Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

"Desarrollo de Aplicación Web Reactiva para la Gestión de un Supermercado"

Tesis Profesional

Para obtener el título de Licenciatura en

Ingeniería en Ciencias de la Computación

Presenta

Jorge Leonel Muñoz Tehuitzil

Asesor

Dr. Abraham Sánchez López

Abril 2023

Agradecimientos

Gracias a Dios por cuidarme, brindarme salud y permitirme cumplir esta meta de obtener mi licenciatura acompañado de mis seres queridos.

A mis padres quienes me acompañaron y apoyaron brindándome las palabras, el aliento y recursos necesarios a lo largo de toda mi vida. Gracias a sus enseñanzas hoy día me permitieron crecer y convertirme en la persona que soy hoy en día, además de ello agradezco haberme permitido tomar la decisión de elegir mi carrera profesional siendo mi mejor herencia entregada.

A mi asesor el Dr. Abraham Sánchez López por la dedicación y el esfuerzo que realiza con sus estudiantes en cada una de sus clases formando a profesionales con los conocimientos necesarios para enfrentar el campo laboral. Gracias por brindarme el apoyo necesario durante mis últimos semestres de la carrera permitiéndome tomar clases externas a mi mapa curricular aportándome conocimientos necesarios y finalmente por acompañarme a desarrollar la tesis para obtener mi grado.

A mi compañero de carrera Ian quien fue un asesor más, brindándome el apoyo, asesorías y las palabras adecuadas para seguir adelante, por las vivencias que pasamos juntos, gracias.

Contenido

Agradecimientos 3
Capítulo 1 7
Introducción 7
Planteamiento del Problema 8
Antecedentes 9
Objetivo 10
Capítulo 2 11
Estado del arte 11
Paradigmas de la Programación 12
Aplicaciones Web 13
Cliente / Servidor 13
Programación Reactiva 15
Programación Funcional 17
Aplicaciones Web Reactivas 18
Capítulo 3 20
Presentación del Problema 20
Propósito 20
Ámbito del Sistema 20
Definiciones, Acrónimos, y Abreviaturas 21
Visión General del Documento 22
Perspectiva del Producto 22
Funciones de Producto 22
Características de los Usuarios 23
Restricciones 23
Suposiciones y Dependencias 23
Requisitos Futuros 24
Requisitos Específicos 24
Interfaces Externas 25
Funciones 25
Requisitos de Rendimiento 25

Restricciones de Diseño 26
Atributos del Sistema 26
Capítulo 4 27
Análisis de Requerimientos 27
Diagrama de Casos de Uso 27
Descripción de Casos de Uso 28
Auditoria de aplicaciones similares 35
Análisis de usuario 39
Mapa heurístico para el intercambio de ideas 39
Maquetado de interfaces 40
Paleta de colores 44
Interactividad 44
Gestión del espacio 46
Capítulo 5 48
React 48
Firebase 48
TypeScript 49
Github Pages 49
Desarrollo 50
Funciones Puras 50
Programación Declarativa 51
Programación Reactiva 53
Vistas Finales de Aplicación 55
Capítulo 6 58
Pruebas y Resultados 58
Capítulo 7 62
Conclusiones y trabajos futuros 62
Capítulo 8 63
Referencias 63

Capítulo 1

Introducción

El término reactivo es posible visualizarlo acompañado en diferentes conceptos que lo incorporan, como aplicaciones reactivas, programación reactiva, flujos o manifiesto reactivos, en el área de tecnología la palabra no es reciente, tiene apariciones desde 1985 con los mencionados sistemas reactivos, los cuales hacen referencia a sistemas estimulados por su ambiente externo y la manera en que estos dan respuesta ante estos estímulos. Actualmente se tiene mayor presencia del término reactivo debido a las nuevas exigencias ante la aparición de nuevas tecnologías que son integradas a los desarrollos actuales, brindando soluciones más robustas y satisfactorias.

El desarrollo de aplicaciones web reactivas es una respuesta ante nuevos requerimientos, anteriormente se desarrollaban aplicaciones que únicamente se dedicaban a generar resultados desde su propia lógica implementada, posteriormente se fueron adaptando a nuevas necesidades como integración de servicios web, disponibilidad, asincronismo, respuesta ante eventos y solución de problemas sin intervención por parte del usuario. En este primer capítulo se muestran los requerimientos del proyecto a realizar con base en la reactividad, así como de los antecedentes y objetivos de realizar el presente trabajo de tesis.

Planteamiento del Problema

Actualmente las aplicaciones requieren implementar paradigmas que les permitan cumplir los nuevos requerimientos de los usuarios de manera que sigan siendo de su preferencia ante otras propuestas. Para cumplir con estas nuevas exigencias se debe lograr una fluidez e interacción con el usuario de manera eficiente, de esta manera se obtiene respuesta inmediata ante eventos o peticiones realizadas por el usuario. También son considerados los eventos inesperados o errores que lleguen a detectarse y corregirse de manera efectiva [12], lo que evita la interrupción de su flujo de trabajo.

El empleo de estos paradigmas no es por convención aplicable a todo desarrollo que debamos realizar, la aplicación del paradigma de programación reactiva es aplicable a proyectos que requieran de grandes cantidades de consultas a alguna API, múltiples usuarios, actualización inmediata ante eventos e interacción de los usuarios. Por lo anterior en este trabajo de tesis se busca desarrollar una aplicación web reactiva en la cual se emplee el paradigma de programación reactiva, esta aplicación será una propuesta para la administración de un supermercado.

Antecedentes

Los antecedentes de la programación reactiva tienen su aparición desde 1970, aunque fue hasta 2010 que inicio a dar de que hablar entre los desarrolladores y desde entonces comenzó su evolución la cual se ha establecido en generaciones que permitieron llegar a su madurez actual. La generación 0 consta de APIs como java.util.Observable y algunas addXXXListener con las cuales se presentaban deficiencias de composición y que por sí mismas resultaban limitadas. En la generación 1 fueron reconocidas las deficiencias presentadas en la generación 0 por lo que aparecieron las primeras librerías reactivas entre las que se encuentran RX.NET, Reactive4Java y RxJava entre 2010 - 2013, en esta generación los inconvenientes se centraban en la implementación de IObservable/IObserver debido a cancelación de ejecución con el operador take(). Consideradas las anteriores deficiencias para la generación 2 el equipo de RxJava introdujo la clase Suscriber, la interfaz producer y el método lift() con las cuales se presentaba limitación en la optimización y la compatibilidad con otras librerías.

