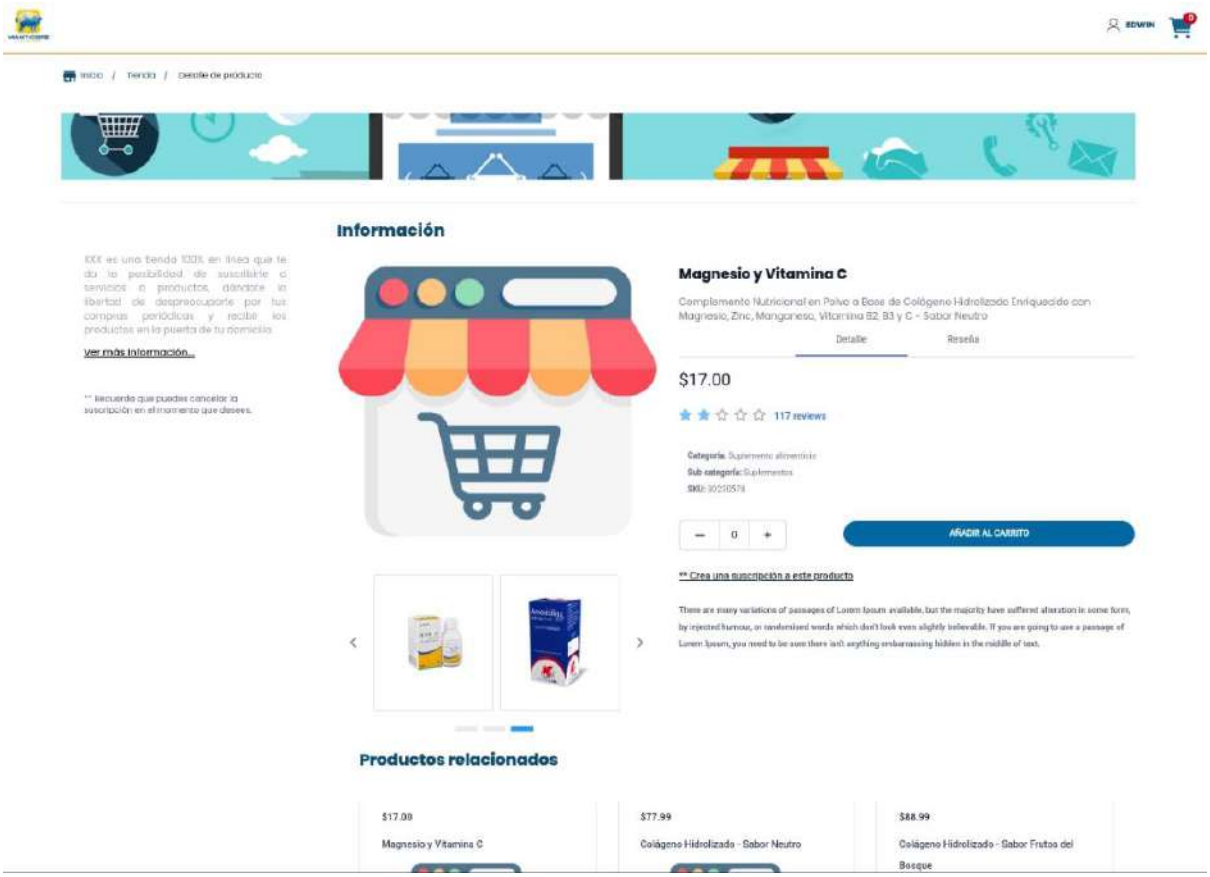


Figura 45: Pantalla detalle de producto

Por cada producto que el usuario visualice, va a poder dejar uno o varios comentarios como reseña del producto (Figura 46).

Figura 46: Sección para dejar reseñas a un producto

Además, se creó un modal con su respectivo formulario y validaciones, que nos permita agregar un producto como suscripción y poderle dar la gestión correspondiente a ese producto. El modal se pude visualizar en la Figura 47.

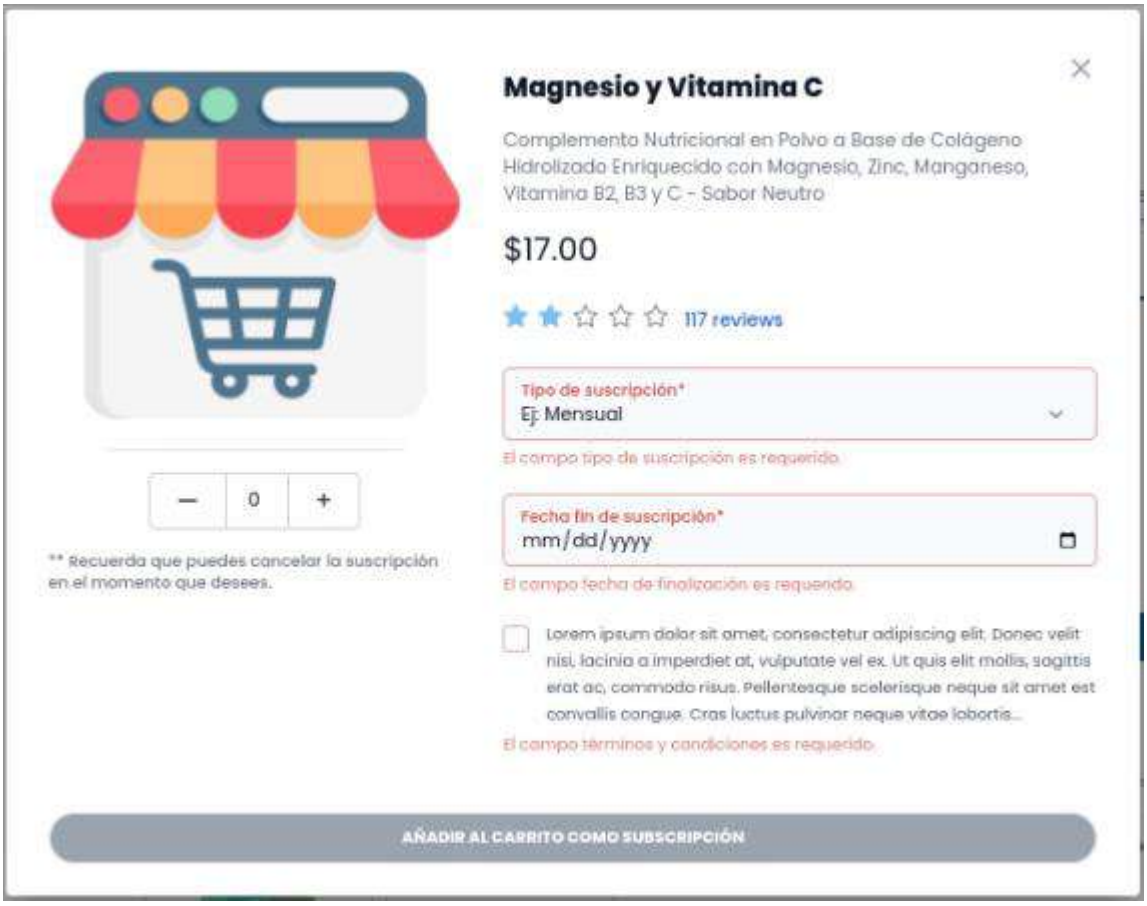


Figura 47: Modal para agregar producto como suscripción

Finalmente, se creó la ruta para listar los productos que han sido agregados al carrito de compras (Figura 48). Por el momento como aún no se ha implementado la lógica del carrito de compras, se utiliza datos de prueba.

Dentro de esta ruta el usuario podrá:

- Editar el tipo de compra haciendo uso del modal de la Figura 47, que es reutilizado en esta pantalla.
- Eliminar un producto que ya no sea de su interés.
- Ver el desglose de impuestos a agregarse en el precio final.
- Precio final de su compra.

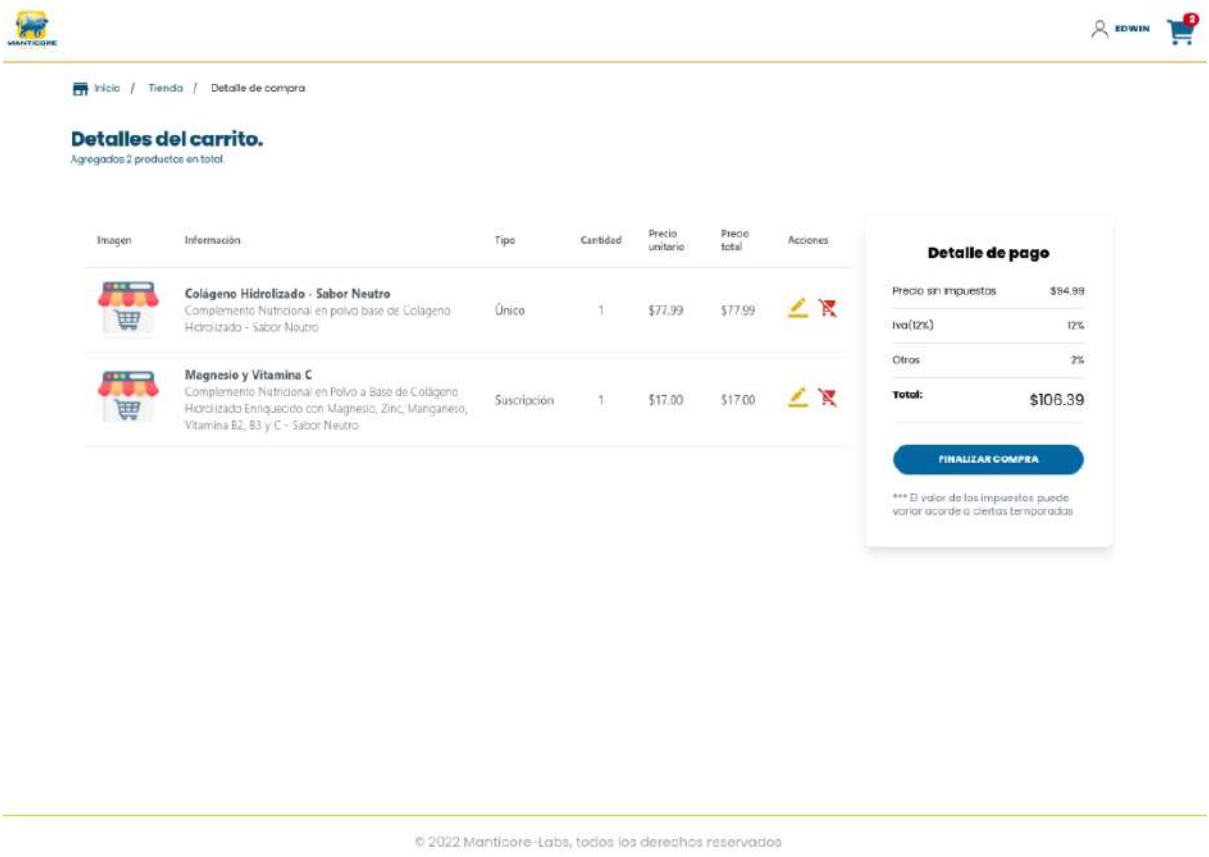


Figura 48: Pantalla detalle de carrito

2.12.3. Sprint Review

Tras finalizar el sprint 4, se obtuvieron los resultados mostrados en la Tabla *19* donde podemos observar el cumplimiento de las historias de usuario y el estado de cada una de ellas.

Tabla 19:Resultados del sprint 4

Sprint Review				
HU	**Código**	**Descripción**	**Cumplimiento**	**Observación**
EC-20	EC-20.1	Servicio REST para obtener detalle completo de producto por id.	SI	NA
	EC-20.2	Mostrar imagen principal e imágenes secundarias en un carrusel.	SI	NA
	EC-20.3	Deshabilitar botón de agregar al carrito mientras no haya seleccionado una cantidad.	SI	NA

	EC-20.4	Cargar productos relacionados en la parte inferior de la pantalla.	SI	NA
	EC-20.5	Mostrar precio principal del producto, junto a su información.	SI	NA
	EC-20.6	Maquetar interfaz acorde a mockups aprobados.	SI	NA
EC-21	EC-21.1	Cargar información completa del producto en un modal.	SI	NA
	EC-21.2	Deshabilitar botón de agregar al carrito mientras no haya seleccionado una cantidad.	SI	NA
	EC-21.3	Crear formulario para generar una compra de tipo suscripción.	SI	NA
	EC-21.4	Mostrar únicamente imagen principal del producto.	SI	NA
EC-22	EC-22.1	Crear servicio REST y DTO's para crear una reseña.	SI	NA
	EC-22.2	Crear formulario con validaciones dentro del detalle de producto.	SI	NA
EC-24	EC-24.1	Crear ruta para detalle de carrito.	SI	NA
	EC-24.2	Crear un servicio REST para obtener los impuestos de la tienda.	SI	NA
	EC-24.3	Calcular el precio final y mostrar al cliente.	SI	NA
	EC-24.4	Permitir editar el tipo de compra que se encuentra en la lista.	SI	NA
	EC-24.5	Permitir eliminar un producto de la lista.	SI	NA

2.12.4. Sprint Retrospective

Se cumplió con el objetivo propuesto al inicio de este.

¿Qué estuvo bien?

- Los componentes creados previamente permitieron un desarrollo más rápido para los formularios.
- El uso de componentes de PrimeNG nos ahorró tiempo.

- La comunicación con el PO fue bastante fluida y quedo satisfecho con los resultados del sprint.

¿Qué estuvo mal?

- No se tenía definido los tipos de suscripciones disponibles.
- No se tiene un estado global para el carrito de compras, por lo que se simuló utilizando un arreglo quemado en la ruta detalle de carrito.

¿Qué se debe empezar a hacer?

- Crear un servicio que permita manejar el estado del carrito de compras de manera global.
- Crear el carrito de compras en tiempo real.
- Tener un ambiente de pruebas para el e-commerce.

¿Qué se debe parar de hacer?

- NA.

El esfuerzo tomado para cumplir el objetivo del sprint se lo puede observar en el gráfico burndown de la Figura *49*. Se debe tomar en cuenta que, aunque hasta ahora ha sido el sprint con mayor peso, se logró llegar a la meta de los puntos planificados sin mayor dificultad, esto quiere decir que la comunicación con el PO y el diseño de la aplicación es bastante fluida.

