ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA WEB DE GESTIÓN DE PEDIDOS EN MINIMARKETS

DESARROLLO DE FRONTEND

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE

MARTÍN ANDRES ROSERO BONILLA

DIRECTOR: JUAN PABLO ZALDUMBIDE PROAÑO

DMQ, Agosto 2023

CERTIFICACIONES

Yo, Martín Andres Rosero Bonilla declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Martin Andrés Rosero Bonilla

Martin.rosero@epn.edu.ec

mrosero.sfq@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Martin Andrés Rosero Bonilla, bajo mi supervisión.

Ing. JUAN PABLO ZALDUMBIDE PROAÑO

Juan.zaldumbide@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

MARTÍN ANDRES ROSERO BONILLA

DEDICATORIA

Quiero dedicar este proyecto, la finalización y demostración de todos los conocimientos adquiridos en esta carrera a la Escuela Politécnica Nacional, Escuela de Formación de Tecnólogos y cada uno de los ingenieros quienes con su paciencia y dedicación a la enseñanza transmitieron su conocimiento para forjarme como un profesional dedicado, honesto y responsable.

Para empezar, mencionando al Ing. Juan Pablo Zaldumbide quien fue una guía total para la realización del proyecto, al Ing. Byron Loarte un excelente profesor que no solo se preocupó por enseñar la materia, sino también mostro un genuino interés en que cada estudiante este preparado para la vida profesional, la Ing. Luz Vintimilla quien me presentó varios retos en sus materias y tuvo la paciencia suficiente para que yo pudiera superarlos, a la Ing. Mayra Álvarez la cual encontró la manera perfecta de mantener el interés en la materia y prepararse para cada evaluación, a la Ing. Vanessa Guevara una excelente maestra con una metodología de enseñanza y evaluación que superaron completamente mis expectativas.

Finalmente, yo no estaría en esta carrera y no habría realizado este proyecto si no fuese por las licenciadas: Myrian Asencio y Patricia Cumbicos quienes fueron las encargadas de mostrarme e instruirme por primera vez en el mundo de la informática y programación, también al Licenciado Luis Chamorro cuyas últimas palabras en mi graduación de colegio fueron “siga estudiando y preparándose señor Rosero”.

AGRADECIMIENTO

Tengo una cantidad exagerada de personas a las que agradecer; mentores, personas que me entretuvieron mucho tiempo, amigos, personajes de ficción y familia.

Me gustaría empezar por mi mayor mentor y a quien le debo la persona que soy, Dalas la persona que me enseñó a ir en contra del instinto de rebaño, a que, si te sientes mal pues no lo estes, que cualquier tipo de alcohol, droga, religión y novia toxica solo generan problemas que no valen la pena, aun me queda mucho que aprender de él, pero seguiré esforzándome para ser la mejor versión de mí. Muchas gracias.

“Da igual donde vivas da igual donde estes, si estas alegre o si tienes estrés, tu familia te espera en este canal” no entendí el significado de esas palabras si no hasta mucho tiempo después, Gona89 siempre la recordare como la persona que siempre me puede sacar una risa en el peor momento y su canal como un lugar lleno de series, juegos, historias y personas que siempre llevare conmigo; Ben and Ed, *Randal’s Monday*, *Skywars*, *Murder Mystery, Stranded Deep, lucky block*s. Muchas gracias.

La gente siempre suele usar la palabra “amigo” muy a la ligera, para mi se trata de alguien totalmente cercano, alguien con quien puedes contar, con quien no la pasas mal, a día de hoy cuento con alguien así, alguien a quien a día de hoy puedo llamar “hermano”, Andy a pesar de no ser la mejor persona del mundo espero que sigamos troleando en lol, muriendo en el *elden ring*, construyendo en minecraft, viendo subir y bajar los *stats* en Vallheim, espero vivir muchas más aventuras juntos. Muchas gracias.

Si tuviera que nombrar a todos los personajes de ficción que me ayudaron a ser quien soy no acabo este agradecimiento así que me limitare a mencionar sus aportaciones, “Camina hacia el futuro”, “El futuro no está escrito, el futuro es el que tú te forjes”, “Piensa, piensa, piensa… Idea”, “Vive para luchar otro día”, “Haría esto todo el día”, “Un gran poder con lleva una gran responsabilidad”, “Mientras me quede un soplo de vida, mi destino está en mis manos”, “Mientras la vanguardia valerosa permanezca en pie Demacia jamás caerá”.

Mi familia quizás no sea la mejor pero sí que me enorgullece el cómo es, el nivel de confianza, comunicación y pasatiempos que tenemos juntos es simplemente envidiable. Primero me gustaría hablar un poco de mi hermana mayor, Sofia, ella desde que tengo memoria a estado dispuesta a cuidarme, quizás demasiado, sin embargo, los momentos en la escuela jugando juntos, yendo a casa caminando para comer un pan con el dinero del pasaje son momentos que no olvidare, incluso todos los momentos y cosas que hemos

jugado y visto, la saga de Metro, Marvel, Star Wars, SAO, Five Nighs at Fredys, me encantaron compartir todos estos productos contigo, yo estaré para ti cuando lo necesites al igual que tú lo estuviste para mí.

A mi hermana menor pues no la conozco lo suficiente, a veces estas feliz a veces triste o enojada, a ti solo espero que llegado el momento me sienta orgulloso de quién eres, sé que no me defraudaras, así como yo no te defraudare, te guiare por el mejor camino para que puedas tener un futuro mejor.

Ahora como lo mejor se deja para el final pues esta es tu parte, mamá. Las cosas que has hecho por mí no han tenido nombre, sacrificándote cada día trabajando para poder darme una educación, comida, ropa y casa. Desde pequeño siempre supe que estabas ahí, cada día llevándome a la guardería, día de foto, día de festejo, en el fondo sabía por qué no podías estar y quiero decirte que nunca guarde ningún resentimiento porque no estuviste, sé que era lo que debías hacer y te lo agradezco. Cada día en el que comemos, hablamos, jugamos, vemos algo, me siento totalmente en calma. Incluso en este gran proceso que fue realizar este proyecto, al igual que Gohan sentía como su padre le ayudo a derrotar a Cell, siento como tú me das la fuerza necesaria para no rendirme y juro que no lo hare, aun si caigo me levantare porque es algo que me has enseñado, prometo que te regresare todos tus esfuerzos porque sé que aun en mi peor momento y si llego a desviarme en algún punto del camino solo tu podrás regresarme a ser la persona que soy, te quiero mamá y estarás orgullosa de mí, lo prometo.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El siguiente trabajo de integración curricular tiene como objetivo desarrollar y documentar, mediante la metodología SCRUM y la herramienta React, un *frontend* para un gestor de tienda online y pedidos enfocado en *minimarkets*. El objetivo es proporcionar una interfaz de usuario intuitiva y agradable tanto para los usuarios como para los administradores, y que pueda consumir el *backend* diseñado específicamente para este sistema.

Con el propósito de llevar a cabo un desarrollo de software enfocado y organizado, se utilizaron requerimientos e historias de usuario, y se realizarán seis sprints para asegurar que el desarrollo no se vea comprometido por falta de orden o comunicación.

Todo software requiere una arquitectura, y en este caso, se hizo uso de la arquitectura Modelo-Vista-VistaModelo (MVVM). Esta arquitectura se enfoca en identificar las vistas, que serán responsables de mostrar la parte visual del sistema; la vista modelo, que contendrá la lógica por parte del *frontend*; y el modelo, que se refiere a la interfaz de programación de aplicaciones (API).

El *framework* principal utilizado para el desarrollo es React. En él, se han dividido las secciones en componentes reutilizables y no reutilizables. Estos componentes pueden ser invocados por otros componentes para formar las vistas, y mediante el uso de funciones JavaScript XML (JSX), se logra realizar la vista modelo. Finalmente, para las peticiones hacia el modelo, se ha utilizado la biblioteca Axios.

PALABRAS CLAVE: React, Minimarkets, Scrum, Frontend

ABSTRACT

The following curricular integration work aims to develop and document a frontend for an online store and order management system focused on minimarkets, using the SCRUM methodology and React tool. The goal is to provide an intuitive and user-friendly interface for both customers and administrators, capable of consuming the specifically designed backend for this system.

To ensure a focused and organized software development, requirements and user stories will be used, and six sprints will be conducted to avoid compromising the development due to lack of order or communication.

Every software requires an architecture, and in this case, the Model-View-ViewModel (MVVM) architecture has been employed. This architecture focuses on identifying the views, responsible for displaying the visual part of the system; the view model, containing the frontend logic; and the model, referring to the Application Programming Interface (API).

The primary framework used for development is React. It involves dividing the sections into reusable and non-reusable components, which can be invoked by other components to form the views. The view model is achieved through the use of JavaScript XML (JSX) functions. Finally, the Axios library is utilized for making requests to the model.