La generación 3 consta de actualizaciones en los avances logrados anteriormente debido a la creación de la especificación Reactive Stream establecida por expertos de diferentes compañías, para la generación 4 fueron realizadas reimplementaciones lo que llevo a establecer la librería reactive-streams-commons con lo cual fue necesario la creación de operadores y componentes que reciben el nombre de operator-fusion. Finalmente, de la generación número 5 a la actualidad la atención era puesta en extensiones que permitieran soportar operaciones reactivas de entrada y salida de manera que fueran realizadas de forma bidireccional [10].

Objetivo

La presente investigación tiene como objetivo desarrollar una aplicación web que cumpla con la infraestructura y características propias de una aplicación reactiva, de manera que se pueda apreciar y justificar su aporte en aplicaciones actuales. Por lo tanto, se plantean los siguientes objetivos específicos:

- Estudiar las bases de la programación reactiva.
- Identificar la aportación de la reactividad en aplicaciones modernas.
- Definir la aportación de implementar programación reactiva-funcional.
- Analizar las diferentes tecnologías que permiten implementar reactividad y programación funcional.
- Desarrollar una aplicación web reactiva que cumpla con las características de la reactividad.

Capítulo 2

Estado del arte

El paradigma de programación reactiva actualmente está permitiendo mejorar el desarrollo de aplicaciones, logrando cumplir con las nuevas tendencias y necesidades tecnológicas requeridas. Mediante la implementación de este paradigma se logra una interacción fluida sin importar la cantidad de datos solicitados a través de peticiones o bien eventos en su entorno que le generen trabajo en cuanto a su funcionalidad, así mismo los errores dejan de ser un problema en el flujo de trabajo pues sin importar que se lleguen a presentar no se inhabilita la interacción de los usuarios o bien se llegan a detectar de manera oportuna antes de suscitarse.

Aplicaciones desarrolladas para el uso de múltiples usuarios requieren de estos paradigmas para mantener al usuario activo sin interrupciones, de esta manera se logra brindar calidad en el servicio que se esté ofreciendo, una vez se ejecute algún evento o cambio se actualiza de manera inmediata y de forma satisfactoria la respuesta ante el evento. Este capítulo describe los conceptos y elementos que conforman la programación reactiva y funcional que permiten implementar el paradigma adecuadamente a las aplicaciones en este caso web.

Paradigmas de la Programación

En la teoría de computación se hace mención del uso de paradigmas de programación que se emplean y utilizan de acuerdo con las necesidades requeridas en la implementación de algún proyecto, un paradigma se define como teoría o conjunto de teorías las cuales suelen ser el núcleo central y son aceptadas sin ser cuestionadas, siendo una base o modelo para dar solución a un problema [8] lo que permite avanzar en el proceso para concebir una solución adecuada. Los paradigmas de programación son aquellas teorías que permiten al desarrollador tener herramientas que permitan un desarrollo de calidad ante algún problema, estos paradigmas suelen tener características, conceptos, abstracciones y pasos que permiten representar la solución de manera adecuada.

Los paradigmas son clasificados en dos grandes grupos principalmente los cuales suelen representar a dos enfoques distintos: imperativo y declarativo. El enfoque imperativo es aquel que indica explícitamente el cómo debe de realizarse la solución, para ello se tiene una secuencia de sentencias ejecutadas de acuerdo con el flujo que modifica el estado, en el caso de las variables esta son definidas, pero pueden ser modificadas. Dentro de esta categoría de paradigmas imperativos tenemos los siguientes que lo conforman: paradigma procedimental, modular y estructurado [13].

A diferencia del paradigma imperativo, el paradigma declarativo se centra en que se debe realizar, sin importar el cómo, no cuenta con un flujo debido a que se contempla una composición funcional y la recursividad siendo de sus principales características de este paradigma, además, las variables son inmutables, una vez sean definidas no deben ser modificadas [11]. Dentro del paradigma declarativo se pueden encontrar los paradigmas: funcional, lógico, la programación reactiva y lenguajes descriptivos.

Los dos enfoques mencionados anteriormente suelen ser las principales categorías, aunque es posible encontrar otros paradigmas que centran su atención en la estructura y la organización e incluso asociados a la concurrencia y a sistemas de tipado. Cada paradigma es una manera de hallar una solución adecuada al problema por resolver por lo que se puede adoptar cualquiera de ellos de acuerdo con las características que se requiera y al lenguaje que se planea emplear a la solución pues algunos de ellos son específicos al uso de un lenguaje en espacial.

Aplicaciones Web

Internet y la web han sido uno de los grandes avances tecnológicos desarrollados para beneficio de la sociedad en general y aplicado en diferentes sectores, con ello se logró satisfacer los requerimientos que surgieron con el paso del tiempo lo que también llevo a actualizar, mejorar y desarrollar tecnologías que apoyaran en lo que hoy día conocemos de ambos desarrollos.

Así como existen ventajas dentro de cada solución, en el caso de las aplicaciones web tenemos limitantes en su funcionamiento que se identifican de manera inmediata como son la conectividad de la red, disponibilidad de hosting, afectación en seguridad de datos y falta de privacidad en datos debido a su alojamiento en servidores que desconocemos su ubicación [7].

Cliente / Servidor

Como se ha mencionado anteriormente las aplicaciones web siguen una arquitectura cliente servidor donde un ordenador es quien solicita servicios es decir el cliente y otro es quien queda en espera de solicitudes a las cuales debe responder nombrado comúnmente como servidor.

La arquitectura cliente/servidor es una arquitectura de red en la que se define con los respectivos roles a ordenadores de acuerdo con su funcionalidad dentro de la red. Los servidores son quienes requieren tener ciertas especificaciones para gestionar cada una de sus tareas entre las que principalmente se encuentra el almacenamiento y distribución de la información [6]. En el caso de los clientes es lo contrario puesto que normalmente no se requiere de mayor potencia debido a que se usan los recursos de los servidores principalmente.