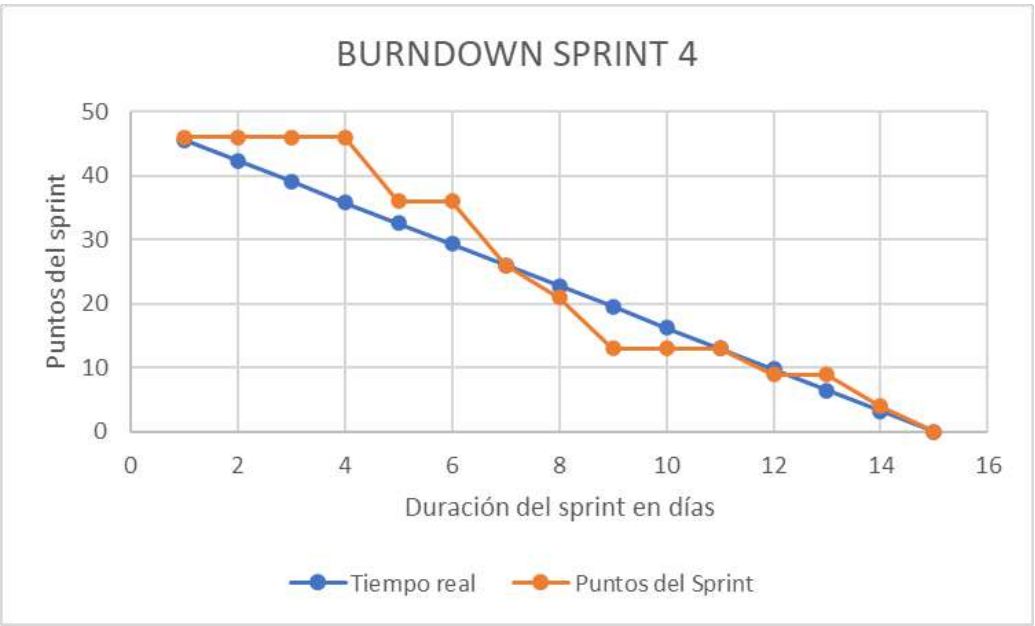


Figura 49: Burndown sprint 4

2.13. **Sprint 5**

Objetivos:

- Crear un carrito de compras, que permita al usuario gestionar sus compras.
- Crear una pantalla donde el usuario pueda ingresar sus datos y realizar la compra de uno o varios productos.
- Tener una sección para que el usuario administre su información.

2.13.1. Sprint Planning

En este sprint se finalizará el aplicativo e-commerce a nivel de frontend, en la Tabla 20, se muestra las tareas sobre las que se trabajará.

Tabla 20: Backlog sprint 5

Product Backlog			
Código	**Nombre**	**Peso**	**Prioridad**
EC-17	Listar artículos por empresa	5	Alta
EC-23	Carrito de compras	20	Alta
EC-25	Información de pago y facturación	13	Media
EC-26	Perfil de usuario	8	Media

A continuación, en la Tabla 21 se detalla las tareas técnicas a realizar para cada historia de usuario presente en el backlog.

Tabla 21: Tareas técnicas para HU del sprint

Product Backlog		
HU	**Código**	**Descripción**
EC-17	EC-17.1	Crear servicio REST que permita recibir parámetros dinámicos para poder obtener productos por medio de filtros.
	EC-17.2	Mostrar listado de productos en una card.
	EC-17.3	Realizar un diseño responsive para pantallas móviles.
EC-23	EC-23.1	Crear un aside para visualizar estado de compra.
	EC-23.2	Crear un servicio de tipo BehavorSubject, para mantener un estado global en la aplicación.
	EC-23.3	Crear métodos para agregar y eliminar productos del carrito.
	EC-23.4	Agregar diferenciador para productos que se agregan como suscripción.
	EC-23.5	Validar productos agregados en el carrito.
EC-25	EC-25.1	Crear servicio REST y Dto's para crear una nueva compra con información de pago y factura.
	EC-25.2	Crear ruta para realizar pago.
	EC-25.3	Maquetar vista de información de pago y facturación.

	EC-25.4	Crear formularios para pago y facturación.
	EC-25.5	Validar campos de formularios
EC-26	EC-26.1	Crear servicio REST y Dto’s para administrar el perfil de usuario.
	EC-26.2	Crear ruta para dashboard de usuario.
	EC-26.3	Maquetar dashboard de usuario
	EC-26.4	Modal y formulario para editar información relevante del usuario.

2.13.2. Implementación

En la Figura 50 y Figura 51, se puede observar el listado de productos cargados en base a una empresa, el diseño se realizó tanto para pantallas desktop y móviles respectivamente.

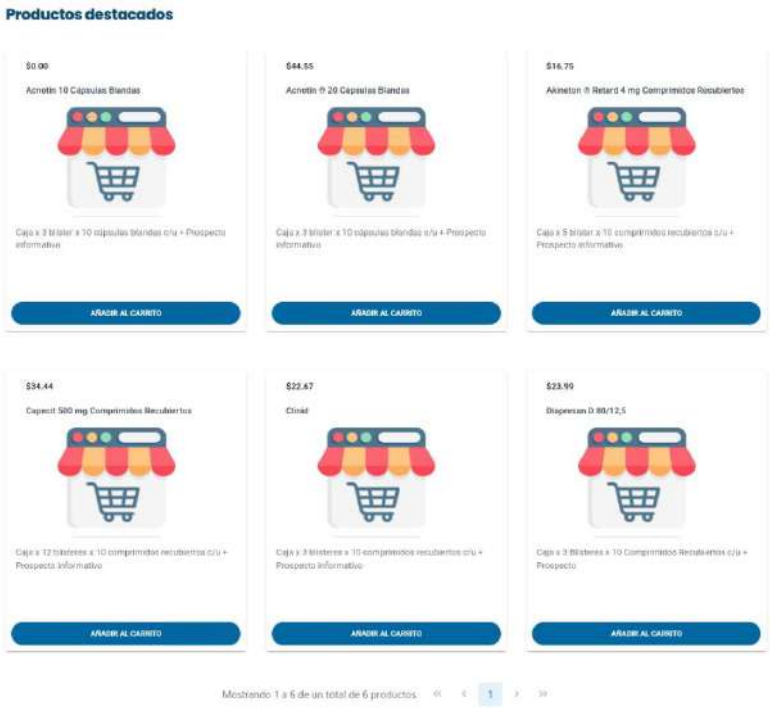


Figura 50: Listado de productos en pantallas desktop

Figura 51: Listado de productos en pantallas móvil

Para manejar el carrito de compras en la aplicación se optó por usar “BehaviorSubject” el cual nos ofrece la librería RxJS, este al ser un tipo especial de observable nos permitirá almacenar y transmitir múltiples valores, y compartir dicha información entre los componentes que se encuentren suscritos a dicho observable.

Adicionalmente se utilizó el patrón de diseño observador (Figura 52) el cual nos sugiere crear mecanismos de suscripción a la clase notificadora y cada que cada una de las clases individuales puedan suscribirse a un flujo de eventos provenientes de la clase notificadora.

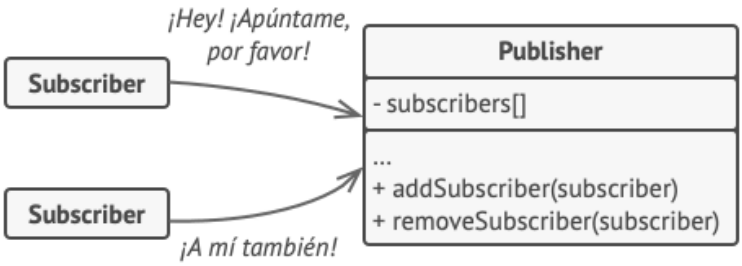


Figura 52: Patrón de diseño observador

Haciendo uso de los conceptos antes mencionados procedimos a realizar un servicio el cual manejara el estado del carrito de compras de manera global en la aplicación. Finalmente, en la Figura 53 se pude observar el listado de productos agregados al carrito de compras. También cabe recalcar que, para mantener la persistencia de datos en la aplicación, se utilizó el LocalStorage del navegador.

Figura 53: Carrito de compras

2.13.3. Sprint Review

Tras finalizar el sprint 5, se obtuvieron los resultados mostrados en la Tabla *22* donde se pude observar el cumplimiento de las historias de usuario y el estado de cada una de ellas.

Tabla 22: Resultados del sprint 5

Product Backlog				
HU	**Código**	**Descripción**	**Cumplimiento**	**Observación**
EC-17	EC-17.1	Crear servicio REST que permita recibir parámetros dinámicos para poder obtener productos por medio de filtros.	SI	NA
	EC-17.2	Mostrar listado de productos en una card.	SI	NA
	EC-17.3	Realizar un diseño resposive para pantallas móviles.	SI	NA
EC-23	EC-23.1	Crear un aside para visualizar estado de compra.	SI	NA

	EC-23.2	Crear un servicio de tipo BehavorSubject, para mantener un estado global en la aplicación.	SI	Se creó con el patrón de diseño observador.
	EC-23.3	Crear métodos para agregar y eliminar productos del carrito.	SI	Existe un solo servicio que maneja el estado del carrito de manera global.
	EC-23.4	Agregar diferenciador para productos que se agregan como suscripción.	SI	NA
	EC-23.5	Validar productos agregados en el carrito.	SI	NA
EC-25	EC-25.1	Crear servicio REST y Dto's para crear una nueva compra con información de pago y factura.	SI	NA
	EC-25.2	Crear ruta para realizar pago.	SI	NA
	EC-25.3	Maquetar vista de información de pago y facturación.	SI	NA
	EC-25.4	Crear formularios para pago y facturación.	SI	NA
	EC-25.5	Validar campos de formularios	SI	NA
EC-26	EC-26.1	Crear servicio REST y Dto's para administrar el perfil de usuario.	SI	NA
	EC-26.2	Crear ruta para dashboard de usuario.	SI	NA
	EC-26.3	Maquetar dashboard de usuario	SI	NA
	EC-26.4	Modal y formulario para editar información relevante del usuario.	SI	NA

2.13.4. Sprint Retrospective

Se cumplió con los objetivos propuestos al inicio del sprint.

¿Qué estuvo bien?

- Se cumplió todas las tareas en el tiempo establecido.

- El uso del patrón de diseño observador ayudo a generar un servicio escalable y mantenible.
- Las pantallas del cliente final se encuentran plenamente terminadas y su fidelidad a los mockups es alta.

¿Qué estuvo mal?

- NA

¿Qué se debe empezar a hacer?

- Configurar el CD para tener un ambiente de staging.
- Configurar el backend para el manejo de suscripciones y reportería.

¿Qué se debe parar de hacer?

- NA.

La reutilización de componentes e implementación de patrones de diseño facilitó la implementación y cumplimiento del sprint. En la Figura 54, se puede observar que el esfuerzo requerido no fue grande si tomamos como referencia la totalidad de los puntos sobre los que se trabajó.

Debido a que la gestión del tiempo en este sprint fue buena, se pudo tomar tiempo para correcciones responsive en pantalla ya implementadas en anteriores sprints.

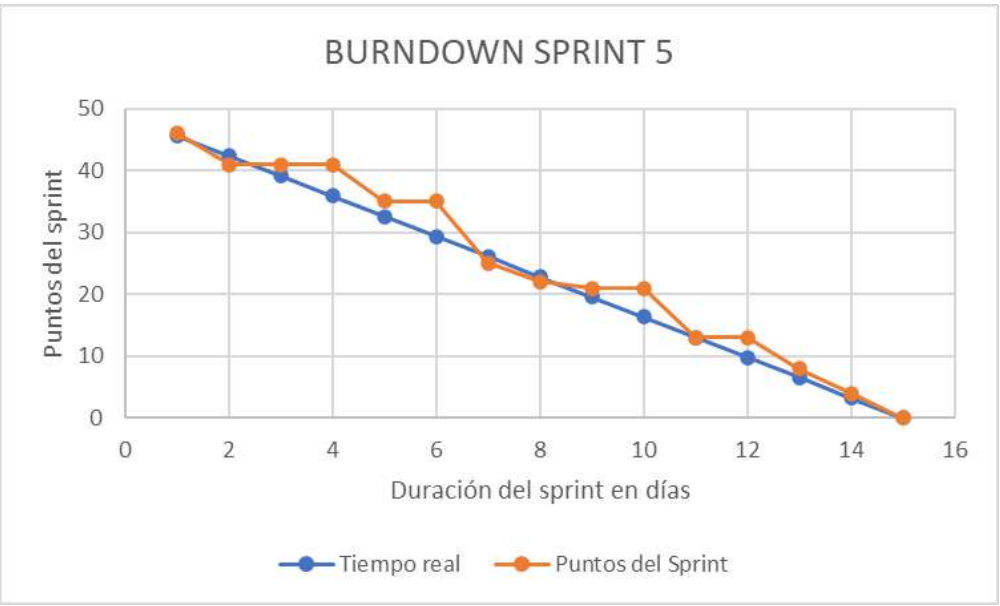


Figura 54: Burndown sprint 5

2.14. **Sprint 6**

Objetivos:

- Implementar recordatorio a producto suscrito.
- Crear colas auto gestionables para envío de notificaciones por medio de correo electrónico al cliente y administrador.
- Configurar el CD con Pulumi y AWS para tener un ambiente de staging del e-commerce.

2.14.1. Sprint Planning

En este sprint se trabajará netamente en el backend del submódulo e-commerce para la automatización de suscripciones y notificaciones. También se configuró el CD, utilizando el marco de trabajo “infraestructura como código”.

En la Tabla 23 se muestra el backlog sobre el que se trabajará.

Tabla 23: Backlog sprint 6

Product Backlog			
Código	**Nombre**	**Peso**	**Prioridad**
EC-28	Configuración de colas con Redis	13	Alta
EC-29	Generar suscripción a productos por medio de colas	20	Alta
EC-31	Configuración de CD	13	Media

A continuación, en la Tabla 24 se detalla las tareas técnicas para cada historia presente en el backlog.

Tabla 24: Detalle técnico de HU sprint 6

Product Backlog		
HU	**Código**	**Descripción**
EC-28	EC-28.1	Configurar la librería BullQueue en el backend.
	EC-28.2	Crear métodos processor y register.
	EC-28.3	Configurar CRON para cada tipo de suscripción.
	EC-28.4	Crear una nueva entidad para administrar los posibles valores de un CRON.
EC-29	EC-29.1	Integrar colas con método de comprar productos.
	EC-29.2	Probar comportamiento de las colas con la librería Bull.
	EC-29.3	Crear servicio para envío de emails
	EC-29.4	Crear templetes base para emails.