KEYWORDS: React, Minimarkets, Scrum, Frontend

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

En la actualidad, en la ciudad de Quito contamos con una gran cantidad de *minimarkets* los cuales llegado el momento van a requerir de solución tecnológica para la gestión de pedidos *online*, muchos de estos solo cuentan con los paquetes de office, que, aunque no están mal, si generan problemas de eficiencia e integridad a la hora de manjar los datos de los pedidos, estos problemas pueden ser solucionados gracias al uso de la tecnología, a día de hoy muchas personas cuentan con un dispositivo móvil o de escritorio con conexión a internet y un navegador, así que se usaran estos recursos para realizar la solución respectiva.

Se propone solucionar esta problemática con un sistema que automatice el proceso de creación y seguimiento de pedidos de la mejor manera para la experiencia de los usuarios, ya sean estos los clientes o los administradores, específicamente este componente se encarga del desarrollo del *frontend* para este sistema.

Para poder interactuar con un sistema de manera simple, agradable e intuitiva, se cuenta con un diseño de *frontend* que cumpla con todos requisitos mencionados, así como también lograr consumir el *backend* del sistema de la manera más optima y eficiente teniendo en cuenta la experiencia de usuario.

Se tiene que contar con un levantamiento de requerimientos respectivos para poder realizar un prototipado adecuado, el cual será la idea principal para poder desarrollar la estructura grafica lo más apegado a el diseño inicial, sin descartar los posibles cambios que pueden surgir a la hora de codificar.

El diseño de este *front* se basa en el tipo minimalista y estéticamente limpio, sin recargar de información la pantalla, priorizar el manejo intuitivo del sistema para todos los perfiles disponibles, así como también iconos identificables por su funcionamiento, cada campo con su respectiva validación, tomando en cuenta los parámetros básicos de seguridad.

En pocas palabras, el componente presente se trata en lograr desarrollar un *frontend* adecuado que cumpla con los estándares de calidad y de experiencia de usuario centrado en resolver la problemática de la falta de una solución tecnológica para poder controlar y gestionar los pedidos de los clientes contando con un perfil de administrador el cual contara con los permisos de realizar distintos cambios dentro del sistema haciendo uso del *frontend* para consumir el *backend* del mismo.

1.1 Objetivo general

Desarrollar un *frontend* para el sistema web de gestión de pedidos en *minimarkets*.

1.2 Objetivos específicos

Levantar los requerimientos para el componente del *frontend*.

Diseñar las interfaces de usuario intuitiva y fácil de usar.

Codificar los diseños con la herramienta react.

Comprobar las pruebas de los requisitos establecidos.

Desplegar el componente en una plataforma adecuada para este *frontend*.

1.3 Alcance

El presente componente tiene como intención el desarrollo e implementación de un*frontend* [1] para el sistema web de compra de productos, gestión de pedidos, gestión de productos y gestión de cuentas en *minimarkets*, además de poder consumir el *backend* [1]desarrollado para este proyecto.

Los módulos principales son: el catálogo de productos con los que un cliente pueda interactuar con su carrito de compras, las vistas para poder controlar los CRUD [2] de productos, perfiles de administradores, perfiles de empleados y pedidos.

Para esto se hace uso de herramientas tales como: React [3] principal *framework* para codificar, JSX [3] el cual permite combinar archivos HTML (*HyperText Markup Language*) [4], CSS (*Cascading Style Sheets*) [5] y JavaScript [6] y hacerlos código simple, fácil de leer y reutilizable.

Además de usar varios tipos de recursos disponibles para agilizar el desarrollo del componente, tales como: *axios* [7] y MDB [8].

Manejando durante todo el ciclo de vida del software una metodología SCRUM [9], realizando actividades semanales y dentro de estas semanas pequeñas actividades que ayuden a avanzar en el desarrollo del componente.

1.4 Marco Teórico

La investigación previa ayuda a establecer bases y definiciones de conceptos, ideas, metodologías y procedimientos, los cuales son plasmados, formando de esta manera el marco teórico [10] evidenciando que se ha basado en conceptos sostenidos por una investigación y documentación científica de fuentes confiables [11].

Frontend

Se refiere a la parte del desarrollo de software donde se enfoca en la parte visual de los sistemas, contando con la estructura, estilos, colores, fondos, tamaños, animaciones y efectos. [1]

Prototipado

Es una versión de referencia del diseño del producto que se pretende realizar, es decir: una vista precia del funcionamiento y la estética del sistema. [12]

Metodologías de desarrollo de software

Son las técnicas y métodos a seguir en el proceso del ciclo de vida de un *software* [13]. Estas son indispensables para un correcto manejo del trabajo, generando organización en todos los miembros del equipo y el tiempo empleado. [14]

Metodologías ágiles

Se tratan de un tipo de metodologías de desarrollo de software que se caracterizan por ofrecer un producto con una entrega incremental, es decir, hace entrega de partes de funcionalidad en cortos periodos de tiempo. [14]

Componente

En el contexto de react, un componente es la unión de html, css y javascript [4], los cuales unidos tienen como objetivo ser “elementos reutilizables de UI para tu aplicación” [15]

Teniendo en cuenta estos conceptos el componente presente se fundamentará en una metodología de desarrollo de software ágil para realizar un respectivo prototipado del sistema de pedidos en los *minimarkets* de la ciudad de Quito, basado en esta versión del prototipado para poder realizar un correcto *frontend*.

2 METODOLOGÍA

El uso de una metodología dentro del desarrollo de software es indispensable, ya que esta nos permite llevar a cabo un orden de tiempo, tareas, pruebas y documentación [14]. Dado a que este componente tiene un estimado de 10 semanas para su finalización se hace uso de la metodología ágil “SCRUM” [9], se toma cada semana como un sprint de la metodología, las cuales llevan tareas asignadas dentro del plazo de tiempo estimado, para poder realizar la codificación, pruebas, corrección y documentación respectiva de dicho sprint.

2.1 Scrum

La metodología solicita la división de los roles que se aplican en la misma, así como: Recopilación de requerimientos, historias de usuario, producto backlog y sprint backlog (Artefactos) [9].

Roles

Los roles son los puestos críticos de la metodología, los cuales se dividen en: “product owner”, “scrum master” y “development team”, estos hacen referencia a: el principal encargado en velar por un producto de calidad y en los intereses del cliente; persona especializada en la metodología y encargada de ver que los miembros del equipo sigan la misma de la mejor manera; encargados de la realización de los objetivos de los sprints y dar un entregable al final de estos [16].

La división de estos puestos para el desarrollo del presente componente se da a conocer en la Tabla 1.

Tabla 1 Roles de SCRUM

ROLES	NOMBRES
Product Owner	Martín Rosero
Scrum Master	Martín Rosero, Ing. Juan Pablo Zaldumbide
Development Team	Martín Rosero

Fuente: Elaboración propia.

Artefactos

Los artefactos para toda metodología son las partes elementales para cada una de ellas, en el contexto de SCRUM ayudan a mejorar la organización, comunicación y priorizar tareas. Estos se artefactos se dividen en 4: Recopilación de requerimientos, para ayudarnos a tener las ideas principales de que es lo que debe tener el componente final, ejemplo en la tabla 2; Historias de usuario, técnica para describir cada funcionalidad y su riesgo en desarrollo, ejemplo en la tabla 3; *Product Backlog*, es la lista dinámica de requerimientos y funcionalidades del producto final, ejemplo en la tabla 4; *Sprint Backlog*, se trata de la lista detallada de cada tarea asignada a cada sprint a realizar en un tiempo determinado, ejemplo en la tabla 5 [17].

Tabla 2 Requerimientos 1 - Interfaz de inicio de sesión

LEVANTAMIENTO DE REQUERIMIENTOS		
ID - RQ	REQUERIMIENTO	DESCRIPCION
RQ – 01	Interfaz de inicio de sesión	Formulario intuitivo y agradable para todo usuario que necesite ingresar a su cuenta

Fuente: Elaboración propia.

Tabla 3 Historia de Usuario 1 - Barra de navegación

HISTORIA DE USUARIO	
Identificador (ID): HU-01	**Usuario:** Todo usuario
Nombre Historia: Barra de navegación	
Prioridad: Alta	**Riesgo en Desarrollo:** Baja
Iteración: 1	
Responsable: Martin Rosero	
Descripción: Componente indispensable para la navegación por el sistema	
Observación: estará disponible en todas las vistas	

Fuente: Elaboración propia.

Tabla 4 *Product backlog* 1 - Interfaz de inicio de sesión

PRODUCT BACKLOG				
ID - HU	**HISTORIA DE USUARIO**	**SPRINT**	**PRIORIDAD**	**ESTADO**
HU-01	Interfaz de Inicio de sesión	1	Alta	En desarrollo

Fuente: Elaboración propia.

Tabla 5 *Sprint backlog* 0 - Configuración del ambiente de desarrollo

SPRINT BACKLOG					
ID - SB	**NOMBRE**	**ID - HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB000	Configuración del ambiente de desarrollo	-----	-----	Diseñar un prototipado Instalación de herramientas de desarrollo. Creación del proyecto Creación de repositorio en github	25H

Fuente: Elaboración propia.