La arquitectura cliente servidor es una de las arquitecturas más citadas, siendo el modelo más importante y utilizado hasta hoy en día, esta arquitectura nos permite separar la funcionalidad en tres capas o niveles que suelen distribuirse de la siguiente manera:

- Lógica de presentación: se encarga de recibir la información necesaria del usuario para poder procesarla y entregar como salida los resultados obtenidos del procesamiento.
- Lógica del negocio: se identifica como el enlace entre el cliente y los datos, se recibe la información proporcionada por el cliente, la lógica de datos se ejecuta cumpliendo las reglas del negocio o aplicación y envía los resultados a nivel presentación.
- Lógica de datos: se encarga de gestionar la información.

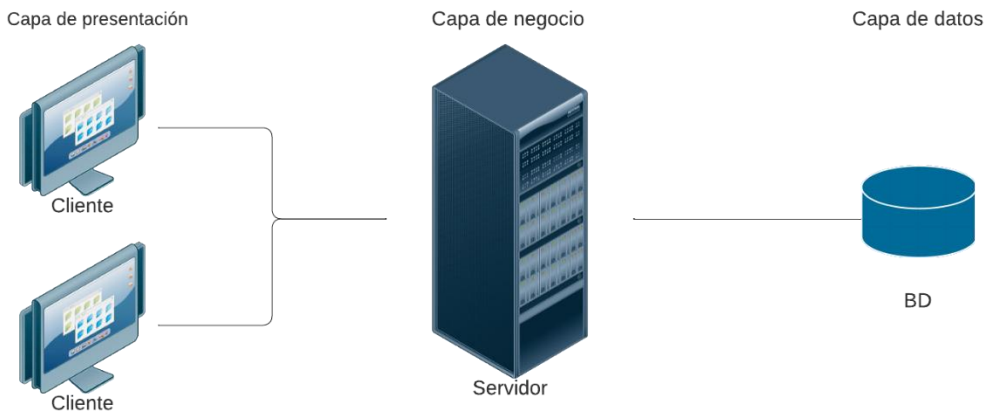

Figura 2-1 Arquitectura Cliente/Servidor

Programación Reactiva

Reactive Programming o **Programación Reactiva** es un paradigma de programación enfocado en el trabajo con flujos de datos finito o infinitos de manera asíncrona. Permite la propagación de los datos dentro de la aplicación de manera que pueda realizar cambios, es decir, reacciones a datos ejecutando una serie de eventos.

Este paradigma de programación es relacionado con el **Patrón Observer**. Si existe un cambio de estado en un objeto los otros objetos reciben una notificación que les permite actualizar, sin necesidad de estar sondeando, todo es realizado de manera asíncrona para que los observadores puedan procesarlos [9]. Un ejemplo al que se hace referencia para entender el funcionamiento de reactividad es Excel, cuando nosotros realizamos operaciones con base a datos de otras filas o columnas y la operación es definida en una celda esta mantendrá actualizado el resultado ante cualquier cambio realizado en alguno de los datos que son tomados como referencia, sin necesidad de volver a ejecutar la operación.

La programación reactiva se centra en lo establecido en el **Manifiesto Reactivo** el cual fue consolidado para cumplir los nuevos retos que enfrentan los sistemas, entre los que se encuentran la escalabilidad, más robustos, grandes cantidades de peticiones, etc., para atender estos nuevos retos se implementan los principios de reactividad que deben ser adoptados en su diseño, arquitectura y a menudo en su modelo de programación [3], a continuación, se presentan estos principios:

Responsivos

Como se ha mencionado una aplicación se basa en una arquitectura cliente/servidor, el cliente es quien se encuentra de manera directa con el usuario siendo la presentación inicial de la calidad, usabilidad y utilidad de la aplicación, es por ello que todo sistema debe ser receptivo en todo momento, no únicamente responder de manera inmediata si no que debe ser funcional dentro de sus limitaciones de tiempo de respuesta y administrar los cambios, para que las respuestas a las solicitudes puedan interpretarse en el contexto adecuado.

Hay que considerar que la respuesta adecuada ante el flujo normal y correcto no significa que el sistema sea receptivo, pues inclusive en situaciones de error debe responder y fluir de manera normal o bien en el peor de los casos responder con un mensaje.

Resilientes

El sistema debe permanecer responsivo frente a fallas y para ello se deben contener los fallos dentro de cada uno de los componentes que conforma el sistema, de manera que cada componente sea responsable de la lógica que le corresponde y de esta manera recuperarse ante cada fallo que se presente evitando siempre comprometer el sistema de manera completa.

Elásticos

El sistema se mantiene responsivo incluso ante el incremento de cantidad de peticiones, para ello se realiza una distribución de los recursos para mejor su eficiencia.

Orientado a Mensajes

El intercambio de mensajes asíncronos dentro de los sistemas reactivos le permiten gestión de carga, elasticidad y control del flujo, lo que resulta en menor sobrecarga del sistema y evita el bloqueo por completo.

Programación Funcional

La programación funcional es considerada un paradigma de programación, lo cual se considera como una manera de dar solución a los problemas que se presente mediante funciones. La programación funcional tiene sus orígenes en la matemática debido al manejo de funciones que permiten la relación entre parámetros de entrada y parámetros de salida de un proceso definido, por lo anterior es que en cada computación el resultado dependerá de una u otra forma de los parámetros, siendo una buena razón para especificar la computación. Este paradigma se centra en que a través del lenguaje de programación que se utilice, se exprese lo que se quiere realizar sin importar el cómo, todo lo contrario, a lo que tradicionalmente realizamos que es codificar paso a paso lo que se realiza hasta llegar al resultado.

La programación funcional se compone de 4 principales características que le permite generar código conciso, legible e incluso testeable, estas características son: manejo de funciones puras, datos inmutables, evitar efectos secundarios y el estado compartido [2].

Las funciones puras son parte de la programación funcional estas cumplen su objetivo al devolver únicamente un resultado, de manera que cada que se ingresen los mismos parámetros de entrada se tendrá el mismo resultado lo que elimina efectos secundarios, este tipo de funciones requieren de al menos un parámetro de entrada para poder llevar a cabo su funcionalidad. Su implementación es independiente del estado externo del proyecto lo que permite modificar su lógica sin interrumpir o introducir errores en el código de otras partes. Parte de evitar los efectos secundarios es la implementación de funciones puras las cuales al ser independientes son localizables y posiblemente pueden llegar a aislarse para evitar cualquier incidencia que se presente.