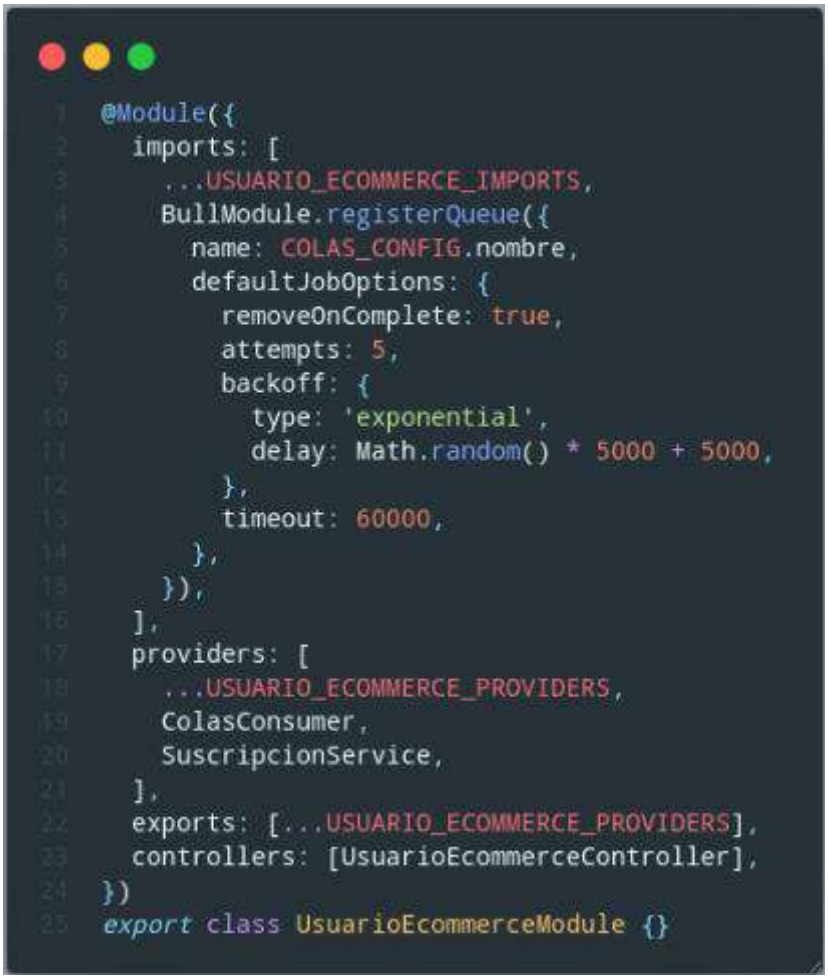
	EC-29.5	Comprobar correcta integración entre envío de emails, colas y método de compra.
EC-31	EC-31.1	Crear un proyecto de Pulumi dentro del proyecto e-commerce.
	EC-31.2	Generar script para deployar la aplicación en un S3 de AWS.
	EC-31.3	Dockerizar el proyecto.
	EC-31.4	Probar integración con los pipelines de GitLab.

2.14.2. Implementación

Para manejar las suscripciones a productos o servicios que se decidió utilizar colas FIFO, para lo cual se utilizó la librería BullQueue y la base de datos Redis.

Gracias a lo antes mencionado se pudo almacenar los procesos y notificar al usuario y administrador por medio de un correo electrónico, la entrega de su producto en la fecha establecida.

En la Figura 55, se puede observar la configuración inicial para el registro de una cola en el módulo de "*usuario ecommerce*".



```
@Module({
  imports: [
    ...USUARIO_ECOMMERCE_IMPORTS,
    BullModule.registerQueue({
      name: COLAS_CONFIG.nombre,
      defaultJobOptions: {
        removeOnComplete: true,
        attempts: 5,
        backoff: {
          type: 'exponential',
          delay: Math.random() * 5000 + 5000,
        },
        timeout: 60000,
      },
    }),
  ],
  providers: [
    ...USUARIO_ECOMMERCE_PROVIDERS,
    ColasConsumer,
    SuscripcionService,
  ],
  exports: [...USUARIO_ECOMMERCE_PROVIDERS],
  controllers: [UsuarioEcommerceController],
})
export class UsuarioEcommerceModule {}
```

Figura 55: Registro de colas en módulo “usuario ecommerce”

A continuación, en la Figura 56 se muestra la configuración del procesador, este se encargará de administrar los procesos que se encuentren encolados, a través de un identificador único, que en este caso será “subscription-job-name”.

```
@Processor(COLAS_CONFIG.nombre)
export class ColasConsumer {
  constructor(private readonly _suscripcionService: SuscripcionService) {}

  @OnQueueActive()
  onActive(job: Job) {
    console.log(`Procesando job ${job.id} de tipo ${job.name}`);
  }

  @Process(COLAS_CONFIG.job.name)
  async procesarSuscripciones(job: Job): Promise<void> {
    const {
      data: { suscripcion, email },
    } = job;

    const response = await this._suscripcionService.findSubscriptionById(
      suscripcion.id,
    );

    if (!response.sisHabilitado) {
      await job.finished();
    }

    await this._suscripcionService.manejarSuscripciones(suscripcion, email);
  }
}
```

Figura 56: Configuración de procesador

```
export const COLAS_CONFIG = {
  nombre: 'subscription',
  job: {
    name: 'subscription-job-name',
    id: 'subscription-job-id',
  },
};
```

Figura 57: Variables de configuración

Para comprobar el funcionamiento de las colas creadas en redis, procedemos a crear una nueva suscripción; utilizando un cliente de Redis podemos observar en la Figura 58 que se generó dentro de la cola llamada **“subscription”,** un nuevo proceso de tipo repetitivo.

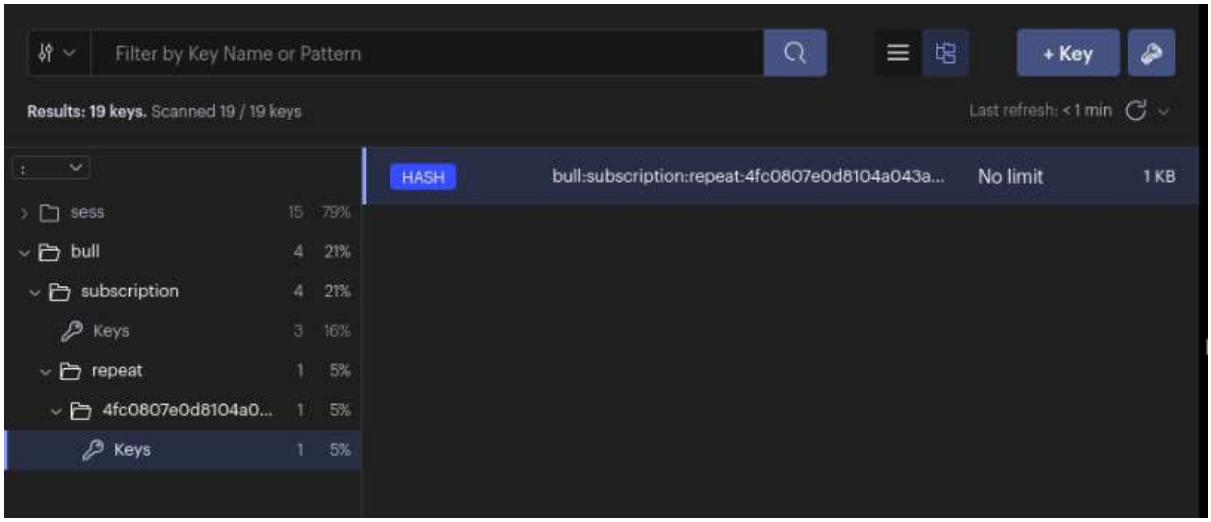


Figura 58: Información de cola en Redis

Dando clic sobre el hash podemos observar el detalle completo del proceso que se encuentra encolado, tal como se muestra en la Figura 59.

Los campos que podemos destacar son los siguientes:

- **Opts:** Incluye las opciones con las que fue creado el “job”, en donde se destaca su id interno, su identificador, el tipo y lo más importante, cada que tiempo se ejecutará.
- **Timestamp:** Es la fecha en la que fue creado.
- **Data:** Tiene toda la información referente al producto o servicio suscrito.
- **Name:** Es el nombre del proceso, esto servirá para que le procesador que se muestra configurado en la Figura 56 lo ejecute.

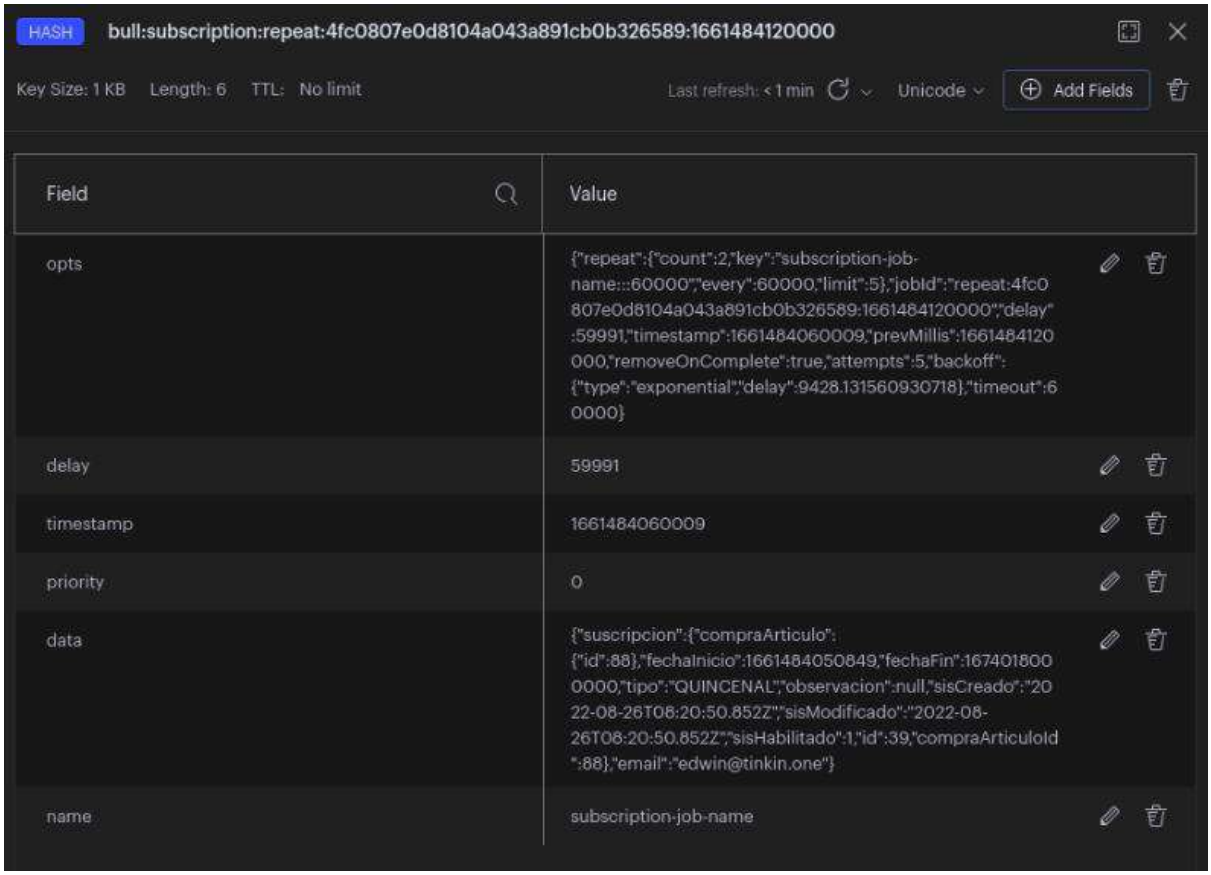
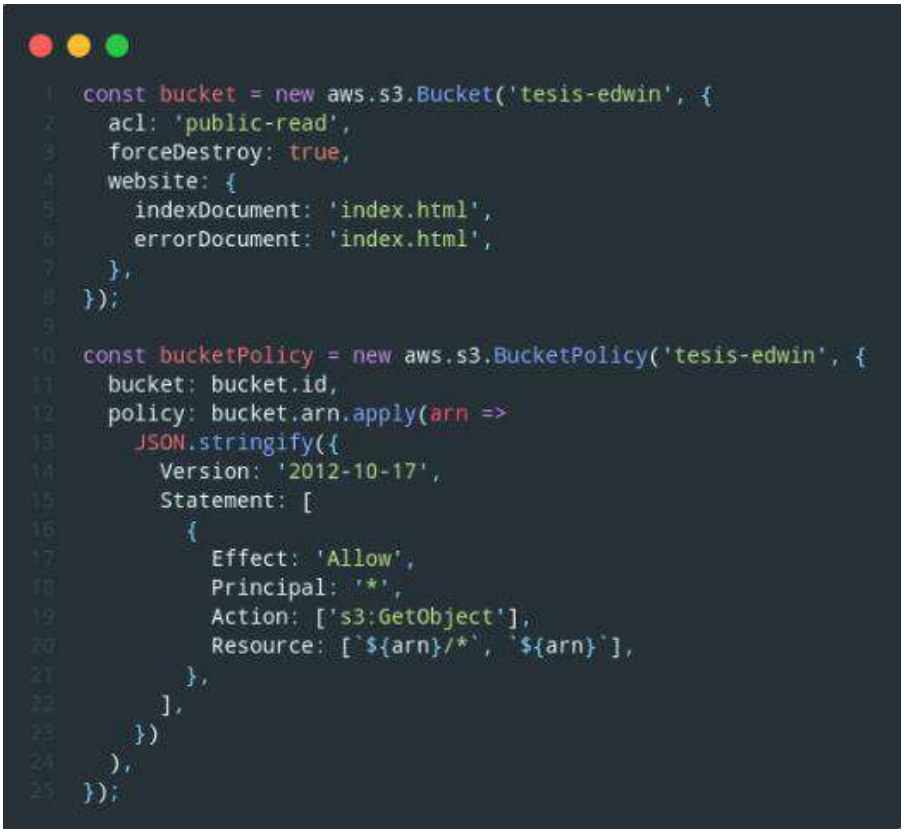


Figura 59: Detalle de tarea repetitiva (suscripción)

Para finalizar el sprint se desplegó la página e-commerce en un Bucket de S3 utilizando Pulumi, gracias a esta herramienta se pudo manejar las políticas y configuraciones del Bucket a nivel de código.