2.2 Diseño de interfaces

Partir de un diseño inicial es una parte importante a la hora del desarrollo *frontend* en aplicaciones web, ya que nos sirven como guía para poder irla mejorando y adaptando a las necesidades del sistema [18].

Herramienta utilizada para el diseño

La herramienta Figma es una de las principales en el desarrollo de prototipado ya que no tiene servicios exclusivos de paga y ayuda también a manejar una idea inicial de la navegación entre vistas [19].

Sistema Web

Para empezar, se tiene que diseñar la identidad del sistema, esto se refiere a los colores principales, logo, nombres y tipografía. Una vez identificados estos detalles se procede a diseñar las diferentes vistas, a continuación, la figura 2.1 muestra el diseño de la vista de registro donde tenemos la posibilidad de crear cuentas con redes sociales o por datos personales y una opción para ir a la vista de *login,* cabe recalcar que este por ser un prototipado este sujeto a cambios en el desarrollo [19].

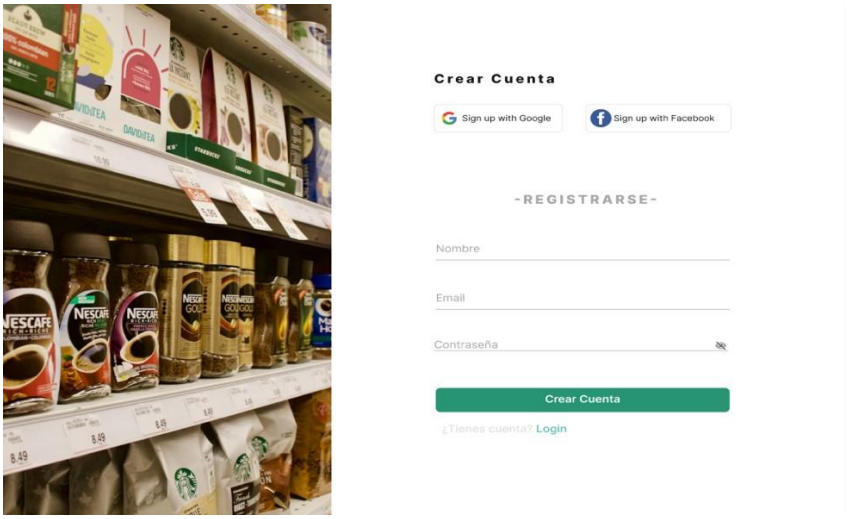

Figura 2.1 Prototipo - Formulario de creación de cuenta

2.3 Diseño de la arquitectura

Contar con una arquitectura para el desarrollo *fontend* es una buena práctica para todo desarrollador de software, ya que ayuda a mejorar la calidad del producto y plantear una base sólida sobre la cual poder trabajar [20].

Para este componente es indispensable identificar el tipo de sistema que se utiliza, así como sus herramientas y componentes. Al ser un sistema web la principal forma de interactuar con el usuario será a través de un navegador y peticiones a un *backend* en la nube, tomando en cuenta estos aspectos se puede llegar a la arquitectura de datos adecuada.

Arquitectura de Datos

El componente por las características que posee requiere de una arquitectura modelo, vista, vistamodelo (MVVM) [21], ya que este posee la capacidad de comunicar el *frontend* con las respectiva API que necesite a través de peticiones HTTP.

Las vistas producidas en React se encargan de ayudar al usuario a entender el sistema de manera agradable y simple, las vistas modelos son las intermediarias entre la lógica del sistema y las vistas comunes ya que se encargan de enviar los datos al modelo, así como de la lógica de exposición a la vista, en cambio el modelo será el que maneje los datos enviados por las vistas modelos que las controlará el usuario por medio de las vistas [21], esto se ve reflejado en la figura 2.2.

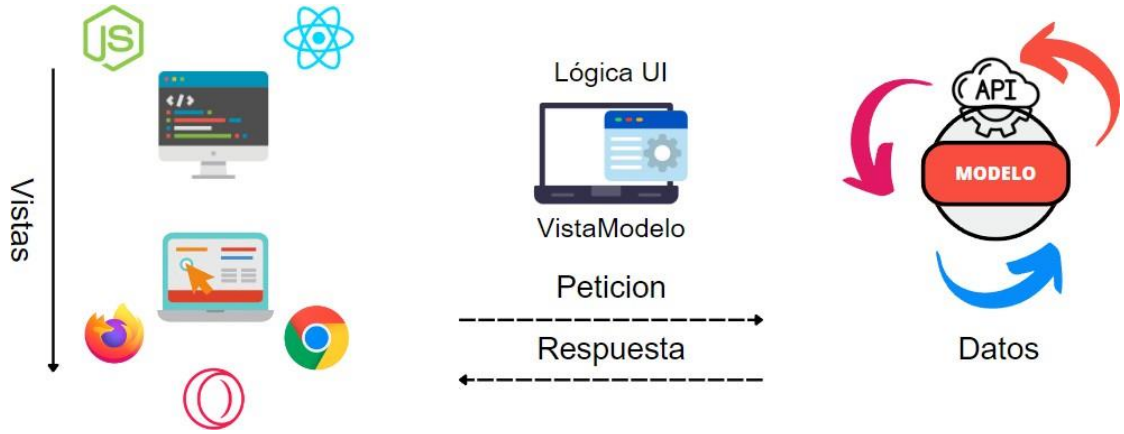

Figura 2.2 Arquitectura del proyecto

2.4 Herramientas de desarrollo

Con el tiempo se ha desarrollado una gran cantidad de herramientas que facilitan el desarrollo de software, para este componente se hace uso de una gran variedad de estas, para tener controlador de versiones, entorno de desarrollo, despliegue y componentes reutilizables [22], las herramientas utilizadas en el desarrollo se encuentran en la tabla 6.

Tabla 6 Herramientas para el desarrollo del componente

Herramienta	Justificación
Visual Studio Code	Principal editor de texto para el desarrollo de software con *frameworks* como react por su adaptabilidad y extensiones [22].
React	Es una biblioteca de JS de código abierto que se enfoca en brindar ayuda en el desarrollo *frontend*, al dividir en componentes de código reutilizable [3].
Netlify	Herramienta de despliegue de aplicaciones web estáticas, sirve de apoyo para poder desplegar avances en línea [23].
Axios	Cliente para *end points* HTTP con ayuda de promesas para su funcionamiento [7].
Node.js	Proporciona un entorno de ejecución en tiempo real, ideal para un proyecto en react que requiere dependencias, servidor y ejecución de código javascript [22].
Bootstrap	Códigos reutilizables de componentes que se pueden usar en aplicativos web y móviles por su propiedad responsive [8].
GitHub	Controlador de versiones para el componente, además de poder respaldar los avances en la nube [22].
HTML – CSS - JS	Estas herramientas son consideradas como una sola debida a que siempre tienen que estar en sinergia para una buena estructura, presentación y funcionalidad [3].
MDBootstrap	Es un recuso online donde se puede encontrar varios componentes react de tipo plantilla para poder utilizar libremente, la cual se basa principalmente en boostraps [8].

3 RESULTADOS

Sprint 0. Configuración del ambiente de desarrollo

Para el sprint 0 no se tienen contempladas historias de usuario, ya que se trata de preparar todos los recursos que se utilizaran a lo largo del desarrollo del componente con un tiempo estimado de 25 horas. Esta iteración cuenta con una serie de tareas, las cuales se observan en la tabla 7.

Tabla 7 Sprint 0 - Configuración del ambiente de desarrollo

SPRINT SB000					
NOMBRE	**MÓDULO**	**ID - HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
Configuración del ambiente de desarrollo	-----	-----	-----	Instalación de herramientas de desarrollo. Creación del proyecto Creación de repositorio en github	25H

Configuración del ambiente de desarrollo.

Como editor de texto la mejor opción es *Visual Studio Code* ya que cuenta con una interfaz simple y fácil de usar, además de tener una cantidad de extensiones que ayuda a agilizar el proceso de desarrollo [24], tales como: Emmet JSS, *React Native Snipets*, *Format* HTML-CSS-JS.

Ahora que ya está configurado el editor de texto se requiere una herramienta que sirva de entorno de ejecución para javascript, el más popular es Node.js, la cual también funciona como biblioteca de js para el desarrollo *frontend*.

Con las herramientas instaladas se crea el proyecto a partir de *créate react app* [25] la cual proporciona la estructura inicial adecuada para el componente y se puede observar cómo está estructurado en la Figura 3.1.