La inmutabilidad es la característica que no permite que una variable sea modificada de ninguna manera después de haber sido creada, esto evita alteraciones fuera de su ámbito al que fueron creadas y por tanto su uso es reservado.

En general la programación funcional nos otorga beneficios importantes desde un código más limpio y legible para futuros desarrolladores, facilidad en el mantenimiento y refactorización mediante las funciones puras, facilidad de concurrencia y recursión en funciones.

Aplicaciones Web Reactivas

El flujo tradicional de una aplicación web tienen deficiencias observables para el usuario, actualmente un usuario busca rapidez de respuesta antes sus peticiones y flujo de trabajo que mantenga mientras usa la aplicación, es por ello por lo que el primer acercamiento es la carga de la aplicación, desde ese momento el usuario toma su juicio de lo buena que puede llegar a ser. En las aplicaciones tradicionales se manejan refresh o recarga de todos los componentes dentro de la aplicación o por lo menos los que se encuentran en la vista actual, lo cual conlleva un tiempo de respuesta mayor en lo que el DOM realiza un renderizado de elementos y estilos para poder ser visualizados por el usuario, además el manejo de datos es de manera síncrona lo que aumenta el tiempo de respuesta y cambios o renderizados completos por cada cambio que se presente.

Por lo anterior es que se buscaron técnicas que ayuden a mejorar y optimizar los recurso que se utilizan, para ello la implementación de la reactividad en las aplicaciones web promueven un cambio en la arquitectura de las aplicaciones y estos cambios son lo que caracterizan a las aplicaciones web reactivas.

- Lógica del lado del servidor y del cliente
- Modelos de programación asincrónicos y basados en eventos.
- Comunicación entre componentes

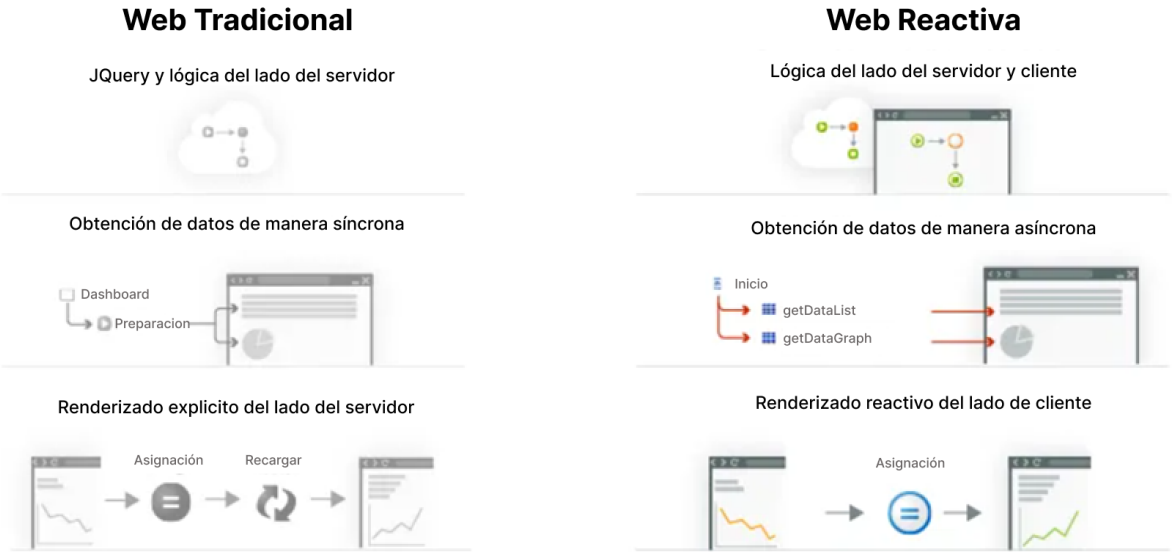

Figura 2-2 Web tradicional VS Web Reactiva

Estas características de las aplicaciones web reactivas muestran de manera inmediata la mejora en el rendimiento lo que conlleva a actualizar los elementos de la interfaz de manera inmediata reconociendo que elementos son los que sufren cambios y requieren renderizarse para presentar estos cambios al usuario. Además de la optimización y el tiempo de renderizado nuestras aplicaciones logran ser más receptivas y escalabres de manera que no se ve afectado el flujo de aplicación – usuario dando una respuesta optima.

Capítulo 3

Presentación del Problema

Realizar una aplicación web con tecnologías que permitan aplicar programación Funcional y Reactiva, se debe considerar la presencia de dos tipos de usuario, Administrador y Cliente, el tipo de usuario definirá la vista que se presenta, así como las funciones que tiene acceso. Se tendrá un carrito de compras, consulta de productos, categoría de productos y un control de pedidos para identificar la situación (estado) del pedido realizado en el proceso de la tienda.

Propósito

El propósito del documento es presentar las diferentes fases que se realizaron a cabo en el desarrollo de una aplicación web basada en ciertas especificaciones y requerimientos a fin de lograr un producto usable y eficiente. Como público objetivo a quien es dirigido es principalmente a personas del área de Desarrollo de Software especialmente en la rama de Desarrollo Web siendo una pauta para el desarrollo de un proyecto a partir de un proceso de construcción.

Ámbito del Sistema

Se requiere desarrollar una aplicación web que ayude en la gestión del super “La Jacaranda”, la aplicación debe contar con un registro e inicio de sesión que permita personalizar el perfil del usuario, los usuarios en general serán de tipo cliente y un único usuario de tipo administrador, quien será el dueño del establecimiento.

Para el registro de usuarios se deberá solicitar nombre completo, correo electrónico y contraseña o bien mediante su cuenta de Google lo que hará más fácil el proceso. Una vez realizado el registro se podrá iniciar sesión mostrando de manera personalizada su cuenta.

La vista de Inicio de la aplicación presentara un banner donde se presentarán promociones de temporada, y el listado de productos que serán definidos por categoría, cada producto tendrá su imagen, nombre y precio.