En la Figura 60, se puede observar la configuración a aplicar en el Bucket de nombre “tesis-edwin”, donde se desplegará nuestro e-commerce. Esta configuración está hecha en TypeScript, lo cual es un gran plus de Pulumi ya que, a diferencia de otras herramientas existentes en el mercado, no es necesario aprender un nuevo lenguaje para poder implementar infraestructura como código.



```
const bucket = new aws.s3.Bucket('tesis-edwin', {
  acl: 'public-read',
  forceDestroy: true,
  website: {
    indexDocument: 'index.html',
    errorDocument: 'index.html',
  },
});

const bucketPolicy = new aws.s3.BucketPolicy('tesis-edwin', {
  bucket: bucket.id,
  policy: bucket.arn.apply(arn =>
    JSON.stringify({
      Version: '2012-10-17',
      Statement: [
        {
          Effect: 'Allow',
          Principal: '*',
          Action: ['s3:GetObject'],
          Resource: [`${arn}/*`, `${arn}`],
        },
      ],
    })
  ),
});
```

Figura 60: Configuración Pulumi para Bucket S3

Para automatizar la ejecución del código de Pulumi, se agregó un nuevo stage dentro del archivo ".gitlab-ci.yml"(Figura 61) llamado "*deploy_s3*"; esto permitirá que el desarrollador se despreocupe del despliegue, ya que ahora el encargado de realizar dicha acción será Pulumi utilizando los pipelines de GitLab.

```
cd-deploy:
  image: ubuntu:20.04
  stage: deploy_s3
  before_script:
    - apt-get update -y
    - apt install curl -y
    - curl -sL https://deb.nodesource.com/setup_16.x | bash
    - curl -fsSL https://get.pulumi.com/ | bash
    - export PATH=$PATH:$HOME/.pulumi/bin
    - apt -y install nodejs
    - pulumi login
  script:
    - cd ./ecommerce-front/deploy-app
    - npm install
    - pulumi stack select dev
    - pulumi up -y
    - pulumi stack output url
  only:
    - desarrollo
```

Figura 61: Configuración Pulumi en archivo ".gitlab-ci.yml"

Una vez realizada la configuración se procedió a revisar los pipelines del repositorio, en la Figura 62 se pude observar que el stage de nombre "*deploy_s3*" se ha ejecutado con éxito.

Figura 62: Pipelines para despliegue de aplicación en S3

2.14.3. Sprint Review

Tras finalizar el sprint 6, se obtuvieron los resultados mostrados en la Tabla 25, ahí se puede observar el cumplimiento de las historias de usuario y el estado de cada una de ellas.

Tabla 25: Resultados del sprint 6

Product Backlog				
HU	**Código**	**Descripción**	**Cumplimiento**	**Observación**
EC-28	EC-28.1	Configurar la librería BullQueue en el backend.	SI	NA
	EC-28.2	Crear métodos processor y register.	SI	NA
	EC-28.3	Configurar CRON para cada tipo de suscripción.	SI	NA
	EC-28.4	Crear una nueva entidad para administrar los posibles valores de un CRON.	SI	NA
EC-29	EC-29.1	Integrar colas con método de comprar productos.	SI	NA
	EC-29.2	Probar comportamiento de las colas con la librería bull.	SI	NA
	EC-29.3	Crear servicio para envió de emails	SI	Se configuró un correo emisor, para lo cual se utilizó el correo con dominio de la Escuela Politécnica Nacional.
	EC-29.4	Crear templates base para emails.	SI	NA

	EC-29.5	Comprobar correcta integración entre envío de emails, colas y método de compra.	SI	NA
EC-31	EC-31.1	Crear un proyecto de Pulumi dentro del proyecto e-commerce.	SI	Se desplegó el proyecto e-commerce en un bucket de S3
	EC-31.2	Generar script para deployar la aplicación en un S3 de AWS.	SI	NA
	EC-31.3	Dockerizar el proyecto.	SI	NA
	EC-31.4	Probar integración con los pipelines de GitLab.	SI	NA

2.14.4. Sprint Retrospective

¿Qué estuvo bien?

- Se cumplieron todas las tareas en el tiempo establecido.
- Se logró configurar Pulumi, para poder desplegar los servicios y políticas de S3 desde código, y mantener una trazabilidad de este.
- Las colas utilizadas con Redis funcionan de la manera esperada.
- El envío de correos electrónicos como recordatorio permite al administrador manejar la entrega de sus productos.
- Se pudo indagar más a fondo el funcionamiento de “Infraestructura como Código”.

¿Qué estuvo mal?

- Se tuvo servicios algo limitados debido a la cuenta de prueba de AWS.

¿Qué se debe empezar a hacer?

- NA

¿Qué se debe parar de hacer?

- NA

La implementación de las colas ayudó a resolver el problema de tareas calendarizadas a nivel del backend, ya que en un inicio se tenía planeado utilizar un CronJob que cada noche analice las suscripciones activas. Pero con el uso de colas se pudo

manejar este proceso bajo demanda, es decir que, por cada producto suscrito, se generará un único proceso.

Finalmente se cumplieron con los objetivos propuestos al inicio del sprint, además de que se aprendió nuevas tecnologías como Pulumi y colas con Redis. A pesar de que hubo demasiada documentación que leer para generar una arquitectura escalable, se logró cumplir con el tiempo establecido y con los resultados esperados.

En la Figura 63, se puede observar el esfuerzo requerido durante el sprint.

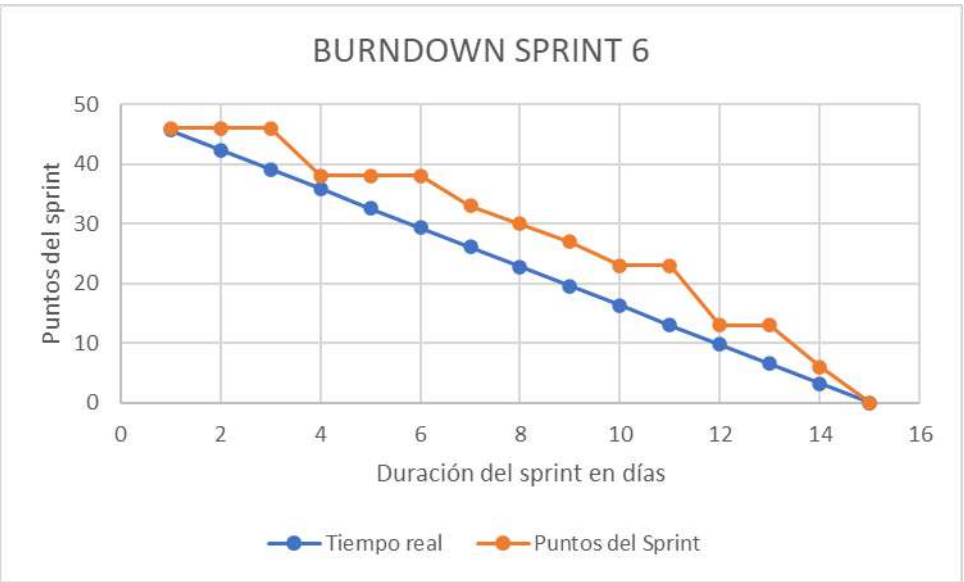


Figura 63: Burndown sprint 6

2.15. **Sprint 7**

Objetivos:

- Conectar el backend del aplicativo a Google Cloud Platform y consumir los servicios de BigQuery.
- Generar reportería en la nube basada en datos capturados del e-commerce.

2.15.1. Sprint Planning

En este sprint se trabajaó sobre la historia de usuario “EC-11”, la cual trata sobre reportería en la nube. Pero durante la creación del backlog inicial no se dimensionó el alcance, por lo cual se decidió junto al PO aumentar nuevas tareas para el presente sprint.

El backlog sobre el que se trabajará en este sprint se puede ver en la Tabla 26:

Tabla 26: Backlog sprint 7

Product Backlog			
Código	**Nombre**	**Peso**	**Prioridad**
EC-33	Realizar configuraciones necesarias en la consola de Google Cloud.	13	Alta
EC-11	Generar reportes con BigQuery	13	Alta

A continuación, en la Tabla 27 se detallan las tareas técnicas a realizar durante el presente sprint.

Tabla 27: Detalle técnico de HU sprint 7

Product Backlog		
HU	**Código**	**Descripción**
EC-33	EC-33.1	Crear una cuenta en Google Cloud.
	EC-33.2	Activar una cuenta de facturación temporal.
	EC-33.3	Generar tokens y habilitar servicios para BigQuery.
	EC-33.4	Generar permisos en KMS (Key Managemet Service API)
EC-11	EC-11.1	Crear un dataset.
	EC-11.2	Crear una tabla con el tipo de dato, dentro del dataset.
	EC-11.3	Agregar registro de compras.
	EC-11.4	Conectar BigQuery con DataStudio para mostrar reportería.

2.15.2. Desarrollo

Para poder utilizar los servicios que ofrece BigQuery, se creó una nueva cuenta dentro de Google Cloud Platform, donde se creó el proyecto "*tesis-ecommerce*", el cual se pude observar en la Figura 64, dentro de este proyecto se habilitó la Api de BigQuery.

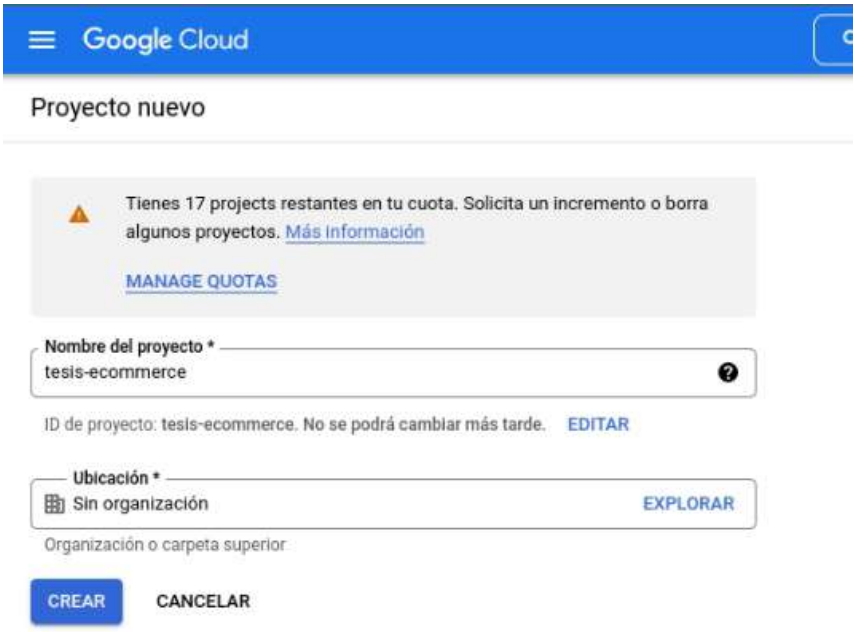


Figura 64: Creación del proyecto en Google Cloud Platform

Posteriormente utilizando "*Cloud Key Management Service (*Figura *65)*" administramos las credenciales de acceso a nuestra cuenta de Google Cloud Platform (GCP). Este servicio nos permitió gestionar las claves de diversos servicios de manera sencilla.

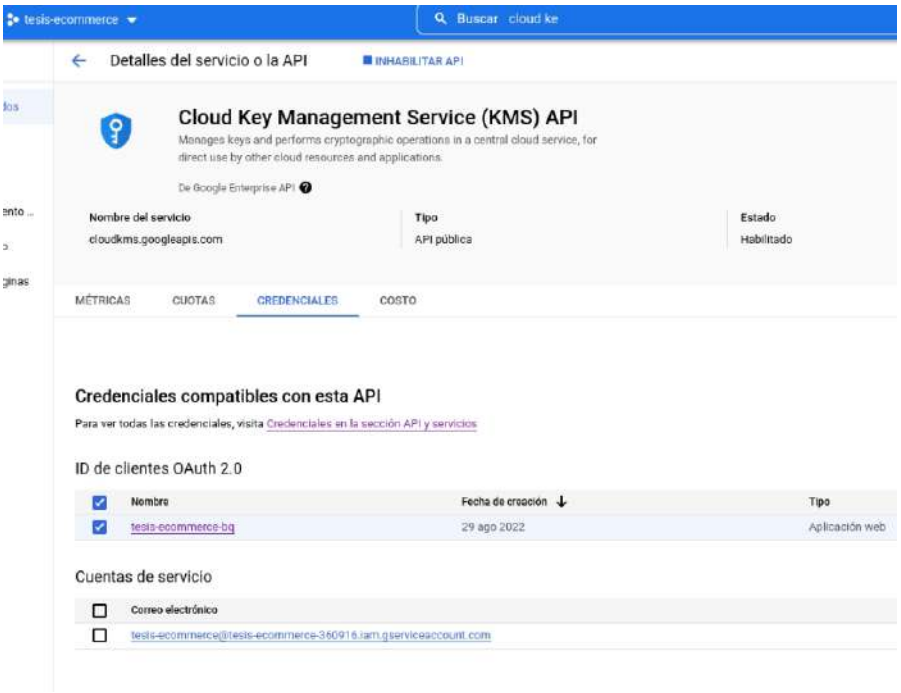


Figura 65: Cloud Key Management Service

A continuación, se creó un dataset (Figura 66) con las propiedades relevantes para generar los reportes solicitados por el PO, y por medió del backend se procedió a cargar los datos cada vez que un usuario realiza una nueva compra, tal como se muestra en la Figura 67.