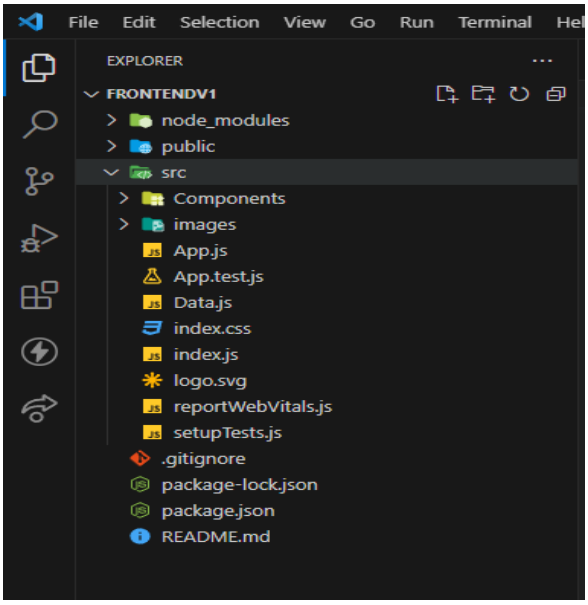

Figura 3.1 Directorio raíz del proyecto

Para finalizar con este sprint se crea el repositorio en github Figura 3.2, el cual será el principal controlador de versiones para el desarrollo del componente, además es donde se albergarán todas las actualizaciones y avances en el mismo.

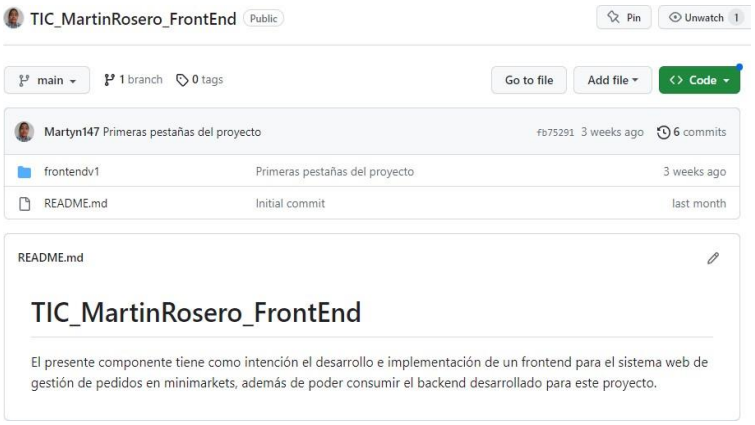

Figura 3.2 Repositorio en Github

Sprint 1. Navegación y autenticación

Barra de navegación.

La barra de navegación es un componente indispensable para que el usuario pueda navegar por la aplicación, esta se debe adaptar a la situación, ya sea de manera *responsive* convirtiéndose en un menú de tipo “hamburguesa”, desaparecer en el menú de registro e inicio de sesión y ocultar opciones según los privilegios de usuario, el diseño de este componente está en la Figura 3.3.

Figura 3.3 Barra de navegación de la página web

Este componente es creado con la ayuda de MDB el cual proporciona elementos pre creados y listos para ser utilizados, además de ser personalizado con la ayuda de etiquetas html que se pueden usar en los archivos jsx.

Este componente también se adapta según el tipo de usuario que este en el sistema limitando las opciones a según el rol del usuario y sus funciones, es decir: un *superadmin* solo debe de poder tener las opciones de salir y agregar administradores, un administrador debe ver la opción de administrar cuentas y productos, un empleado debe de poder administrar los pedidos y finalmente un cliente debe ver el catálogo, su perfil, su carrito de compras y la opción de salir.

Para poder llevar a cabo la contabilidad de ítems en el carrito de compras cuenta con un evento para que actualice el número de ítems en el carrito realizando una petición *get* para poder comprobar cuantos ítems tiene el carrito del usuario, esto se ve en la Figura 3.4.

```
axiosInstance
  .get(`/cart?id=${cartId}`)
  .then((response) => {
    const productsCount = response.data.length;
    setItemTotal(productsCount);
  })
  .catch((error) => {
    console.log(
      "Error al obtener el número de productos del carrito:",
      error
    );
  });
```

Figura 3.4 Ejemplo petición *Get*

Interfaz de inicio de sesión.

La interfaz de inicio de sesión es un formulario de dos campos principales, los cuales solicitaran al usuario su correo y su contraseña para poder acceder al sistema, también cuenta con las opciones para cambiar al formulario de registro y volver a la página principal del sistema.

También cuenta con una respectiva validación de campos, para que el email cumpla con los parámetros estándar para poder serlo, así como también no se pueda enviar en blanco, para la contraseña tenemos el formato de ocultarlo y solicitarlo como obligatorio, tal como se ve en la Figura 3.5.

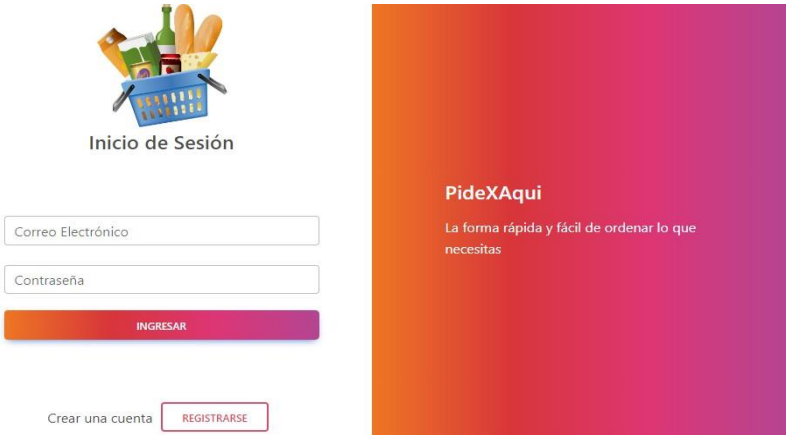

Figura 3.5 Interfaz para el inicio de sesión

Las validaciones de campos se basan en exigir los campos como obligatorios y comprobar que el correo electrónico cumple con los requisitos, estas validaciones se activan al pulsar el botón “INGRESAR”, esto queda evidenciado en la figura 3.6.

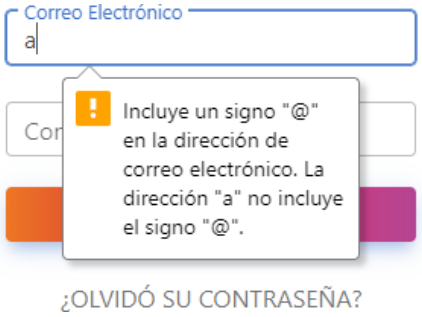

Figura 3.6 Ejemplo de validación de campos

Una vez se complete el formulario con toda la información se envía una petición de tipo *post* con la ayuda de axios, tal como lo podemos ver en la Figura 3.7, el cual valora si la respuesta del servidor es de 200, siendo ese el caso creara una *cookie* “registro” en la que almacenara el token de acceso de tipo *bearer* que nos proporciona la API además de redirigirnos a la página de inicio de la aplicación.

```
try {
  const response = await axios.post(
    "http://127.0.0.1:8000/api/login",
    userData
  );
  if (response.status === 200 && response.data.access_token) {
    // El inicio de sesión fue exitoso y se recibió un token de acceso
    console.log("Inicio de sesión exitoso");

    // Establecer cookie con el token de acceso
    document.cookie = `registro=${response.data.access_token}; path=/`;

    navigate("/"); // Redirigir a la ruta de inicio
```

Figura 3.7 Petición tipo post con axios

Caso contrario si el servidor nos enviase un mensaje de error relacionado a algunos de los campos, este será visualizado en la parte inferior del formulario para que el usuario pueda proporcionar las credenciales correctas, ejemplo de esto en la Figura 3.8.

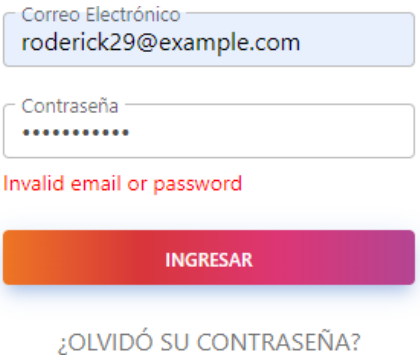

Figura 3.8 Ejemplo de error por información incorrecta

Para finalizar, una vez el ingreso sea exitoso se crearán las *cookies* necesarias para la aplicación, incluyendo el token de acceso proporcionado por el *backend*, tal como lo vemos en la Figura 3.9.

```
// Establecer cookies con informacion necesaria

Cookies.set('token', response.data.access_token, { path: '/' });
Cookies.set('registro', JSON.stringify(response.data));
Cookies.set('idRole', response.data.user_role, { path: '/' });
Cookies.set('id_user', response.data.user_id, { path: '/' });
```

Figura 3.9 Creación de *cookies*

Interfaz de registro de usuario.

Es un formulario con los campos: Nombres, Apellidos, Correo Electrónico, Teléfono, Contraseña y Confirmar Contraseña, para estos se tiene un estricto control para la validación, siendo todos obligatorios y comprobando que el correo cumpla con los parámetros comunes, los campos de contraseña deben de ser iguales para poder enviar el formulario, tal como se ve en la Figura 3.10.