Cada producto permitirá al usuario dirigirse a una vista más detallada, donde tendrá mayores detalles como una descripción, disponibilidad del producto en tienda, selección de cantidad requerida por el cliente y agregar a su carrito de compra.

El carrito de compras permitirá visualizar los productos seleccionados para comprar, serán listados mostrando su imagen, nombre, precio, y un botón de borrar para eliminar el producto. Contará con un total previo a su compra de esta manera el usuario sabrá el costo total del pedido.

El usuario podrá visualizar sus pedidos dando seguimiento a su estado del pedido y sus detalles como id de pedido y la lista de productos solicitados.

El administrador podrá registrar, editar y eliminar productos, cada uno de ellos tendrá nombre de producto, id de producto, precio, imagen, descuento(opcional), cantidad disponible, categoría, descripción y estado que le permitirá controlar disponibilidad de producto en caso de algún detalle.

El usuario tendrá una sección de pedidos en la que podrá visualizar los pedidos actuales, ver productos solicitados y cambiar el estado del pedido (preparado, enviado, listo).

Definiciones, Acrónimos, y Abreviaturas

Stock: cantidad disponible de producto que se tiene en almacén para su venta.

Superset: conjunto de elementos adicionales que se proveen a un conjunto como mejora adicional. En el contexto de TypeScript como superset de JavaScript, se dice que proporciona mejoras para reducir las limitantes en capacidad de tipado de datos principalmente.

Framework: marco de trabajo que permite la construcción de proyectos de software teniendo una estructura básica de la cual se parte para la solución a ciertos problemas en específico.

Librería: conjunto de funciones, métodos y clases determinadas que ya se encuentran desarrollas para la implementación de funcionalidades a nuestros proyectos sin necesidad de codificar de 0.

Hooks: función o método que es ejecutado ante cierto evento que sucede dentro del sistema, estos pueden ser proporcionados por el software o bien definidos por los desarrolladores.

Asincronismo: ejecución de tareas independientes que no bloquean al programa.

APIs: interfaz que permite la comunicación de proyectos con servicios y funcionalidades de otras aplicaciones de forma estructurada y segura.

Visión General del Documento

El objetivo de este documento es la presentación del proceso que se realizó para el desarrollo de la tesis con objetivo en programación reactiva y funcional, así mismo se detalla cada una de las etapas que se llevaron a cabo explicando desde el proyecto que se pretende realizar, así como la definición de conceptos, herramientas y requerimientos necesarios para cumplir con el proyecto de manera funcional. El documento ofrece una visión general del proyecto para tener un contexto de partida al producto presentado como aplicación web.

Además, se presentan las bases obtenidas a través de análisis de aplicaciones con enfoque en comercio electrónico similares a la que se busca desarrollar, con base en lo anterior se presenta prototipado y diseño para llevar a cabo como resultado productivo.

Perspectiva del Producto

Aplicación web para el comercio electrónico del Supermercado “Jacaranda”, su objetivo para el administrador (dueño del supermercado) es tener un catálogo de productos que tiene en venta en su tienda de manera física, este catálogo de productos está disponible para todo cliente que requiera solicitar productos a recoger en tienda.

El cliente podrá solicitar la cantidad de productos que requiera a su carrito de compra, realizar la compra y acceder a sus pedidos realizados para saber el estado en el que se encuentra. Por otro lado, el administrador podrá agregar, editar y cambiar disponibilidad de productos, así como consultar y modificar el estado de los pedidos que recibe.

Funciones de Producto

La aplicación tiene el objetivo de ser un canal de venta online para el supermercado, presentando los productos que se encuentran en venta para ello la aplicación permite:

- A nivel interno permite el almacenamiento de los productos, pedidos y usuarios que interactúan con la aplicación.
- A nivel de Administrador le permite agregar, editar y consultar tanto su catálogo de productos, como los pedidos a preparar.
- A nivel Cliente le permite registrarse, iniciar sesión, consultar productos, almacenar productos en su carrito de compra, realizar compra y consulta del estado de su pedido.

Características de los Usuarios

La aplicación está dirigida a público en general, esperando que los usuarios sean mayores de 12 años, con conocimientos básicos del uso de un computador principalmente y posteriormente de acceso a navegadores de internet para el acceso a la aplicación a través de la URL, también se requiere que cuenten con una cuenta de correo electrónico para poder realizar su registro de manera exitosa dentro de la aplicación.

Restricciones

La aplicación no permite la eliminación de productos únicamente cambiar el estado a no disponible siempre y cuando sea Administrador.

Solo permite la visualización de una imagen de producto en la vista de Detalles de Producto.

No genera ticket de compra por el pedido que realice el cliente.

La aplicación no permite el pago de compras dentro de la aplicación.

No se permite la configuración de perfil de cliente, es decir no se puede modificar el nombre, contraseña con los que se realizó el registro.

El cambio de imagen de producto no está disponible.

El historial de pedidos se muestra de manera general de modo que no se eliminan aquellos pedidos que se han finalizado.

No se encuentra disponible la aplicación para dispositivos móviles en esta primera versión.

Suposiciones y Dependencias

El sistema puede llegar a ser afectado por alguno de los siguientes puntos.

- Conexión de internet no activa en el dispositivo.
- Problemas de servidor.
- Fallas en la base de datos.
- Problemas de compatibilidad con navegadores.
- Problemas de dependencias de terceros.
- Falta de actualización a estándares de la web.

Requisitos Futuros

Como requisitos futuros para mejorar la herramienta que es esta aplicación tanto para el cliente como el administrador se tiene como requisitos a implementar en futuro lo siguiente:

- Implementar cobro de pedidos dentro de la aplicación.
- Disponibilidad de acceso desde dispositivos móviles.
- Estadísticas para el Administrador que le permitan identificar sus ventas por semana, así como la disponibilidad de sus productos para evitar posibles faltas a productos de mayor demanda.

Requisitos Específicos

La aplicación se encargará de presentar el catálogo de productos disponibles del supermercado, estos productos se podrán visualizar de acuerdo con la categoría a la que pertenezcan para poder preséntaselos a los clientes de una manera más organizada.

La aplicación permitirá diferentes acciones y vistas de acuerdo con el tipo de usuario con el que acceda, reconociendo dos tipos de usuario administrador siendo un único usuario y cliente siendo n cantidad de clientes.