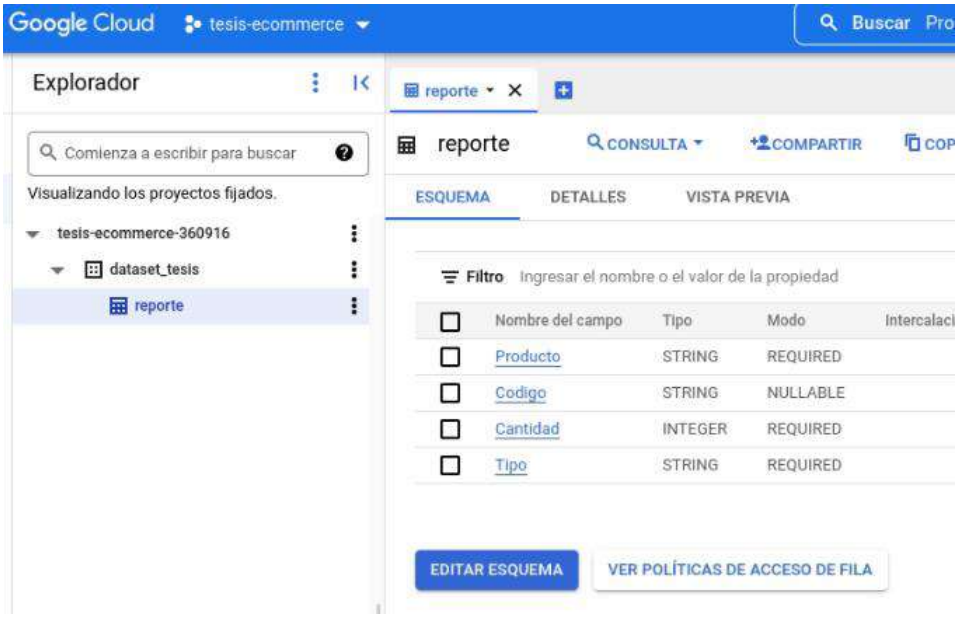


Figura 66: Dataset para reportería

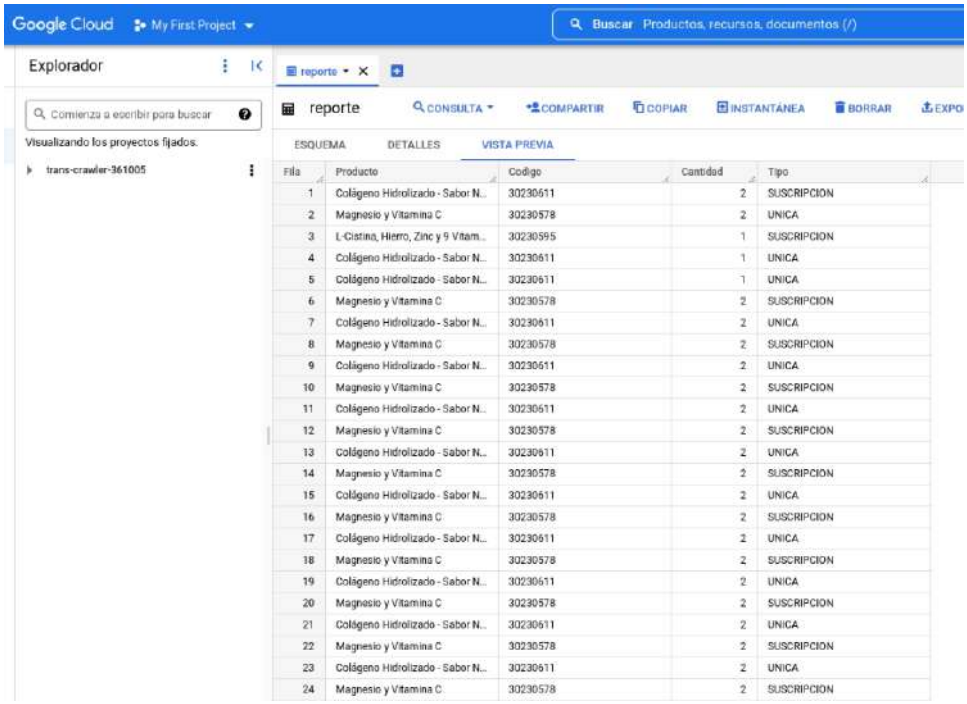


Figura 67: Datos cargados en dataset

Finalmente, utilizando BigQuery y Data Studio generamos un reporte que permite al administrador comparar qué tipo de compra se realiza con más frecuencia en la aplicación y observar los productos más vendidos. El reporte se lo puedo observar en la Figura 68.

Figura 68: Reporte de compras en Data Studio

2.15.3. Sprint Review

Al finalizar el sprint 7, se lograron los resultados descritos en la Tabla 28, ahí se puede observar el cumplimiento de las historias de usuario.

Tabla 28: Resultados del sprint 7

Product Backlog				
HU	**Código**	**Descripción**	**Cumplimiento**	**Observación**
EC-33	EC-33.1	Crear una cuenta en Google Cloud.	SI	NA
	EC-33.2	Activar una cuenta de facturación temporal.	SI	Se agregó una tarjeta de crédito para poder acceder a los servicios requeridos.
	EC-33.3	Generar tokens y habilitar servicios para BigQuery.	SI	NA
	EC-33.4	Generar permisos en KMS (Key Managemet Service API)	SI	NA
EC-11	EC-11.1	Crear un dataset	SI	NA
	EC-11.2	Crear una tabla con el tipo de dato, dentro del dataset	SI	NA
	EC-11.3	Agregar registro de compras	SI	NA

	EC-11.4	Conectar BigQuery con DataStudio para mostrar reportería.	SI	NA

2.15.4. Sprint Retrospective

¿Qué estuvo bien?

- Se avanzó correctamente con las tareas del sprint, además de realizar QA de tareas anteriores.
- Se generó el reporte solicitado por el PO
- Se lograron integrar las herramientas de Google Cloud Platform a el backend de Manticore-Labs.
- El sistema se encuentra estable, y todo funciona acorde a los requerimientos iniciales.

¿Qué estuvo mal?

- No se midió bien el alcance de la tarea de reportería, por lo que durante el planning del sprint se debió aumentar tareas.
- Para tener acceso a todas las funcionalidades de BigQuery se debe tener una cuenta de pago dentro de Google Cloud Platform.

¿Qué se debe empezar a hacer?

- Utilizar una tarjeta de crédito de la empresa.

¿Qué se debe parar de hacer?

- NA

A pesar de haber agregado una nueva tarea a la ya existente sobre reportería, se pudo llegar a cumplir con el objetivo del sprint sin mayor problema, además se realizaron tareas de QA sobre la aplicación en el tiempo restante del sprint.

En el gráfico burndown que se muestra a continuación en la Figura *69*, se pude observar que se terminó con las tareas del sprint antes del tiempo previsto, por lo que los últimos días no se realizaron más que tareas de QA.

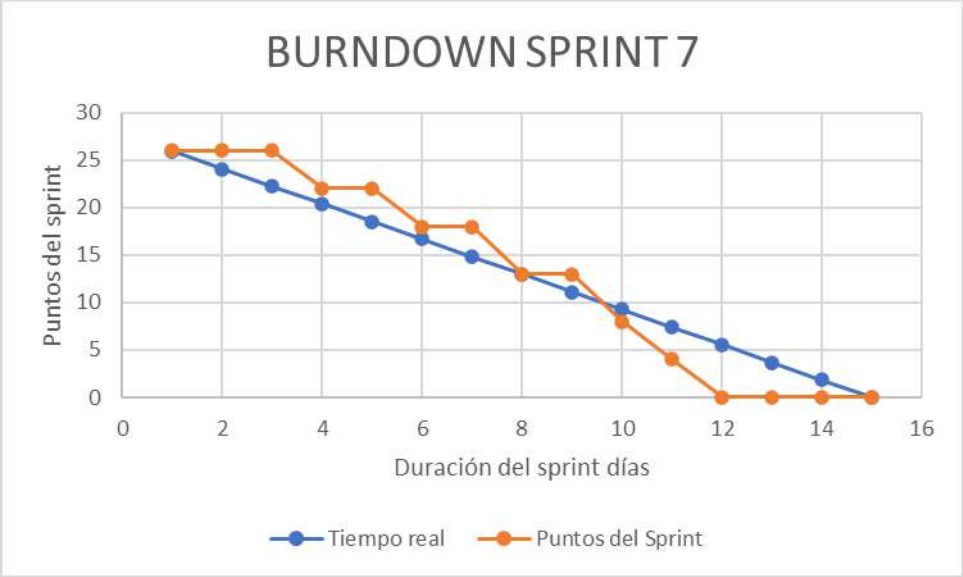


Figura 69: Burndown sprint 7

3. RESULTADOS Y DISCUSIÓN

3.1. **Usabilidad del Sistema**

En el presente capítulo se describen los resultados obtenidos a nivel de usabilidad, para lo cual se utilizó el método de *Escalas de Usabilidad de un Sistema* o por sus siglas en inglés SUS.

3.1.1. Método SUS

El método SUS fue desarrollado por John Brooken en el año de 1986, trata de diez afirmaciones muy generales, a las cuales los usuarios deben valorar usando una escala de cinco valores, de “completamente de acuerdo” a “completamente en desacuerdo” [40]. Este método nos va a permitir conocer la usabilidad que percibe los usuarios que harán uso del sistema.

Algunas de las ventajas de utilizar SUS son[41]:

- Pequeñas muestras nos proporcionan resultados confiables.
- Es una escala fácil de administrar a los participantes.
- Diferencia de manera efectiva entre sistemas utilizables e inutilizables.

Las preguntas que se encuentran dentro del cuestionario de SUS son las siguientes [41]:

1. Creo que me gustaría usar este sistema con frecuencia.
2. Encontré el sistema innecesariamente complejo.
3. Encontré el sistema fácil de usar.
4. Creo que necesitaría el apoyo de un técnico para poder utilizar este sistema.
5. Descubrí que las diversas funciones de este sistema estaban bien integradas.
6. Pensé que había demasiada inconsistencia en este sistema.
7. Me imagino que la mayoría de la gente aprendería a usar este sistema muy rápidamente.
8. Encontré el sistema muy engorroso de usar.
9. Me sentí muy confiado usando el sistema.
10. Necesitaba aprender muchas cosas antes de poder ponerme en marcha con este sistema.

Para el cálculo del resultado final se debe realizar lo siguiente:

- Utilizando la escala de Linkert, cada enunciado tendrá un valor equivalente a 1,2,3,4 o 5, en función de la respuesta, donde 1 será “completamente en desacuerdo” y 5 “completamente de acuerdo”.
- Para las preguntas impares (1, 3, 5, 7, 9), se debe sumar cada una y restar el resultado total para 5.
- Para las preguntas pares (2, 4, 6, 8, 10), se debe sumar cada una y restar ese total a 25.
- Sumar ambos resultados y multiplicar por 2.5.
- El resultado final obtenido será evaluado sobre 100.

Jeff Sauro, tras realizar un estudio en más de 500 webs y aplicaciones, concluyó que el puntaje promedio es de 68 [42], por lo cual un resultado inferior a este valor, nos indica que nuestro aplicativo tiene varios aspectos que corregir.

3.1.2. Realización de encuesta

La encuesta se realizó a 12 personas en un rango de edad entre 24 y 36 años, en su mayoría estudiantes y ex estudiantes de la Escuela Politécnica Nacional, personal del área de la salud y personas que realizan compras de medicinas de manera recurrente en farmacias físicas.

Las pruebas que cada usuario realizó forman parte de las actividades a realizar por el usuario final, quien tendrá acceso total al aplicativo e-commerce, por lo cual podrán:

- Registrarse en el aplicativo
- Navegar por el listado de productos
- Ver detalles de un producto
- Ver estado de carrito, agregar y eliminar productos de este
- Realizar una compra
- Listar sus suscripciones
- Modificar datos de su perfil

A cada uno de los encuestados se les pidió realizar 12 tareas específicas, las cuales llegaron a cubrir aspectos principales de funcionalidad de la aplicación e-commerce, y tras lo cual se le pidió a cada uno responder la encuesta SUS.

En la Tabla 29, se muestra de manera global los resultados obtenidos por cada pregunta y participante, junto a su score total y promedio SUS.

Tabla 29: Resultados de encuesta SUS

-	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Raw Score	Total
Usuario 1	5	1	5	1	5	1	5	1	5	3	38	95
Usuario 2	4	2	5	1	3	1	5	1	4	2	34	85
Usuario 3	4	3	5	1	5	1	5	1	4	3	34	85
Usuario 4	5	1	4	1	5	1	4	2	4	3	34	85
Usuario 5	5	1	4	3	5	1	5	1	3	3	33	82.5
Usuario 6	5	1	5	1	4	1	5	1	5	2	38	95
Usuario 7	5	1	5	1	5	2	4	1	5	4	35	87.5
Usuario 8	4	2	4	1	5	1	4	1	4	2	34	85
Usuario 9	5	1	5	2	4	2	5	1	5	3	35	87.5
Usuario 10	5	1	5	1	5	1	5	1	5	3	38	95
Usuario 11	5	1	5	1	5	1	4	1	4	3	36	90
Usuario 12	4	1	5	1	5	1	5	1	4	1	38	95
										Promedio	35.58	88.96

Una vez realizado los cálculos correspondientes, obtuvimos un resultado de 88.96/100. Por lo cual, basándonos en el valor mínimo de usabilidad (68) podemos concluir que la aplicación tiene una usabilidad bastante buena, lo cual para una aplicación e-commerce es algo crítico, debido a que las distintas pantallas deben tener interfaces de usuario que sean amigables e intuitivas y que motiven al usuario volver a usar la aplicación.

Aunque se obtuvo una nota bastante buena, existen algunos aspectos que se pueden mejorar en versiones posteriores, como por ejemplo el cambiar el método de crear suscripciones a un producto, en el cual se podría reemplazar el modal por una nueva pestaña dentro de la información del producto.

En el Anexo 6.6, se encuentra el enlace hacia la encuesta realizada y las gráficas de los resultados por respuesta.

3.2. **Producto final**

El sistema consta de dos aplicativos frontend y un backend.

Uno de los frontend (BackOffice) y el backend se encuentra integrados dentro del sistema principal de Manticore-Labs como submódulos. Este nuevo submódulo se integró con los ya existentes, como Empresa, Articulo y Artículo Empresa.

Finalmente, para el aplicativo del cliente final se creó un proyecto frontend el cual consume servicios y datos del backend de Manticore-Labs.

A continuación, se detalla cada una de las pantallas implementadas, tanto a nivel de BackOffice como del cliente final.

3.2.1. Módulo Grupo

En la Figura 70 se muestra la pantalla de grupos, este módulo nos permite gestionar los grupos que tendrá el sistema; por lo cual esto nos permitirá categorizar y parametrizar los artículos existentes en el sistema; por ejemplo, tener un grupo solo de artículos referente a medicina.