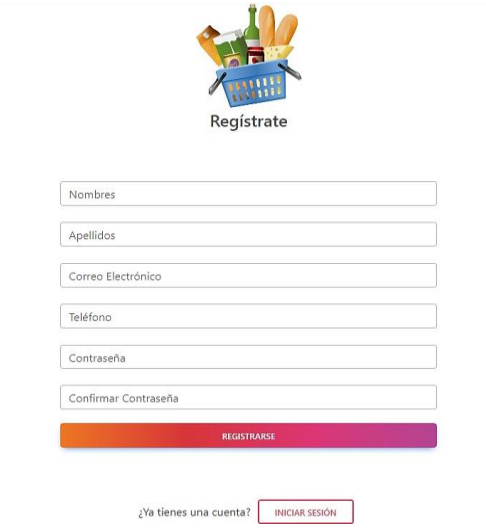

Figura 3.10 Interfaz para el registro del usuario

En la Figura 3.11 se observa que igual que en el formulario de ingreso, si al momento de enviar los datos a la API esta responde con un error, será visualizado debajo del campo donde el error se está generando, de esta manera indicando al usuario que es lo que está realizando mal.

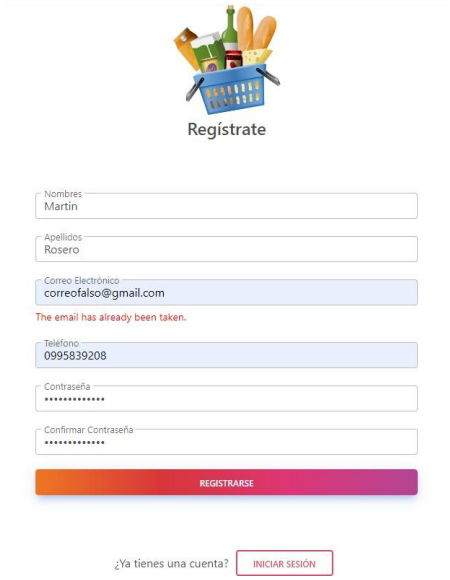

Figura 3.11 Ejemplo de error al ingresar información del API

Para poder enviar los datos del formulario al *backend* se tiene un método post en axios, con la información del formulario en formato json, una vez realizado con éxito el registro se procede a generar una cookie con el token de acceso para poder verificar si el usuario está ya ingresado a su sesión, esto se ve en las figuras 3.12 y 3.13.

```
// Realizar la solicitud POST al endpoint de registro
axios
.post("https://marketplaceppc.fly.dev/api/register", formData)
.then((response) => {
  // Manejar la respuesta del servidor
  console.log(response.data);

  // Obtener el token de acceso de la respuesta
  const { token } = response.data;
```

Figura 3.12 Solicitud POST al *endpoint* de registro

```
// Guardar el token en la cookie
Cookies.set("token", token);
Cookies.set('registro', JSON.stringify(response.data));
// Cookies.set('id_user', response.data.client, { path: '/' });
Cookies.set('id_user', response.data.client.id_user, { path: '/' });
Cookies.set('idRole', response.data.user.idRole, { path: '/' });
```

Figura 3.13 Guardado del token en la *cookie*

Sprint 2.

Página de inicio.

Esta es una de las vistas más importantes ya que es lo que primero ve el usuario al momento de entrar en el sitio web, debe ser simple pero llamativo para que el usuario pueda sentirse cómodo e ir explorando las opciones que se le brinda.

Su diseño cuenta con 3 componentes principales: *Header, Home y Footer*, estos se encargaran de darle una estructura a la vista, siendo el *header* la barra de navegación con las opciones de ingreso o registro según desee el usuario, además de presentar con formato tarjeta las tres categorías que maneja el sistema, las cuales son: Limpieza, Tecnología y Alimentos, estos mismos funcionaran como hipervínculos para llevarlos al catálogo de productos según la categoría seleccionada, y finalmente el *footer* será un componente general al igual que *header* el cual mostrara información de contacto, algunos enlaces y las redes sociales del *minimarket*, tal como se ve en la figura 3.14.

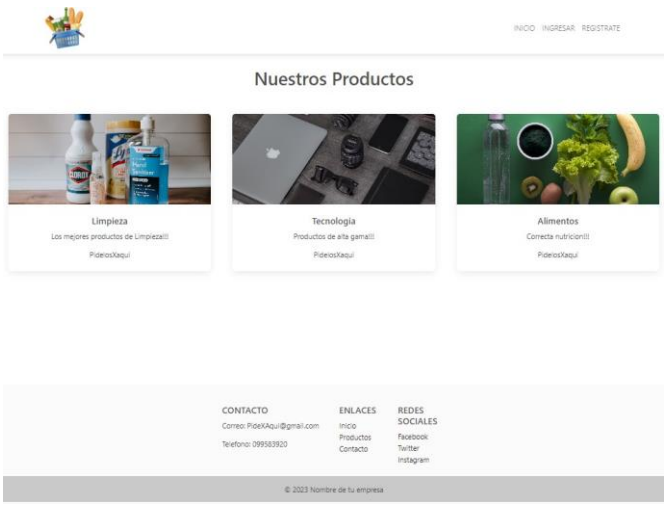

Figura 3.14 Página de inicio

Búsqueda de productos.

Una manera de agilizar la ubicación rápida de uno o varios productos es a través de herramientas de búsquedas, este componente toma en cuenta la búsqueda por escritura y filtrado por categoría.

El primero es la búsqueda por medio de ingreso por teclado, la cual buscara la similitud entre los productos presentes, figura 3.15 y los actualizara en tiempo real según el usuario siga escribiendo.

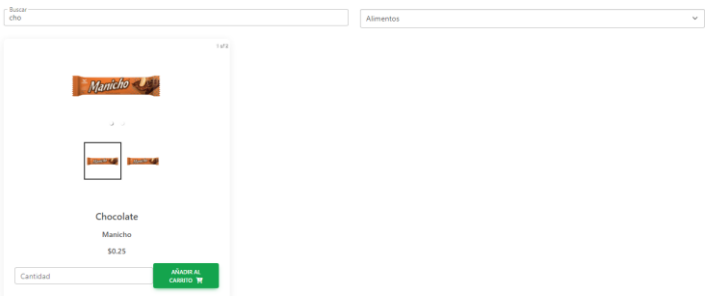

Figura 3.15 Búsqueda de productos por teclado

Las categorías de los productos que se manejan ayudan a filtrar con mayor facilidad el tipo de producto que posiblemente estén buscando, en la figura 3.16 se tiene un filtrado en base a la categoría seleccionada por el usuario.

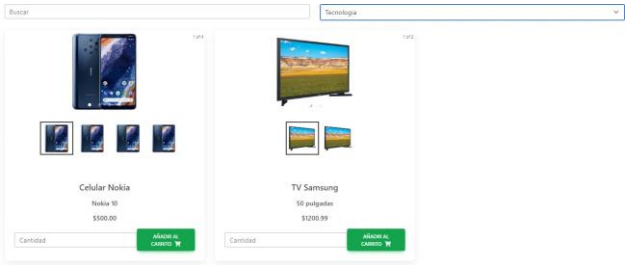

Figura 3.16 Búsqueda por categoría de productos

Para lograr esto se hay un campo de tipo *input*, el cual tendrá como valor predeterminado la variable "*serchTerm*" que será la encargada de almacenar lo que el usuario ingrese por teclado y será evaluada por la función "*filteredProducts*", así mismo cuenta con otro campo *select* que albergará las opciones de filtrado según las categorías existentes y enviara la seleccionada por el usuario a la función anteriormente mencionada, tal como está en la figura 3.17.

```
const filteredProducts = products.filter((product) => {
  const matchSearch =
    product.nombre_producto
      .toLowerCase()
      .includes(searchTerm.toLowerCase()) ||
    product.detalle.toLowerCase().includes(searchTerm.toLowerCase());
  const matchCategory =
    categoryFilter === "" || product.id_categoria === parseInt(categoryFilter);

  return matchSearch && matchCategory;
});
```

Figura 3.17 Lógica de la función "*filtredProducts*"

Una vez la función reciba algún parámetro de cualquiera de los dos campos empezara a filtrar la lista total de productos y los mostraremos en el componente de tarjetas usando un mapeo, la lógica para esto se encuentra en la figura 3.18 y 3.19.

```
<MDBRow>
 <MDBCol md="6" className="mb-4">
   <MDBInput
     type="text"
     label="Buscar"
     value={searchTerm}
     onChange={handleSearch}
   />
 </MDBCol>

 <MDBCol md="6" className="mb-4">
   <select
     className="form-select"
     aria-label="Category"
     value={categoryFilter}
     onChange={handleCategoryFilter}
   >
     <option value="">Todas las Categorias</option>
     <option value="1">Alimentos</option>
     <option value="2">Limpieza</option>
     <option value="3">Tecnologia</option>
   </select>
 </MDBCol>
</MDBRow>
```

Figura 3.18 Estructura de escritura y filtrado

```
<MDBRow>
  {filteredProducts.map((product) => (
    <MDBCol md="12" lg="4" className="mb-4" key={product.id}>
```

Figura 3.19 Ejemplo del mapeo para el componente de tarjetas

Carrito de compras.