El cliente podrá registrarse para poder consultar los productos, así como solicitar la cantidad que requieran en su carrito de compra personal, donde una vez tengan listo su pedido puedan realizar la solicitud correspondiente.

El cliente puede solicitar n cantidad de un producto en específico teniendo como limite la disponibilidad del producto, por que llegara a x cantidad sin más por solicitar.

El cliente tendrá una vista que le permita consultar su historial de pedidos, el cual le permite consultar cada pedido en específico de modo que pueda visualizar los detalles del pedido.

El administrador podrá visualizar todos los productos sin importar el estado de disponibilidad.

El administrador podrá agrega y modificar productos, permitiéndole incluso cambiar la disponibilidad de producto para que el cliente pueda o no tener acceso a él.

El administrador podrá consultar todos los pedidos realizados por los clientes.

El administrador podrá realizar cambios de estado de los pedidos que se verán reflejados en el panel de pedidos.

Interfaces Externas

Debido a la estandarización no regular en actualizaciones de los navegadores es posible que el sistema pueda verse afectado por la dependencia de la disponibilidad de elementos en cada uno de los navegadores. También puede verse afectada la interfaz de la aplicación por la diferencia de resoluciones existentes de pantallas de equipos de cómputo.

Funciones

El sistema permitirá el registro de los clientes por medio de su cuenta de Google o bien mediante correo electrónico de algún otro proveedor.

El sistema permitirá el inicio de sesión para el acceso a la aplicación.

El sistema permitirá la consulta de productos disponibles, en el caso del administrador tendrá accesos a todo producto y el cliente solo a los que se encuentren disponibles en tienda.

El sistema permitirá al cliente acceder a los detalles de producto.

El sistema permitirá al cliente agregar productos en n cantidad de lo que requiera y poder realizar el pedido de los productos contenidos en su carrito.

El sistema le permitirá a el administrador realizar cambios que se necesiten a los productos por lo que la vista detalla que se le presentará a el administrador será de edición.

El sistema permitirá consultar el historial de pedidos al cliente y administrador, así como consultar los productos a detalle de cada pedido.

El sistema le permitirá al administrador realizar cambio de estado a el pedido.

Requisitos de Rendimiento

La aplicación dependerá de la conexión a internet con la que cuente el dispositivo, además del tipo de dispositivo debido a que de momento solo se cuenta con la disponibilidad a dispositivos como laptop y equipos de escritorio. Por otro lado, depende de la disponibilidad del servicio de Google enfocado en la conexión a la Base de datos donde se almacena la información, lo que influye en el rendimiento.

Restricciones de Diseño

Se trabajará principalmente con enfoque en Paradigma de programación reactiva y funcional con el manejo de JavaScript y TypeScript principalmente como lenguajes de programación, pero también se emplea maquetado a través de HTML5 y estilización mediante CSS, todo lo anterior dentro del entorno de trabajo de React como librería principal para la creación de este proyecto.

Atributos del Sistema

El acceso a la aplicación será única y exclusivamente mediante el Inicio de sesión, con ello se podrá tener acceso a el panel de Inicio personalizado, con las funcionalidades acordes al tipo de perfil. De modo contrario únicamente se podrá consultar los productos, pero sin ninguna opción más, el sistema se encargará de solicitar registro o inicio de sesión siempre.

Capítulo 4

Análisis de Requerimientos

En este capítulo se muestra la etapa de análisis para el desarrollo del proyecto en el cual se presenta el diagrama de casos de uso, así como la definición de cada uno de ellos lo cual permite identificar la interacción entre actores y la aplicación permitiendo un panorama más claro que lo que se realiza en cada uno de ellos.

Diagrama de Casos de Uso

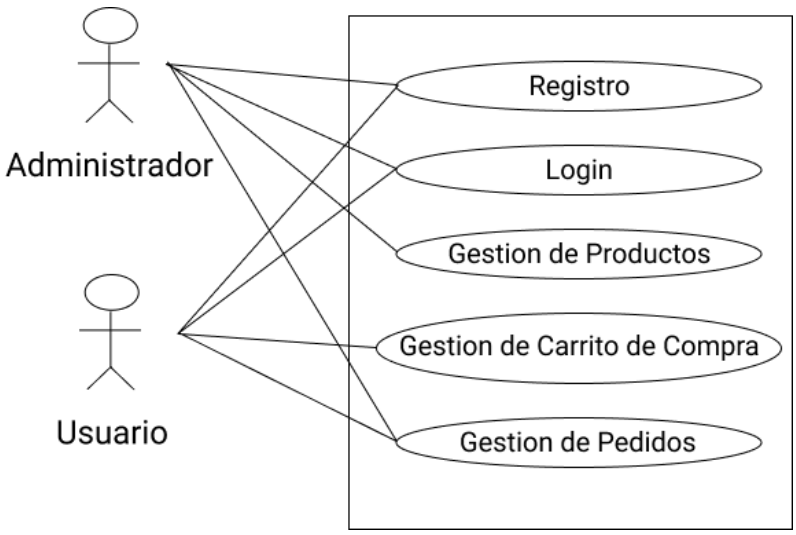

Figura 4-1 Diagrama de Casos de Uso

La Figura 4-1 presenta el Diagrama de Casos de uso en el que se identifican como actores dentro del sistema a Administrador y Usuario, estos dos mantienen una interacción en la funcionalidad del sistema de acuerdo con el flujo de trabajo que realicen.