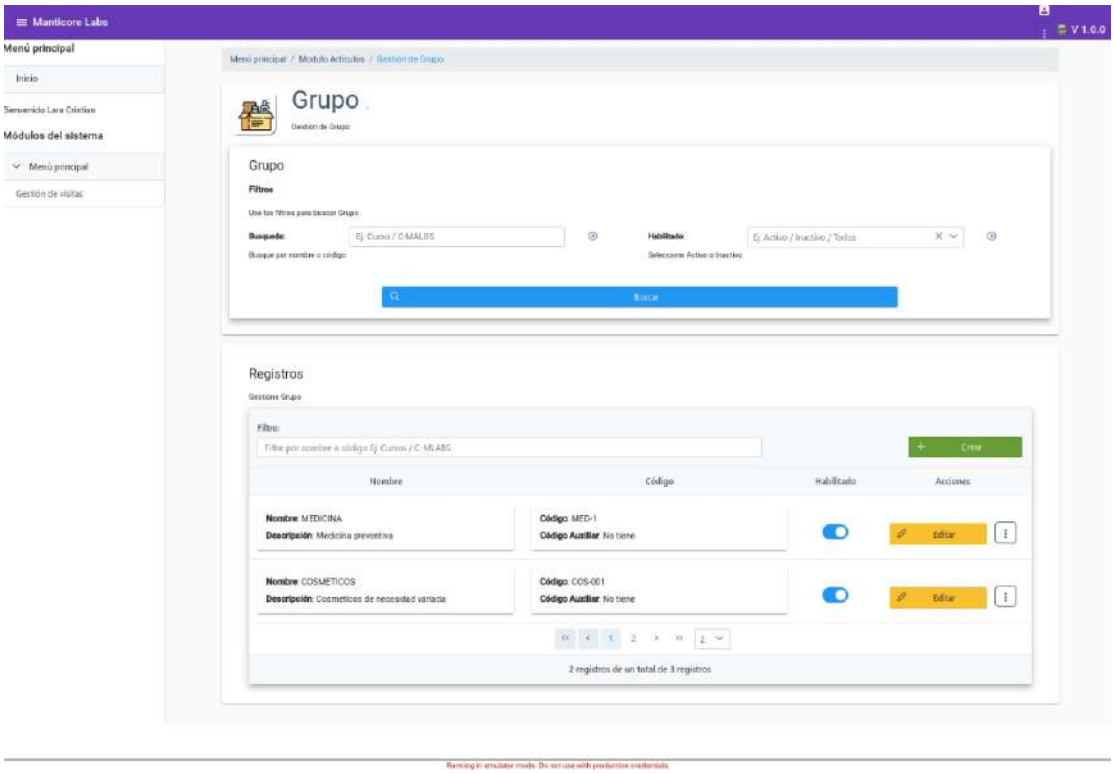


Figura 70: Pantalla módulo grupo

3.2.2. Módulo Subgrupo

En la Figura 71 se muestra la pantalla de subgrupos, este módulo nos permite gestionar los subtipos que pueden existir dentro de un grupo específico; por ejemplo: Medicamentos genéricos.

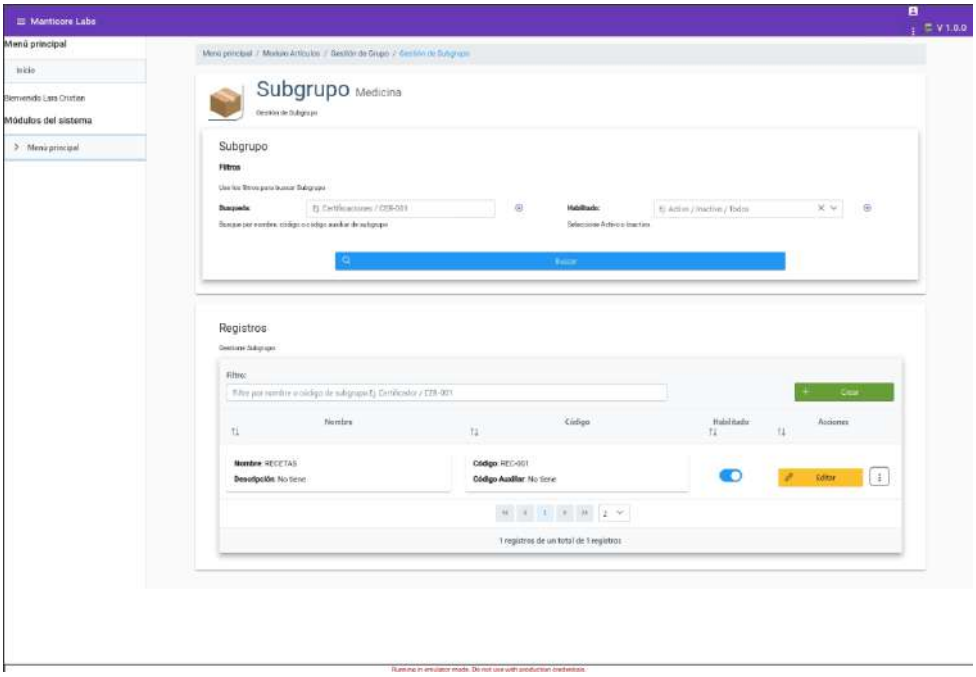


Figura 71: Pantalla módulo subgrupo

3.2.3. Módulo Artículo

En la Figura 72 se muestra la pantalla de artículos. Este módulo nos permite gestionar los artículos que se manejarán de manera global dentro del aplicativo. Cada uno de estos artículos serán asignados a una o varias empresas, con esto evitamos tener artículos duplicados en el sistema.

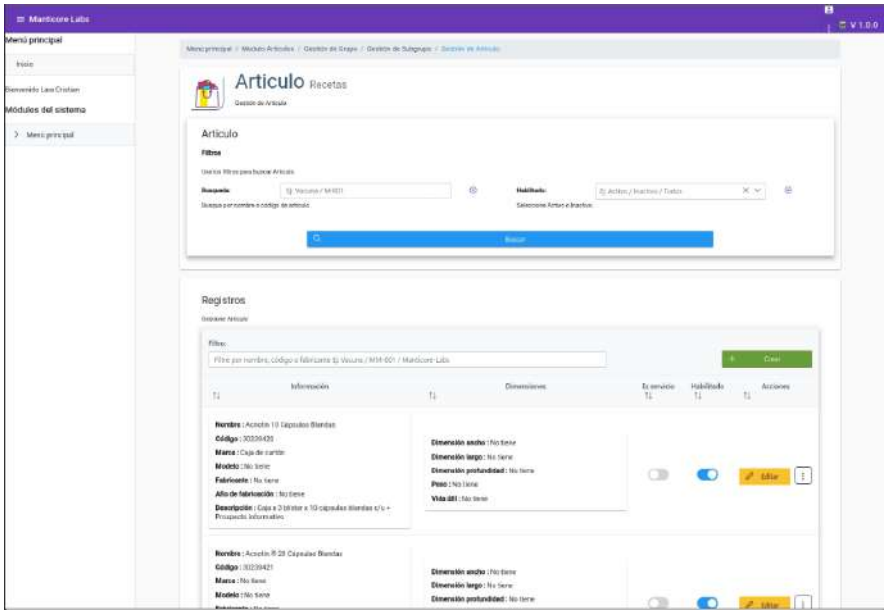


Figura 72: Pantalla módulo artículo

3.2.4. Módulo Etiqueta Artículo

En la Figura 73 se muestra la pantalla de etiqueta por artículo. Este módulo nos permite agregar etiquetas específicas a un artículo, permitiendo al usuario agregar uno o varios identificadores al artículo, por ejemplo: Medicamento Bago.

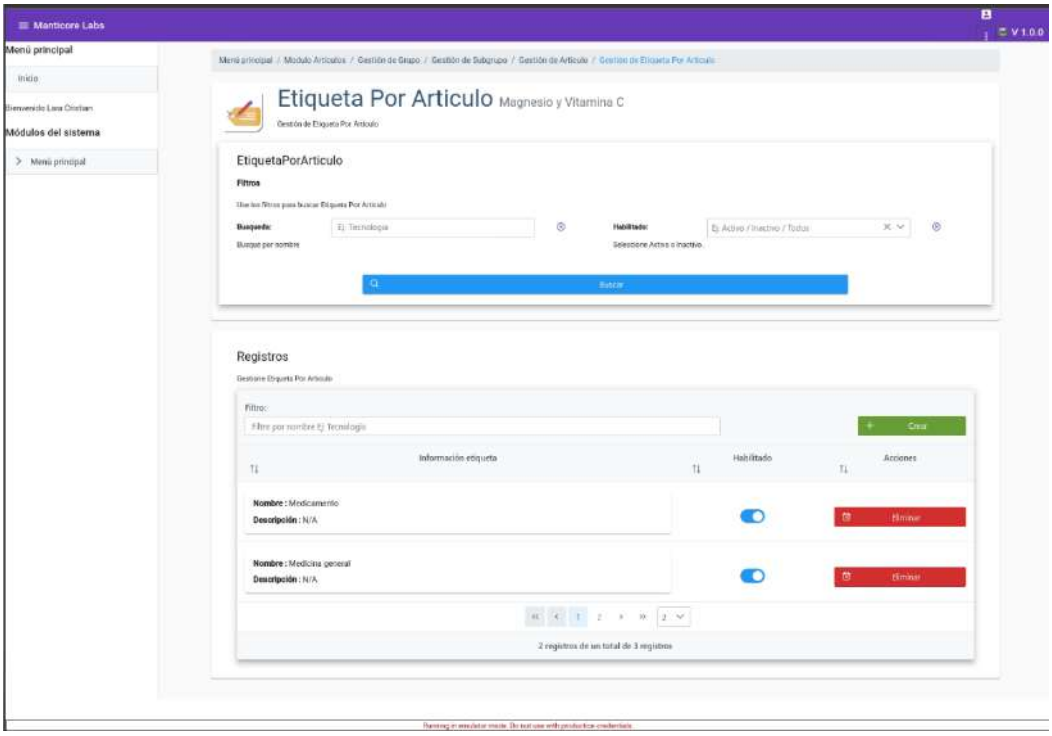


Figura 73: Pantalla módulo etiqueta artículo

3.2.5. Módulo Artículo Empresa

En la Figura 74 se muestra la pantalla de artículo por empresa. Este módulo nos permite agregar artículos a una empresa en específico, y es el módulo que alimentará de los artículos disponibles para ser listados en el aplicativo e-commerce, dependiendo la empresa que lo utilice.

Aquí se pude deshabilitar un artículo, en caso de no tener stock o ya no querer mostrar en el aplicativo e-commerce.

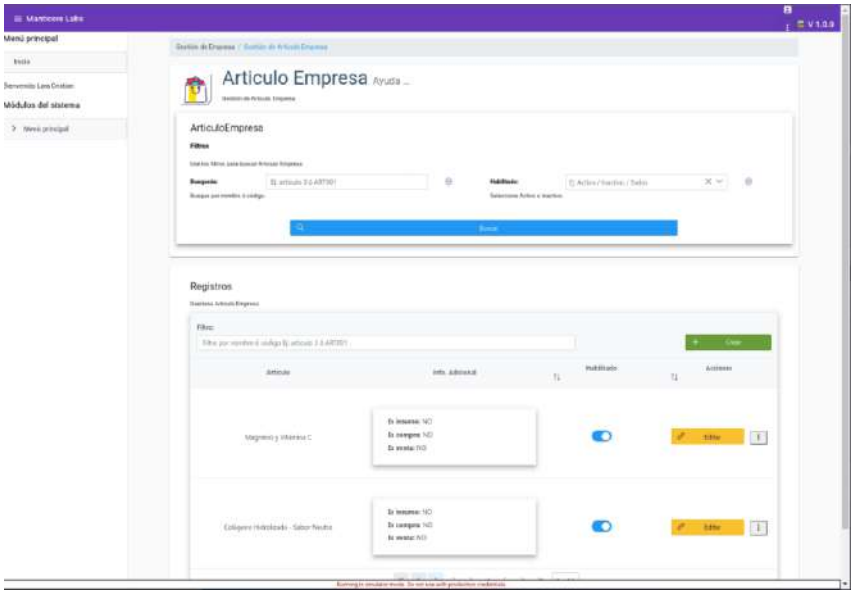


Figura 74: Pantalla módulo artículo empresa

3.2.6. Módulo Precio

En la Figura 75 se muestra la pantalla precio. Este módulo nos permite gestionar los precios asignados a un artículo empresa, se debe tener en cuenta que únicamente se puede tener un precio habilitado, el cual será mostrado en el aplicativo e-commerce, y con el cual se realizará los cobros respectivos.

Esta pantalla permite tener precios dinámicos para los artículos, de esta forma evitamos modificar código para cambiar los precios de cada artículo, dependiendo la empresa.

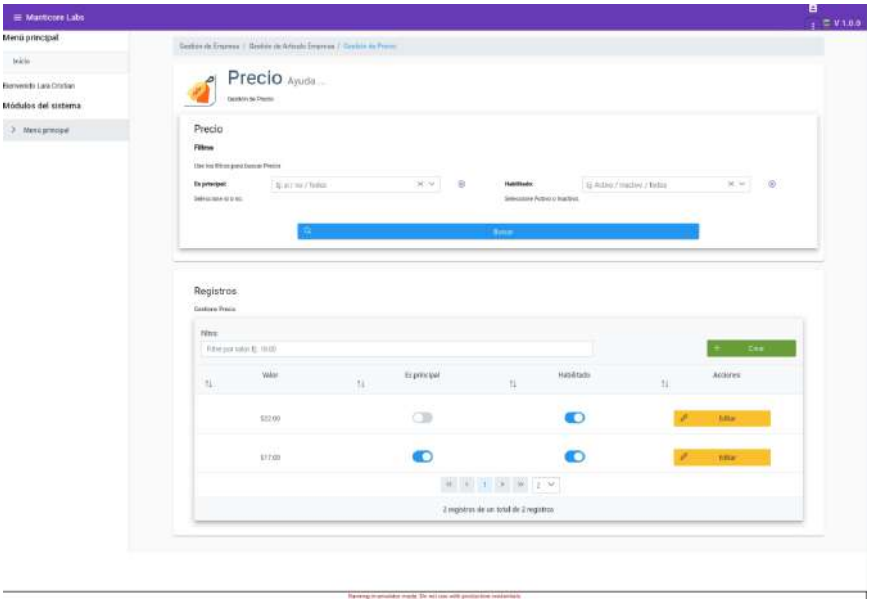


Figura 75: Pantalla módulo precio

3.2.7. Módulo Impuesto

En la Figura 76 se muestra la pantalla de impuesto. Este módulo nos permite gestionar los impuestos a aplicarse a cada una de las compras realizados dentro del e-commerce, lo cual nos ayuda a tener cálculos dinámicos en diversos impuestos que maneje cada empresa.