Para esta característica se tiene varios aspectos, siendo uno de ellos el poder ver cuántos elementos hemos añadido al carrito por medio de un contador en el *header* de la aplicación, ejemplificando esto en la Figura 3.20.

Figura 3.20 Contador de ítems "carrito de compras"

Además de que sirve como icono de redirección a la vista del detalle del carrito de compras, el cual contiene las unidades de cada producto, la suma de los valores de precio, método de pago y un botón para ir a pagar, tal como se ve en la Figura 3.21.

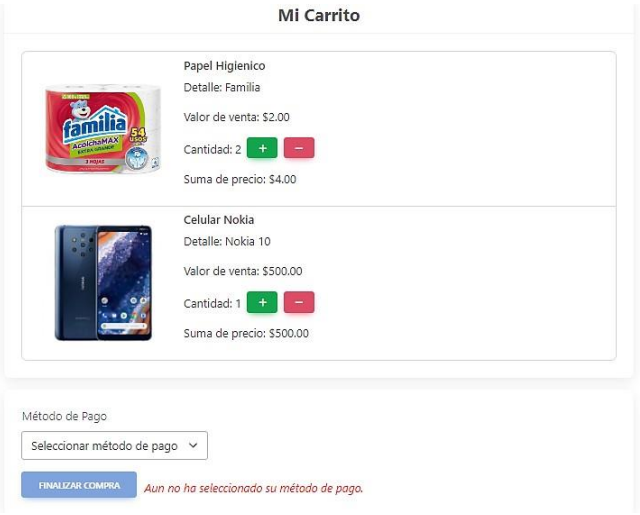

Figura 3.21 Detalle visual del carrito de compras

Para lograr eso, primero se establece que al momento de añadir el primer producto crea una *cookie* “carrito” para guardar el texto “carrito creado” y envía la petición *post* al *backend* para poder crear este carrito mandando la id del usuario y se guarda la respuesta del servidor en otra *cookie* “infocart”, para finalmente enviar otra petición por medio de axios ahora enviando las características del producto que añadió, además del número de unidades, para los demás productos que no sean el primero obtendrá la id de la cookie “infocart” y realizara el mismo proceso, la lógica para este proceso se encuentra en la Figura 3.22.

```
const idUser = Cookies.get('id_user');

if (!idUser) {
  console.log('No se encontró el id_user en la cookie');
  return;
}
const response = await axiosInstance.post('/cart/create', {
  id_user: parseInt(idUser),
});
Cookies.set('infocart', JSON.stringify(response.data));

// Extrae el ID del carrito de la respuesta
const cartId = response.data['cart id'];

// Envía los datos del producto añadido al carrito a la ruta http://127.0.0.1:8000/api/cart/add
const addProductResponse = await axiosInstance.post('/cart/add', {
  id_cart: cartId,
  id_producto: product.id,
  cantidad: parseInt(product.cantidad),
});
```

Figura 3.22 Peticiones del carrito de compras

Sprint 3.

Perfil personal.

Esta vista se encarga de mostrar los datos del cliente y sus pedidos en caso de que los tengo, además de poder entrar en un modo de edición y poder llenar un formulario con la finalidad de proporcionar los datos correctos para poder llevar el producto al cliente, tal como está en la figura 3.23.

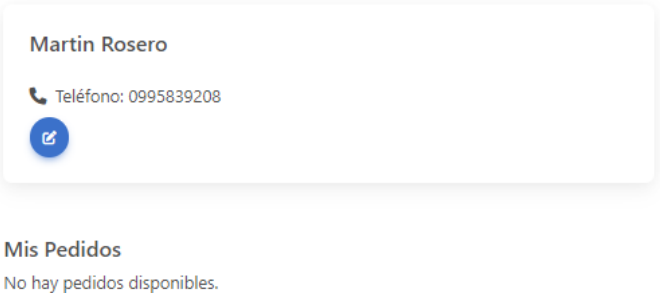

Figura 3.23 Vista del perfil personal del cliente

Sprint 4.

Gestión de cuentas.

Para la gestión de cuentas, tanto de administradores como de empleados se usa el mismo componente solo que realiza diferentes peticiones a diferentes rutas en función del rol del usuario.

Además, para facilitar la navegación entre los elementos en la lista se tiene un componente reutilizable, el cual se centra en la paginación de 5 en 5 elementos, como se ve en la Figura 3.24.

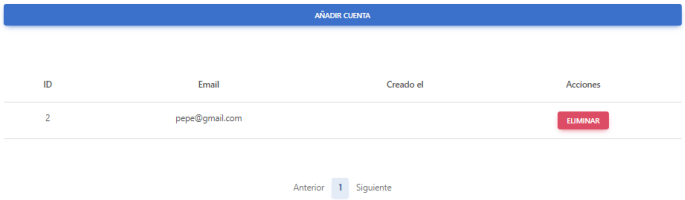

Figura 3.24 Administración de cuentas

Cuenta así mismo con una advertencia antes de eliminar definitivamente una cuenta para evitar errores accidentales.

Para la creación de cuentas se tiene un formulario con los campos necesarias, ya que estas cuentas son solo para control del sistema solo se pide correo, contraseña y confirmar contraseña, tal como está en la figura 3.25.

Figura 3.25 Diseño visual para la creación de cuentas

Con la ayuda de una sentencia condicional se decidirá a que ruta hacer la petición, siendo que si el rol del usuario es 0 crea una cuenta de administrador y si es 1 la cuenta será de empleado, la lógica se puede ver en la figura 3.26.

```
const response = await axiosInstance.post(
  formData.idRole === 0 ? "/registerAdmin" : "/registerEmployee",
  {
    email: formData.email,
    password: formData.password,
    password_confirmation: formData.confirmPassword,
  }
```

Figura 3.26 Sentencia condicional para la creación de usuarios

Gestión de pedidos.

El primer paso para crear un pedido es completar el carrito de compras, una vez conforme con todos los productos y cantidades de los mismos se requiere de la confirmación de datos para poder generar el pedido.

El perfil que se encargara de evaluar los pedidos es el de los empleados, ellos tendrán acceso a esta funcionalidad de visualizar todos los pedidos de la aplicación en formato lista, además de poder buscar según el nombre del cliente y filtrar por el estado del pedido,tal como está en la Figura 3.27.

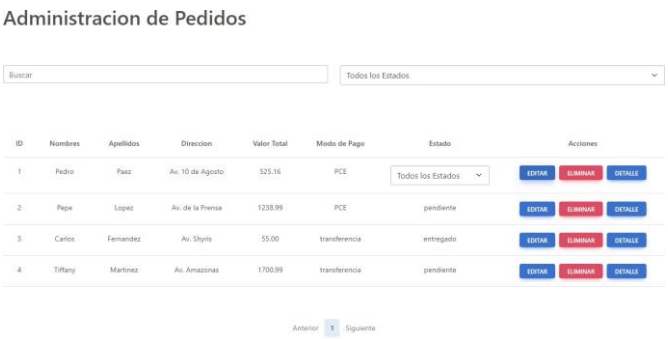

Figura 3.27 Diseño visual para la administración de pedidos

La funcionalidad de eliminar se categoriza como de tipo sensible, así que presenta el mismo funcionamiento que en productos, es decir, dará una alerta de confirmación antes de eliminar para que no existan errores y confusiones por parte del usuario, tal como está en la figura 3.28.

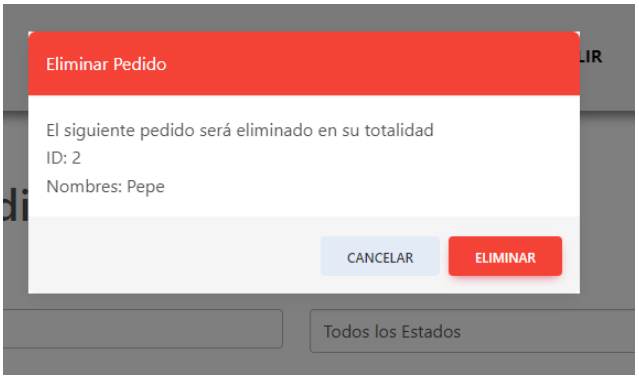

Figura 3.28 Alerta de confirmación para eliminar pedidos

Para este componente se usa el recurso *MDBTable* que permite enviar parámetros de configuración de la tabla, como serian el formato *responsive*, las columnas y de que variables van a estar llenas. Además, se mostrarán los botones de eliminar y editar, su formato se encuentra en la figura 3.29 y 3.30.

```
<MDBTable responsive>
  <MDBTable>
    <MDBTableHead>
      <tr>
        <th className="text-center">ID</th>
        <th className="text-center">Category ID</th>
        <th className="text-center">Image</th>
        <th className="text-center">Name</th>
        <th className="text-center">Detail</th>
        <th className="text-center">Stock</th>
        <th className="text-center">Value</th>
        <th className="text-center">Actions</th>
      </tr>
    </MDBTableHead>
```

Figura 3.29 Uso del recurso MDBTable para la página web

```
<MDBTableBody>
  {filteredProducts.map((product) => (
    <tr key={product.id}>
      <td className="text-center">{product.id}</td>
      <td className="text-center">{product.id_categoria}</td>
      <td className="text-center">···
      </td>
      <td className="text-center">{product.nombre_producto}</td>
      <td className="text-center">{product.detalle}</td>
      <td className="text-center">{product.stock_number}</td>
      <td className="text-center">{product.valor_venta}</td>
      <td className="text-center">···
      </td>
    </tr>
  ))}
</MDBTableBody>
```

Figura 3.30 Parámetros de configuración de las tablas en MDBTable

Estado del pedido.