Descripción de Casos de Uso

Descripción del Caso de uso "Registro"
Identificación: Nombre del Caso de uso: Registro Objetivo: Permitir al usuario crear una cuenta con la cual se le permita el acceso dentro de la aplicación de forma que pueda tener funcionalidad acorde al tipo de cuenta, así mismo se podrá personalizar su vista. Actor: Usuario Responsable: Jorge Leonel Muñoz Tehuitzil
Precondiciones: El usuario deberá contar con un correo y contraseña con los que se registraran, así como su nombre y apellido o bien una cuenta de Google activa.
Secuenciamiento: El caso de uso inicia cuando el usuario requiere crear una cuenta para hacer uso de la funcionalidad del software, permitiéndole realizar la compra de sus productos que ofrece el supermercado.
Encadenamiento Nominal: 1. El sistema muestra la vista login al usuario, donde se visualiza la opción de registro. 2. El usuario selecciona registro de las opciones dentro de la vista Login. 3. El sistema muestra la vista de Registro. 4. El usuario reconoce las dos maneras posibles de realizar su registro, mediante el completado de formulario o bien mediante su cuenta de Google 5. El sistema solicita nombre, apellido, email y contraseña. 6. El usuario completa los datos solicitados por el sistema 7. El usuario confirma él envió de información para creación de cuenta. 8. El sistema valida que este completa la información. 9. Valida la inexistencia de la cuenta. 10. Confirma la creación exitosa de la cuenta. 11. El cliente acepta la notificación. 12. El sistema navega a la vista login para que el usuario ingrese a su cuenta.
Encadenamientos Alternativos: A1: Crear cuenta con Google El encadenamiento inicia justo después de identificar los métodos distintos de crear una cuenta nueva, continuando en el punto 4 del encadenamiento nominal. 5. El usuario confirma crear cuenta mediante cuenta de Google. 6. El usuario inicia sesión o selecciona la cuenta con la que se creara la nueva cuenta en el sistema. 7. Confirma la creación de una nueva cuenta. 8. El sistema valida la cuenta de Google seleccionada. 9. El sistema notifica la creación exitosa de la cuenta 10. El cliente acepta la notificación 11. El sistema navega a la vista login para que el usuario realice su logueo.

Encadenamientos de error: E1: Email empleado es no valido El encadenamiento de error puede iniciar después del punto 9 del encadenamiento nominal. 10. El sistema notifica la invalidez de los datos ingresados para registro. 11. El usuario valida la información proporcionada al sistema. 12. El usuario reenvía información para registro. E2: Formulario incompleto El encadenamiento de error puede iniciar cuando se confirma realizar el registro en el sistema, punto 7 del encadenamiento nominal. 8. El sistema notifica error al realizar el registro. 9. El usuario acepta el error. 10. El usuario completa los datos faltantes. 11. El usuario reenvía el registro.

Tabla 4-1 Descripción del Caso de uso Registro

Descripción del Caso de uso "Login"
Identificación: Nombre del Caso de uso: Login Objetivo: Permite al usuario y administrador identificarse dentro de la aplicación web, permitiendo personalizar la apariencia de la aplicación junto con los permisos de cada actor. Actor: Usuario, Administrador Responsable: Jorge Leonel Muñoz Tehuitzil
Precondiciones: Usuario y Administrador contaran con un correo y contraseña con los que se registraron previamente dentro de la aplicación o bien una cuenta de Google.
Secuenciamiento: El caso de uso inicia cuando algún tipo de actor requiere identificarse para acceder a su panel personal y poder ejecutar las funciones que le provee el tipo de perfil creado.
Encadenamiento Nominal: 1. El sistema muestra la vista Login donde se muestras las diferentes opciones de inicio de sesión (correo electrónico o cuenta de Google) o bien realizar un registro de cuenta nuevo. 2. El internauta decide la manera de iniciar sesión según haya realizado su registro. 3. Ingresa correo electrónico. 4. Ingresa contraseña. 5. El internauta confirma iniciar sesión. 6. El sistema valida la cuenta y muestra el panel de acuerdo con el tipo de perfil.
Encadenamientos Alternativos:

A1: Iniciar sesión con cuenta de Google
El encadenamiento inicio justo después de que el sistema muestra la vista Login con las opciones de inicio de sesión disponibles para el internauta, es decir después del punto 1 del encadenamiento nominal

2. El internauta decide iniciar sesión con cuenta de Google.
3. Confirma iniciar sesión con cuenta de Google.
4. Selecciona y confirma la cuenta con la que inicia sesión.
5. El sistema valida la cuenta y muestra el panel de acuerdo con el tipo de perfil.

A2: No cuenta con cuenta registrada, crear nueva cuenta
El encadenamiento inicia después que el sistema muestre la vista Login con las opciones de inicio de sesión y en su caso registro de nueva cuenta, el punto 1 del encadenamiento nominal.

2. El internauta selecciona Registrarse como opción dentro de la vista.
3. El sistema confirma la opción y navega al internauta a una nueva vista de registro de cuenta.

Encadenamientos de error:

E1: Cuenta no existente
El encadenamiento de error puede iniciar después de que el sistema valida la cuenta confirmada como inicio de sesión en el punto 6 del encadenamiento nominal.

7 El sistema notifica que la cuenta validada no coincide con los datos registrados de inicio de sesión.
8 El internauta valida sus datos ingresados
9 El internauta reenvía su confirmación de inicio de sesión
10 El internauta reconoce no tener una cuenta registrada.

Tabla 4-2 Descripción del Caso de uso Login

Descripción del Caso de uso "Gestión de Productos"
Identificación: Nombre del Caso de uso: Agregar Producto Objetivo: Permitir al administrador poder agregar, actualizar y eliminar productos en la aplicación de su supermercado de forma que sus clientes puedan consultar y realizar sus compras respectivas. Actor: Administrador Responsable: Jorge Leonel Muñoz Tehuitzil
Precondiciones: El administrador deberá contar con información respectiva a sus productos para crear nuevo o bien actualizar información dentro del sistema.
Secuenciamiento: El caso de uso iniciar cuando el Administrador requiere realizar alguna función dentro del sistema respectivo a cambios en sus productos.
Encadenamiento Nominal:

1. El sistema muestra el panel principal de administrador.
2. El sistema lista los productos y muestra las opciones de gestión de productos.
3. El administrador selecciona crear nuevo producto.
4. El sistema muestra vista de nuevo producto con formulario a completar por el administrador.
5. El sistema solicita nombre, precio, descuento, cantidad disponible, categoría, descripción, estado, imagen.
6. El administrador completa los datos solicitados.
7. El administrador confirma la creación de nuevo producto.
8. El sistema valida la información enviada.
9. El sistema confirma la creación exitosa del nuevo producto.

Encadenamientos Alternativos:

A1: Eliminar Producto
El encadenamiento puede iniciar después de visualizar las opciones de gestión en la vista del administrador, punto 2 del encadenamiento nominal.