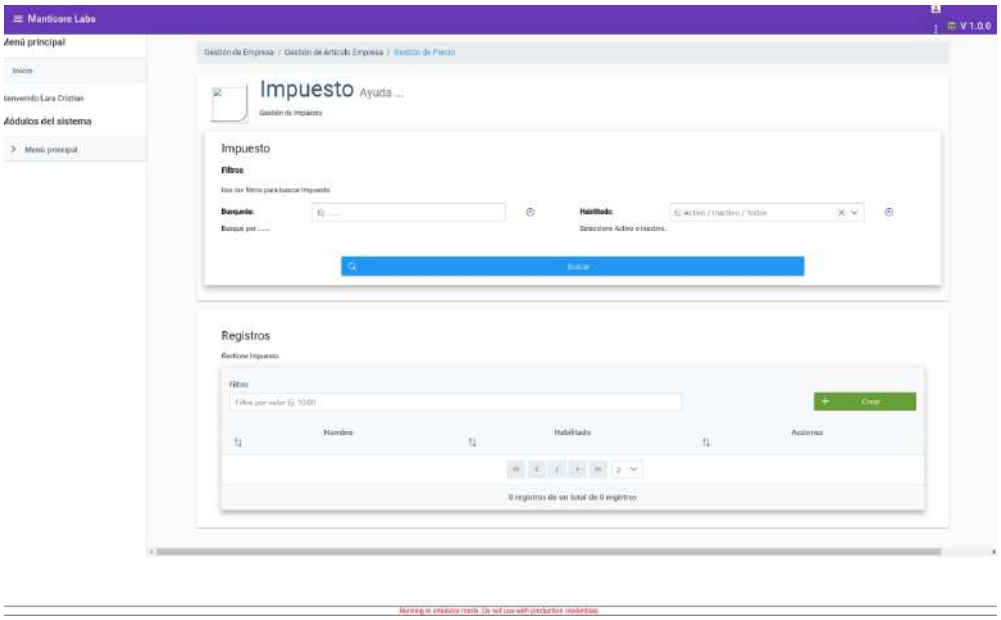


Figura 76: Pantalla módulo impuesto

3.2.8. Módulo Suscripción

En la Figura 77 se muestra la pantalla de suscripción. Este módulo nos permite listar por tipo y estado cada una de las suscripciones realizadas dentro de la plataforma, además de visualizar información complementaria como fecha de inicio y fecha de fin de la suscripción, también permite al administrador deshabilitar una suscripción activa, en caso de que el cliente lo requiera.

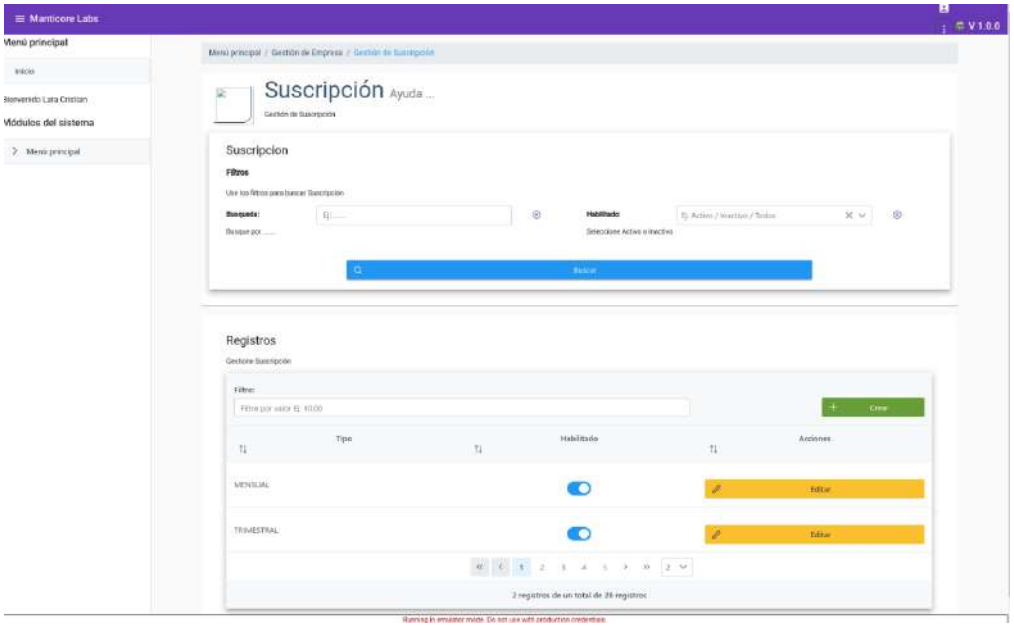


Figura 77: Pantalla módulo suscripción

3.2.9. Módulo Pago

En la Figura 78 se muestra la pantalla de pago. Este módulo es netamente visual, va a ayudar al administrador a visualizar un listado de pagos realizados dentro de la aplicación e-commerce, y le permitirá llevar un balance de las ventas realizadas.

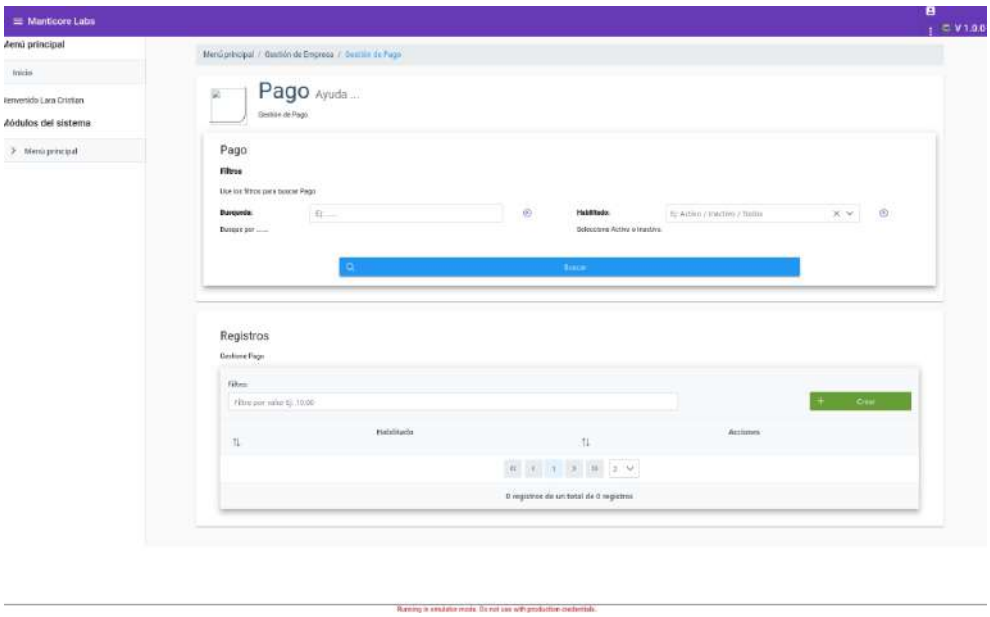


Figura 78: Pantalla módulo pago

3.2.10. Módulo Cliente

En la Figura 79 se muestra la pantalla de clientes. Este módulo va a permitir gestionar la información de todos los clientes registrados en el aplicativo e-commerce, también permitirá que el administrador reinicialice la contraseña en caso de que el cliente lo solicite.

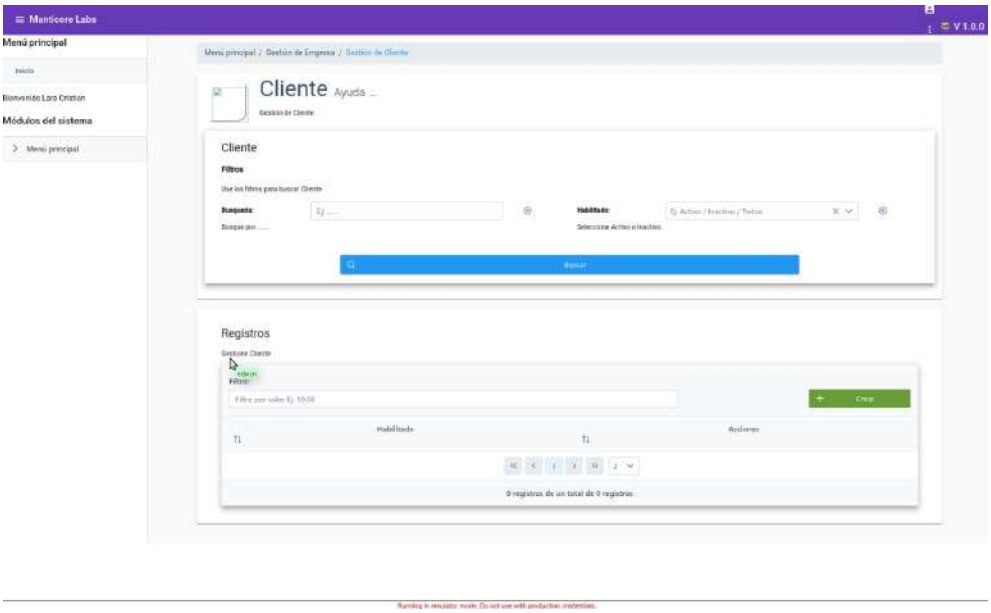


Figura 79: Pantalla módulo cliente

3.2.11. Aplicación E-Commerce cliente final

Para el cliente final se diseñó una aplicación amigable, que permita listar los productos, filtrarlos y visualizar de manera detallada cada uno de ellos tal como se muestra en la Figura 80 y Figura 81.

También esta aplicación permite a un usuario registrarse para realizar una compra. Dentro del aplicativo se manejan dos tipos de compra, los cuales son normales o por suscripción. Finalmente, el administrador podrá visualizar un reporte de productos más comprados dentro de la aplicación, para lo cual se utilizó reportería en la nube con BigQuery, tal como se muestra en la Figura 68.