Este se controla a través del rol de los empleados, los cuales pueden cambiar el estado del pedido para registrarlo en función de la realidad, para esto se entra en un modo de edición para que se pueda editar el campo “Estado” del cliente seleccionado, esto igual se ve reflejado en la Figura 3.27.

Sprint 5.

Componente CRUDs.

Para el componente CRUD de productos se tiene una vista con formato lista, la cual cuenta con 3 botones: añadir producto, borrar y editar, además de contar con la funcionalidad antes creada de búsqueda y filtrado por categoría, tal como se en la Figura 3.31.

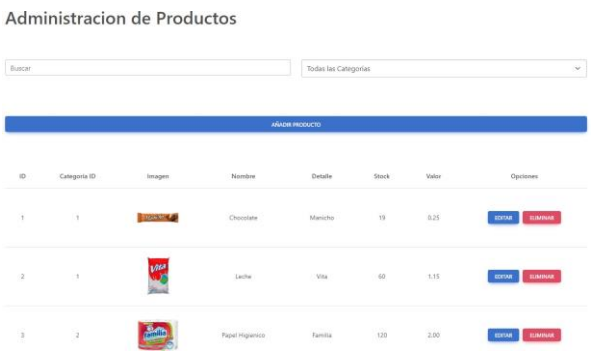

Figura 3.31 Diseño en formato tabla para la administración de productos

Para poder añadir el producto será un formulario con los campos necesarios para lograrlo: escoger categoría, nombre del producto, detalle, precio y stock, la vista se encuentra en la Figura 3.32.

Figura 3.32 Formulario para ingresar un producto al sistema

Para la creación de un nuevo producto se validan los datos y si todo estos son correctos se envían a la ruta respectiva en un formato json, la lógica para la validación se encuentra en la figura 3.33 y la petición *post* al servidor en la Figura 3.34.

```
const formattedValor = parseFloat(formData.valor).toFixed(2);
const productData = {
  id_categoria: parseInt(formData.id_categoria),
  nombre: formData.nombre,
  detalle: formData.detalle,
  stock: parseInt(formData.stock),
  valor: formattedValor,
};
```

Figura 3.33 Validación de datos

```
const response = await axiosInstance.post("/products/store", productData);
if (response.status === 201) {
  const productId = response.data.product_id;
  console.log("Producto creado con ID:", productId);
```

Figura 3.34 Petición post para registrar nuevo producto

Para prevenir accidentes de borrado por error, al dar *click* en el botón de eliminar, saldrá una alerta en forma de ventana emergente, la cual solicitará la confirmación del producto a eliminar, mostrando sus características, se puede observar un ejemplo en la Figura 3.35.

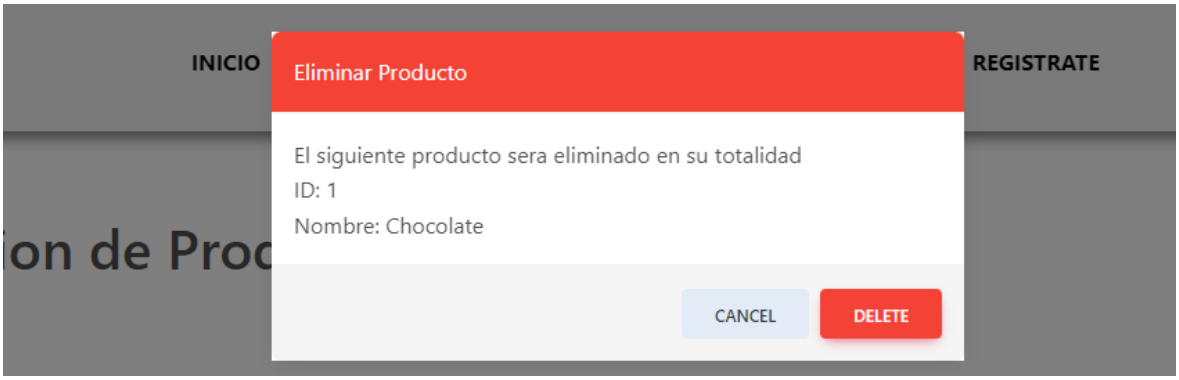

Figura 3.35 Alerta de confirmación para eliminar productos

Sprint 6.

Iconos funcionales.

Dar funcionalidad a iconos visuales proporciona al sistema un formato intuitivo de navegación y funcionalidad no limitándose a botones. Para esto en la funcionalidad del *header* el icono principal ubicado a la izquierda del *navbar* se le dio la redirección a la página *home* del aplicativo, podemos ver el icono usado en la Figura 3.36.

Figura 3.36 Ícono visual en la página web

Se usa el recurso “Link” de la librería react router dom en la cual envolvemos la sección del componente y enviamos la ruta a la que queremos enviar mediante un *click,* la implementación se encuentra en la Figura 3.37.

```
<Link to="/">
  <div className="logo">
    <img src={canasta} alt="logo" width="90" />
  </div>
</Link>
```

Figura 3.37 Uso del recurso "Link"

Optimización.

Para que los componentes no se llenen de códigos individuales se crean algunos componentes auxiliares, los cuales tienen como objetivo ser utilizados por los componentes principales y ser fácilmente editables.

Las diferentes peticiones que se realizan al *backend* son en su mayoría basadas en la misma estructura: tipo de petición, ruta, token y contenido. Para todas aquellas peticiones que sigan la estructura mencionada se tiene el componente “axiosInstance.js” este se encarga de ser un formato de petición editable e invocable para realizar las diferentes peticiones solo enviando los parámetros mencionados, tal como lo podemos observar en la figura 3.38.

```
import axios from 'axios';
import Cookies from 'js-cookie';

// Crear instancia de axios con los encabezados comunes
const axiosInstance = axios.create({
  baseURL: 'http://127.0.0.1:8000/api', // Establece la URL base para todas las solicitudes
  headers: {
    common: {
      Authorization: `Bearer ${Cookies.get('token')}`, // Encabezado de autorización común para todas las solicitudes
    },
  },
});

export default axiosInstance;

```

Figura 3.38 Componente axiosInstance

En la figura 3.39 se encuentra la solución para lograr mostrar una correcta distribución del cometido en formato de tablas, siendo este un elemento de paginación destinado a manejar correctamente el número de páginas, limitar el retroceso y avance dependiendo la situación.

```
import React from 'react';

const Pagination = ({ currentPage, totalPages, onPageChange }) => {
  const pageNumbers = [];

  for (let i = 1; i <= totalPages; i++) {
    pageNumbers.push(i);
  }
```

Figura 3.39 Paginación optimizada

Al ser un sistema web que requiere llevar un inventario en tiempo real se hace uso de un cronometro de 30 minutos para llevar a cabo la compra, si este plazo se cumple la sesión es eliminada y el carrito en cache también, de esta forma mantiene el *stock* en perfectas condiciones, esta lógica se encuentra en la Figura 3.40.

```
  const newRequestTimer = setTimeout(() => {
    // Realizar las peticiones aquí cuando termine el cronómetro
    handleLogout();
    const infocartCookie = Cookies.get('infocart');
    if (infocartCookie) {
      axiosInstance.delete('/api/cart/delete');
      console.log('Petición de eliminar carrito enviada');
    }
  }, durationForRequestsInMilliseconds);

  setTimerRequest(newRequestTimer);
};
```

Figura 3.40 Lógica de expiración de sesión

Despliegue.

Para el despliegue del *frontend* primero se verifica que no existan errores críticos en la ejecución del ambiente de desarrollo, seguido a esto se usa la herramienta GitHub para que Netlify construya la página web a partir del repositorio, vemos la ejecución del mismo en la Figura 3.41.

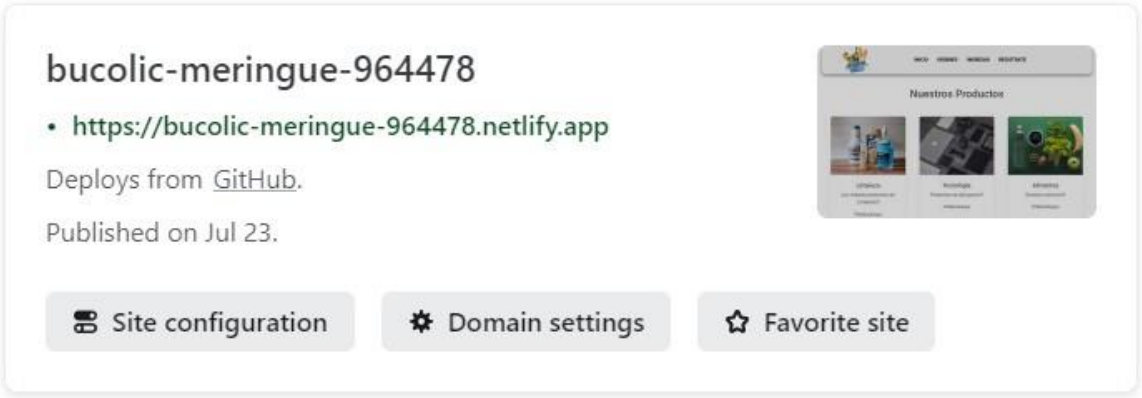

Figura 3.41 Despliegue del componente en Netlify

4 CONCLUSIONES

- Basándose en las entrevistas a dependientes de minimarkets se pudo llegar a la conclusión de que estos no cuentan con un sistema informático de gestión de pedidos y que este puede resultar de mucha ayuda a la hora vender productos de manera online.

- El diseño de prototipado en un desarrollo frontend es una guía que, aunque no es de carácter estricto si es de utilidad para plantearse ideas iniciales que pueden o no ser usadas en el diseño final de la aplicación.

- En todo desarrollo de software es indispensable el uso de una arquitectura para llevar un desarrollo enfocado y estructurado, siendo el usado en este componente la arquitectura MVV la cual ayudo a estructurar los componentes de la aplicación, identificar la lógica UI y como mostrar los datos obtenidos por parte del backend.

- Siendo este componente un proyecto en el cual se requiere velocidad, agilidad y una buena organización en relación del tiempo se hizo uso de la metodología ágil SCRUM, la cual permite tener sprints semanales correspondientes a programación y documentación a la vez para no retrasar el avance del componente en ningún sentido.

- La programación del componente se realizó en el framework React, este se escogió entre otros del mismo tipo debido a que es uno de los más recomendados y populares a la hora del desarrollo frontend, lo cual permite tener acceso a una gran variedad de documentación, tutoriales, cursos y plantillas, las cuales resultaron indispensables a la hora de desarrollar el aplicativo.

- El despliegue del componente se realizó a través de la plataforma Netlify la cual permite construir el proyecto desde un repositorio en Github, idónea para poder visualizar los avances ya en un entorno online y desde cualquier dispositivo, así mismo puede realizar sin problemas los procesos de cada componente react y las peticiones a las rutas API del backend.

5 RECOMENDACIONES

Las siguientes recomendaciones obtenidas al desarrollar este componente tienen como finalidad dar sugerencias y ayuda práctica.

- En el caso de no tener ninguna experiencia desarrollando en react hay que dedicar una parte del tiempo inicial a aprender del mismo a través de su documentación oficial, cursos online, ejercicios de creación de componentes, enrutamiento y consumo de APIs.
- En muchos de los casos se debe priorizar el que los componentes react desarrollados sean reutilizables y no sean muy estáticos, aunque en ciertos casos muy específicos se puede permitir enfocar un componente solo a una única función.
- En toda herramienta que se use siempre antes de instalarla en el proyecto se debe de verificar su compatibilidad de versiones ya que algunas veces no funcionaran de manera correcta o puede llegar a corromper el proyecto en su totalidad.
- Para evitar perdida de progreso en el desarrollo es obligatorio el uso de un gestor de versiones que nos permita tener un registro de todo cambio realizado y si en algún caso es necesario volver a una versión antigua, para esto una de las mejores herramientas es Github el cual además de darnos esta funcionalidad de manera gratuita también nos permite tener compatibilidad con otras herramientas como Netlify y darnos facilidad a la hora de exponer el componente al público debido a la sección “readme”.
- A pesar de que el desarrollo de integración curricular no obligue a trabajar muy en conjunto, en muchos casos una comunicación fluida puede ahorrar mucho tiempo y confusiones, siendo que al ser parte del mismo aplicativo tanto el front como el back se deben compenetrar de una manera óptima para los mejores resultados posibles.
- Se recomienda siempre ir presentando los avances a alguien ajeno al proyecto, con la intención de obtener sugerencias objetivas y constructivas, además de poder encontrar posibles errores los cuales se hayan pasado por alto a la hora de realizar algunas pruebas de funcionalidades.

6 REFERENCIAS BIBILIOGRÁFICAS

[1] MDN contributors, «MDN Web Docs,» 29 Noviembre 2022. [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/Front-end_web_developer. [Último acceso: 2 6 2023].

[2] Mozilla, «MDN Web Docs,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Glossary/CRUD. [Último acceso: 1 Agosto 2023].

[3] R. N. Dan Abramov, «react.dev,» 16 Marzo 2023. [En línea]. Available: https://react.dev/blog/2023/03/16/introducing-react-dev. [Último acceso: 2 6 2023].

[4] K. Ubah, «freecodecamp,» 10 Agosto 2021. [En línea]. Available: https://www.freecodecamp.org/news/html-css-and-javascript-explained-for-beginners/. [Último acceso: 20 Julio 2023].

[5] Mozilla, «MDN Web Docs,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS. [Último acceso: Agosto 1 2023].

[6] Mozilla, «MDN Web Docs,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Último acceso: 2 Agosto 2023].

[7] AXIOS, «axios-http,» [En línea]. Available: https://axios-http.com/docs/intro. [Último acceso: 20 Julio 2023].

[8] MDBoostrap, «MDB - Material Design for Bootstrap,» [En línea]. Available: https://mdbootstrap.com/docs/react/getting-started/installation/. [Último acceso: 20 Julio 2023].

[9] J. S. Ken Schwaber, «SCRUM GUIDES,» Noviembre 2020. [En línea]. Available: https://scrumguides.org/scrum-guide.html. [Último acceso: 2 Junio 2023].

[10] Universidad Continental, «Blogs Universidad Continental,» 11 Mayo 2017. [En línea]. Available: https://blogs.ucontinental.edu.pe/marco-teorico-elaborarlo/temas/consejos/. [Último acceso: 21 Julio 2023].

[11] A. M. Andrada, «Universidad Americana de Europa,» 8 Septiembre 2020. [En línea]. Available: https://unade.edu.mx/fuentes-de-informacion-confiables/. [Último acceso: 21 Julio 2023].

[12] The Interaction Design Foundation, «The Interaction Design Foundation,» 17 Junio 2019. [En línea]. Available: https://www.interaction-design.org/literature/topics/prototyping. [Último acceso: 2 Junio 2023].

[13] INTELEQUIA, «Intelequia,» 28 Noviembre 2020. [En línea]. Available: https://intelequia.com/blog/post/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber. [Último acceso: 2 Agosto 2023].

[14] P. J. Maida Esteban Gabriel, *Metodologías de desarrollo de software,* Buenos aires, 2015.

[15] React, «React,» [En línea]. Available: https://es.react.dev/learn/your-first-component. [Último acceso: 12 Julio 2023].

[16] Coursera , «Coursera,» 15 Junio 2023. [En línea]. Available: https://www.coursera.org/articles/scrum-roles-and-responsibilities. [Último acceso: 4 Agosto 2023].

[17] C. Harris, «Atlassian,» [En línea]. Available: https://www.atlassian.com/es/agile/scrum/artifacts. [Último acceso: 4 Agosto 2023].

[18] M. Granieri, «OBS Business School,» 22 Julio 2022. [En línea]. Available: https://www.obsbusiness.school/blog/los-beneficios-del-prototipado-web-en-el-diseno-de-interfaces. [Último acceso: 5 Agosto 2023].

[19] Figma, «Figma Learn,» [En línea]. Available: https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma. [Último acceso: 5 Agosto 2023].

[20] K. Harsh, «Kinsta,» 14 Octubre 2022. [En línea]. Available: https://kinsta.com/es/blog/arquitectura-aplicaciones-web/. [Último acceso: 7 Agosto 2023].

[21] J. M. M. Ortega, «OpenWebinars,» 4 Febrero 2019. [En línea]. Available: https://openwebinars.net/blog/la-arquitectura-mvvm-y-sus-componentes/. [Último acceso: 7 Agosto 2023].

[22] Indeed, «Indeed,» 25 Abril 2023. [En línea]. Available: https://es.indeed.com/orientacion-laboral/desarrollo-profesional/herramientas-desarrollo-software#:~:text=Las%20herramientas%20de%20desarrollo%20de%20software%20son%20programas%20informáticos%20que,herramientas%20de%20programación%20de%20software.. [Último acceso: 7 Agosto 2023].

[23] Netlify, «Netlify,» [En línea]. Available: https://www.netlify.com/?attr=homepage-modal. [Último acceso: 8 Agosto 2023].

[24] Microsoft, «Visual Studio Code - Code Editing. Redefined,» 3 11 2021. [En línea]. Available: https://code.visualstudio.com/docs. [Último acceso: 8 Agosto 2023].

[25] Create React App, «Getting Started | Create React App,» [En línea]. Available: https://create-react-app.dev/docs/getting-started/. [Último acceso: 8 Agosto 2023].

[26] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.