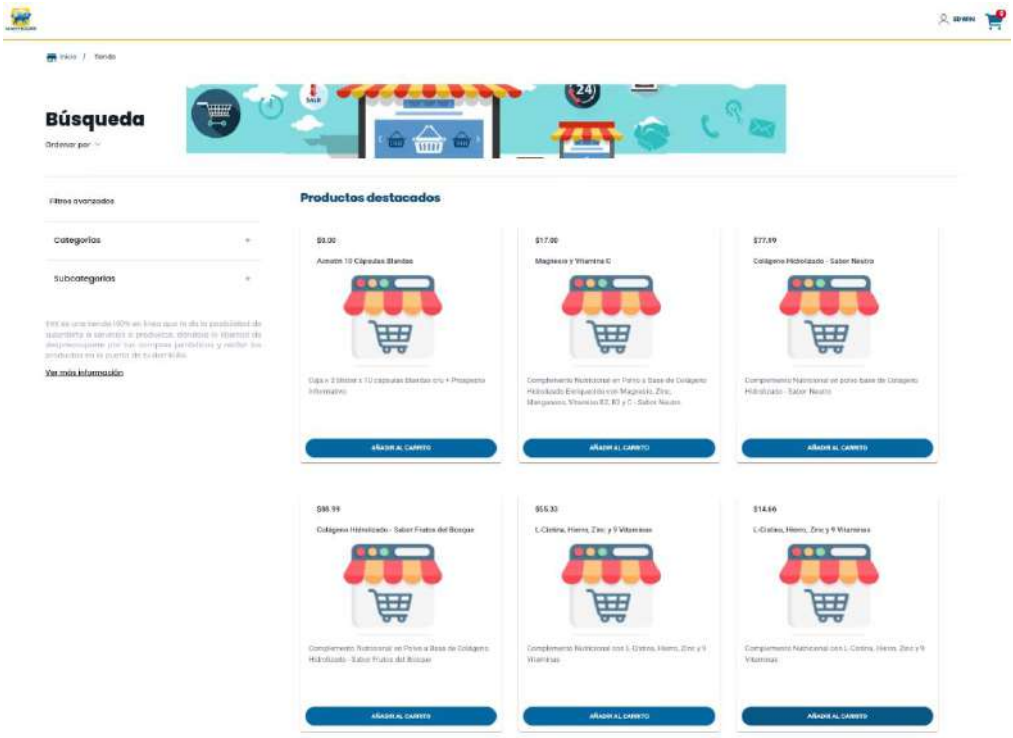


Figura 80: Pantalla inicio aplicativo cliente final

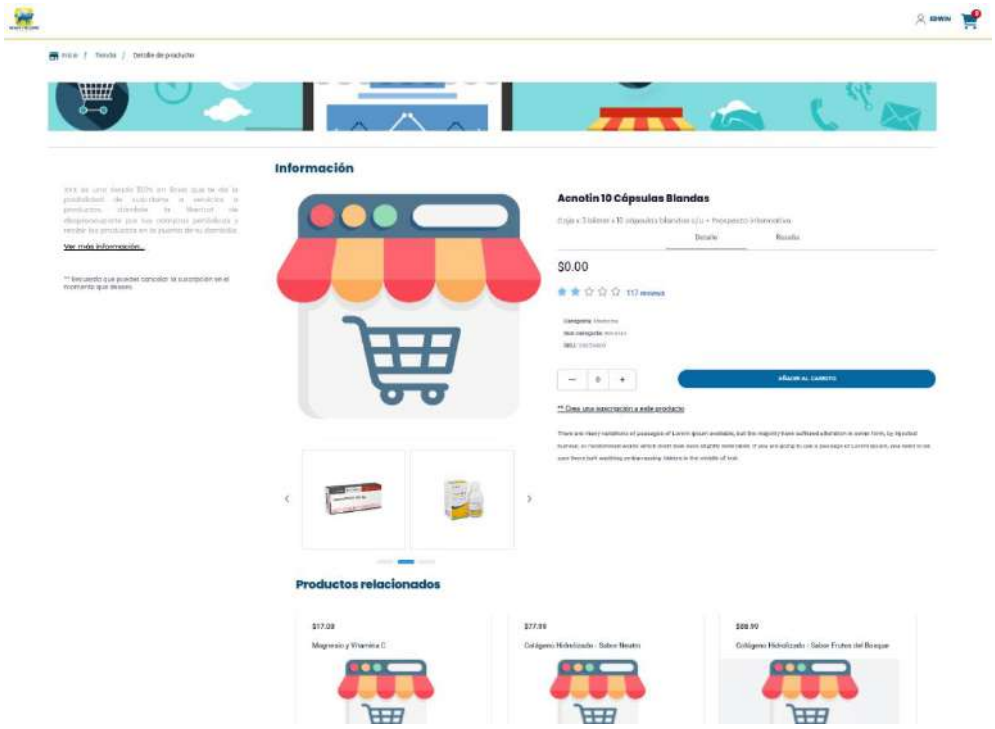


Figura 81: Pantalla descripción de producto, cliente final

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Al realizar un benchmarking a sistemas conocidos como FastFarma, OpenCart y OsCommerce se pudo obtener una idea inicial de los problemas que se quería resolver. Posteriormente, en la entrevista con el PO de Manticore-Labs, se logró definir los requisitos mínimos que debía cumplir la aplicación para solventar las necesidades de la empresa, siendo este el paso inicial que nos permitió el desarrollo de los prototipos y su posterior implementación.
- Se llegó a desarrollar un entorno completo para la gestión de una tienda e-commerce, la cual se comunica con Google Cloud Platform con el objetivo de generar reportes en la nube, haciendo uso de BigQuery y Data Studio.
- El uso de la metodología Scrum ayudó a definir requerimientos de manera óptima, además de priorizar tareas dentro del backlog en cada inicio de sprint. Esto ayudo a llevar un desarrollo ordenado y tener claro los objetivos dentro del equipo; ya que se tuvo una constante comunicación y feedback con el PO de Manticore-Labs; de esta manera se optimizaron tiempos de desarrollo y se llegó a crear un producto de calidad, el cual sea escalable y fácil de mantener.
- Se logró integrar el nuevo submódulo ecommerce dentro del aplicativo principal de la empresa Manticore-Labs sin llegar a impactar en el funcionamiento y flujo de procesos ya existente. Además, se creó una nueva aplicación de frontend que se comunique con el backend antes mencionado y la cual es personalizable para cada empresa que requiera el servicio de una tienda e-commerce.
- Se diseñó una arquitectura escalable, haciendo uso de un stack basado en TypeScript con la finalidad de facilitar el mantenimiento y posibles cambios a futuro, puesto que no sería necesario contratar un desarrollador backend para dar mantenimiento o a un desarrollador frontend para realizar cambios en la parte visual del cliente.
- La estructura de datos que posee Redis nos facilitó la implementación de colas ligeras y persistentes para el manejo de las suscripciones a productos por parte del cliente. El uso de estas colas más la librería BullQueue permitió un desarrollo sencillo, fácil de entender y sobre todo nos ayudó a mejorar en el tiempo de entrega de la tarea. Al ser Redis una estructura de base de datos, utilizando un cliente podemos observar los datos de las suscripciones encoladas y de esta forma identificar posibles errores sin la necesidad de acceder a código.

- La implementación de DevOps a través de “infraestructura como código” permitió ofrecer un entorno estable y escalable de manera rápida para el frontend del cliente final, de esta forma al momento de desplegar nuestra aplicación en un bucket de S3 de AWS. Así no fue necesario acceder a la consola de administración, sino que con el uso de Pulumi y lenguaje TypeScript, se pudo crear todo el ambiente, permitiendo optimizar tiempos en cada despliegue y llevar una trazabilidad de las políticas manejadas en el bucket a través de GIT.
- Se logró implementar la metodología SUS para medirá la usabilidad del sistema a nivel del cliente final (aplicación e-commerce), a través de la cual se obtuvieron excelentes resultados dentro de los usuarios que probaron el sistema. Durante estas pruebas se validó que la navegación sea intuitiva, además de que realizar una compra a modo de suscripción sea lo más transparente para el usuario.
- La librería interna de Manticore-Labs y sus respectivos generadores ayudaron a reducir de manera considerable el tiempo de implementación y el número de posibles errores al momento de integrar el nuevo submódulo con el aplicativo principal. Otro punto valioso es que esta librería nos acortó el tiempo en programar CRUD’s, por lo cual la mayor parte del tiempo en cada sprint fue utilizada en centrarnos de mejor manera en la lógica de negocio.

4.2. RECOMENDACIONES

- Se recomienda en futuras versiones desarrollar e integrar un módulo para hacer uso de pasarelas de pago dentro de la aplicación, con esto se podría generar mejores ingresos y darle al usuario una mejor experiencia al momento de adquirir sus productos.
- Para el despliegue del sistema de Manticore Labs en producción, se sugiere migrar los CI/CD existentes, a uno basado en IaC, ya que esto nos facilitará el manejo de despliegues por parte de los desarrolladores además de gestionar todo el ecosistema de servidores utilizando código escrito en TypeScript mediante la herramienta Pulumi.
- Para obtener métricas del uso de la aplicación e-commerce, es recomendable configurar y utilizar herramientas como Hotjar, la cual nos permita recopilar información para evaluar el comportamiento del usuario dentro de nuestra aplicación. Con esto se puede obtener una trazabilidad de las acciones realizadas por el usuario, y de esta forma poder generar actualizaciones que vayan acoplándose al uso cotidiano de un usuario.

- Para agregar nuevas opciones dentro del aplicativo e-commerce, es recomendable agregar Google Tag Manager, ya que esta herramienta va a permitir al usuario agregar fragmentos de código, como por ejemplo un botón de WhatsApp sin necesidad de pedir a un desarrollador que modifique el código y realice un nuevo despliegue a producción del aplicativo.

5. REFERENCIONAS BIBLIOGRÁFICAS

[1] J. F. Martínez Valverde y F. Rojas Ruiz, Comercio electrónico, Madrid: Ediciones Paraninfo, S.A, 2016.

[2] R. Cáceres Salas, Curso de marketing: Técnicas comerciales aplicadas a las empresas, Barcelona: House Group, 2016.

[3] Breadet Services, «Traditional Reporting vs Business Intelligence,» 28 Noviembre 2019. [En línea]. Available: https://bredetservices.com/blog/traditional-reporting-vs-business-intelligence/. [Último acceso: 4 Abril 2021].

[4] Phocas Software, «Phocas,» 26 Junio 2019. [En línea]. Available: https://www.phocassoftware.com/business-intelligence-blog/how-to-overcome-reporting-challenges-in-your-business. [Último acceso: 4 Abril 2020].

[5] beServices, «beServices,» 9 Abril 2019. [En línea]. Available: https://www.beservices.es/beneficios-google-cloud-platform-n-5370-es. [Último acceso: 4 Abril 2020].

[6] E. Ruiz Larrocha, Nuevas Tendencias en los Sistemas de Información, Madrid: Centro de Estudios Ramón Areces S.A., 2017.

[7] R. Guerrero Cuéllar y L. A. Rivas Tovar, «Comercio Electrónico en Méxio: Propuesta de un modelo conceptual aplicado a las PyMEs,» *Revista Internacion de Ciencias Sociales y Humanidades, SOCIOTAM,* vol. 1, p. 38, 2005.

[8] «Debitoor,» E-Commerce, [En línea]. Available: https://debitoor.es/glosario/definicion-ecommerce. [Último acceso: 26 Abril 2020].

[9] M. Ramos, «Marketing E-Commerce MX,» Marketing, 2 Junio 2020. [En línea]. Available: https://marketing4ecommerce.mx/que-es-el-ecommerce/. [Último acceso: 26 Abril 2020].

[10] J. C. Cortio Pérez, «Barins SINS,» El modelo de suscripció en E-Commerce, 25 Mayo 2019. [En línea]. Available: https://www.brainsins.com/es/blog/suscripcion-ecommerce/6869. [Último acceso: 26 Abril 2020].

[11] R. Caivano y L. Villoria, Aplicaciones Web 2.0, Sevilla: Eduvim, 2009.

[12] S. Lujan Mora, Programación en Internet: Clientes Web, Alicante: Club Universitario, 2001.

[13] C. Casado Iglesias, Entornos de Desarrollo, Madrid: RA-MA, S.A, 2006.

[14] D. d. P. Gallegos y M. Espinoza, «Benchmarking, ¿cómo y de dónde?: una revisión sistemática de la literatura,» *ESPACIOS,* vol. I, nº 1, p. 16, 2019.

[15] J. Resnik, «TiendaNube Blog,» ¿Qués es benchmarking y como aplicarlo en el e-commerce?, Septiembre 2020. [En línea]. Available:

https://www.tiendanube.com/blog/benchmarking/#:~:text=Benchmarking%20es%20el%20concepto%20que,por%20ejemplo%2C%20la%20competencia).. [Último acceso: 28 Abril 2020].

[16] OpenCart, «OpenCart Documentation,» [En línea]. Available: http://docs.opencart.com/en-gb/introduction/. [Último acceso: 12 Mayo 2022].

[17] OsCommerce, «OsCommerce,» [En línea]. Available: https://www.oscommerce.com/. [Último acceso: 12 Mayo 2022].

[18] Microsoft, «Azure,» [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-devops/. [Último acceso: 4 Mayo 2021].

[19] Mora P y J. J. Mora Pérez, DevOps y el camino de baldosas amarillas, Buenos Aires: Creative Commins, 2015.

[20] E. Kaim, «Docs Microsoft - Azure DevOps,» 29 Junio 2021. [En línea]. Available: https://docs.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code. [Último acceso: 12 Julio 2022].

[21] B. Straub y S. Chacon, Pro GIT, Apress, 2014.

[22] Google, «Google Cloud Platform,» [En línea]. Available: https://cloud.google.com/docs/overview?hl=es. [Último acceso: 26 Abril 2020].

[23] Servi Información, «Servi-Información, localización inteligente,» 27 Agosto 2016. [En línea]. Available: https://servinformacion.com/g-cloud-que-es-bigquery/. [Último acceso: 27 Abril 2021].

[24] IPNET Growth Partner, «Blog IPNET Growth Partner,» 6 Agosto 2020. [En línea]. Available: https://es.blog.ipnet.cloud/bigquery/. [Último acceso: 27 Abril 2021].

[25] K. A, Agile mothodology for Developing & Measuring Learning, Bloomington: AuthorHouse LLC, 2013.

[26] incubic Ingeniería Industrial Innovación, Metodología Agile y Scrum, incubic , 2018.

[27] R. E. López Menéndez, «Metodologías Ágiles de Desarrollo de Software Aplicadas a la,» *Revista Tecnológica ITCA-FEPADE,* vol. I, pp. 6-11, 2015.

[28] P. Subra y A. Vannieuwenhuyze, Scrum, Un método ágl para sus proyectos, ENI, 2018.

[29] J. Sutherland y K. Schwaber, La Guía Scrum: Reglas del Juego, Creative Commons, 2020.

[30] A. Peralta, Metodología SCRUM, Universidad ORT, Uruguay, 2013.

[31] C. Alves y J. Bach, TypeScript para principiantes, Madrid: Independiente, 2021.

[32] L. Pucciarelli, NodeJs: instalción, arquitectura, node y npm, Buenos Aires: USERS ebooks, 2020.

[33] J. Bell, G. Mangolan, P. Housley y A. De Peretti, NestJs> A pregressive NodeJs framework, California: Bleeding Edge Press, 2018.

[34] Quality Devs, «¿Que es Angular y para que sirve?,» 16 Septiembre 2019. [En línea]. Available: https://www.qualitydevs.com/2019/09/16/que-es-angular-y-para-que-sirve/. [Último acceso: 26 Abril 2021].

[35] P. Huet, «OpenWebinars,» 21 Enero 2022. [En línea]. Available: https://openwebinars.net/blog/que-es-tailwind-css-y-por-que-deberias-usarlo/. [Último acceso: 12 Julio 2022].

[36] LinkedIn, GitLab esencial, linkendin, 2016.

[37] M. Lassoff y S. eLearning, Figma for Digital Product Design, California: Stone Riber eLearning, 2020.

[38] N. Rozentals, Mastering TypeScript 3, Birmingham: Packt Publishing Ltd., 2019.

[39] K. Schwaber y J. Sutherland, The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Gam, Creative Commons, 2020.

[40] X. Ganzábal García, Aplicaciones técnicas de usabilidad y accesibilidad en el entorno cliente, Madrid: Ediciones Paraninfo, SA, 2015.

[41] usability.gov, «Usability Gov,» [En línea]. Available: https://www.usability.gov/. [Último acceso: 2022 08 17].

[42] J. Sauro, «MeasuringU,» 03 Febrero 2011. [En línea]. Available: https://measuringu.com/sus/. [Último acceso: 2022 Agosto 18].

[43] R. Guerrero y L. Rivas Tovar, «Comercio Electrónico en M,» *Revista Internacional de Ciencias Sociales y Humanidades, SOCIOTAM,* 2005.

[44] D. Solis Fonseca, W. Roque Pérez y L. Morilla Faurés, «Pasarela de Pagos para la Seguridad de Transacciones Bancarias en Linea,» *3Ciencias,* vol. I, p. 2, 2013.

[45] R. Cárdenas, Manual de Buenas Prácticas de Pasarela de Pago, Bogota: Camara de Comercio Electrónico de Colombia, 2018.

6. ANEXOS

6.1. **Repositorio de submódulos y aplicativo para el cliente final**

https://gitlab.com/manticore-labs/trabajo/proyectos/submodulos/submodulo-ecommerce

6.2. **Mockups de aplicación del cliente final**

https://www.figma.com/file/RH3d8ZrhBnwcoV6C0TuZzf/Wireframe---Tesis

6.3. **Sitemap de la aplicación**

https://drive.google.com/file/d/1_ptdrqqnxWQWC7C0NlDoiY29HBiB9Ylp/view?usp=sharing

6.4. **Esquema de base de datos**

https://drive.google.com/file/d/1309qqkzVzNDSCYDi7FNsYE5rnBr8fmFq/view?usp=sharing

6.5. **Gráficos complementarios y estándares de Manticore Labs**

https://lucid.app/lucidchart/4c8a903f-9753-4208-afa8-1669b4daf29c/edit?viewport_loc=-22%2C-589%2C3064%2C3443%2C0_0&invitationId=inv_f4af75d9-72d1-47e5-be4f-ef38748b3083#

6.6. **Historias de usuario**

https://docs.google.com/document/d/1nvUBf8lVaRUs3LRm4jHdEMIyhpgWaTDv/edit?usp=sharing&ouid=104677146737705087303&rtpof=true&sd=true

6.7. **Encuesta de usabilidad del sistema**

https://forms.gle/yZagAWP1WaMh7UGT7

6.8. **Reporte en Google Data Studio**

https://lookerstudio.google.com/reporting/4d6572bb-7c64-4e99-85f8-7d48fed7d52c