3. El administrador consulta los productos que tiene disponibles dentro del sistema.
4. El administrador selecciona el producto a modificar.
5. El sistema muestra los detalles actuales del producto.
6. El administrador selecciona la opción de eliminar el producto.
7. El sistema muestra mensaje de confirmación.
8. El administrador confirma la eliminación
9. El sistema elimina el producto.
10. El sistema indica que se ha elimina el producto correctamente.

A2: Actualizar Producto
El encadenamiento puede iniciar después de visualizar las opciones de gestión en la vista del administrador, punto 2 del encadenamiento nominal.

3. El administrador consulta los productos disponibles dentro del sistema.
4. El administrador selecciona el producto a modificar.
5. El sistema muestra los detalles actuales del producto.
6. El administrador actualiza los campos necesarios del producto.
7. El administrador confirma actualizar los cambios del producto.
8. El sistema valida se encuentre completa la información.
9. El sistema guarda los cambios realizados por el administrador.
10. El sistema confirma el guardado exitoso de los cambios.

A3: Consultar Producto
El encadenamiento puede iniciar después de visualizar las opciones de gestión en la vista del administrador, punto 2 del encadenamiento nominal.

3. El administrador identifica el buscador de productos.
4. El administrador ingresa el nombre del producto a consultar.
5. El sistema actualiza la lista de productos acorde a las coincidencias de productos con la entrada de datos.
6. El administrador selecciona el producto.
7. El sistema presenta los detalles del producto.

Encadenamientos de error: E1: Formulario incompleto El encadenamiento inicia después del paso 8 del encadenamiento nominal una ves es validada la información proporcionada por el administrador. 9 El sistema notifica los campos no completados por el administrador. 10 El administrador completa la información requerida. 11 El administrador valida la información enviada. 12 El administrador reenvía la información para nuevo producto.

Tabla 4-3 Descripción del Caso de uso Gestión de Productos.

Descripción del Caso de uso "Gestión de carrito de compra"
Identificación: Nombre del Caso de uso: Agregar producto a carrito Objetivo: Permitir al usuario seleccionar cada uno de los productos que va a integrar a su compra, de manera que pueda indicar la cantidad de producto y agregarlo, también podrá eliminar productos de su carrito. Lo anterior mantendrá actualizado el total de su compra. Actor: Usuario Responsable: Jorge Leonel Muñoz Tehuitzil
Precondiciones: El usuario deberá estar registrado y tener su sesión iniciada, esto le permitirá tener su carrito de compra y demás acciones dentro de la aplicación. En el caso de eliminar se requiere tener productos dentro de su carrito.
Secuenciamiento: El caso de uso inicia cuando el usuario tiene a la vista algún producto que requiere comprar o requiere eliminar producto de su carrito actual.
Encadenamiento Nominal: 1 El usuario identifica o consulta el producto que requiere comprar. 2 El usuario selecciona el producto. 3 El sistema muestra la vista de detalle de producto. 4 El usuario selecciona la cantidad de producto que requiere comprar. 5 El sistema valida la cantidad requerida por el usuario. 6 El usuario confirma agregar al carrito el producto. 7 El sistema notifica que se ha agregado el producto con éxito.
Encadenamientos Alternativos: A1: Eliminar producto de carrito El encadenamiento puede inicia antes de identificar el producto a comprar, punto 1 del encadenamiento nominal. 0 El usuario selecciona el botón de carrito de compra 1 El sistema muestra al usuario el carrito de compra con la lista de productos y costo acumulado actual. 2 El usuario identifica el producto a eliminar.

3 El usuario da clic en eliminar del producto. 4 El sistema actualiza la lista y costo actual del carrito. A2: Comprar productos de carrito El encadenamiento puede iniciar después de que el sistema muestre la lista de productos y costo acumulado actual en el carrito, punto 1 del encadenamiento alternativo 1. 2 El usuario corrobora los productos del carrito. 3 El usuario selecciona el tipo de pedido . 4 El sistema actualiza el costo de acuerdo con el tipo de pedido. 5 El usuario da clic en comprar 6 El sistema valida la cantidad solicitada de cada producto solicitado 7 El usuario envía la información 8 El sistema confirma el pedido con éxito y muestra la confirmación con un numero de referencia de pedido.
Encadenamientos de error: E1: Cantidad de producto no existente para agregar producto a carrito El encadenamiento de error puede iniciar después de que el sistema valida la cantidad requerida por el usuario en el punto 5 del encadenamiento nominal. 6 El sistema notifica la inexistencia de la cantidad requerida 7 El usuario reduce la cantidad solicitada de producto 8 El usuario confirma agregar el producto requerido 9 El sistema notifica que se ha agregado correctamente. E2: Cantidad de productos no existente para realizar la compra. En encadenamiento de error puede iniciar después de que el sistema valida la cantidad de cada uno de los productos en el carrito, punto 6 del encadenamiento alternativo 2. 7 El sistema interrumpe el envío de pedido, notifica el producto con cantidad no disponible. 8 El usuario modifica la cantidad del producto con cantidad inexistente. 9 El usuario confirma la compra de pedido. 10 El sistema valida la cantidad de cada producto solicitado. 11 El sistema muestra los formularios correspondientes. 12 El usuario envía la información. 13 El sistema valida la información. 14 El sistema confirma con éxito el pedido mostrando un numero de referencia.

Tabla 4-4 Descripción del Caso de uso Gestión de carrito de compra

Descripción del Caso de uso "Gestión de pedidos"
Identificación: Nombre del Caso de uso: Gestión de pedidos Objetivo: Permitir al usuario visualizar los pedidos realizados sin recoger, de manera que pueda ver los productos de la compra el total y estado actual. En el caso del administrador

puede visualizar los pedidos que tiene por entregar, puede modificar el estado para poder notificar al cliente cuando esté listo para entregar. Actor: Usuario, Administrador Responsable: Jorge Leonel Muñoz Tehuitzil
Precondiciones: El usuario requiere tener una cuenta dentro del sistema para poder acceder a la zona de pedidos.
Secuenciamiento: El caso de uso inicia cuando es seleccionada la opción de pedidos dentro del sistema para ambos actores (Administrador y Usuario).
Encadenamiento Nominal: Administrador: 1 El sistema muestra la vista de pedidos. 2 El administrador selecciona el pedido a visualizar a detalles. 3 El sistema muestra la vista de detalle de pedido con los productos, estado y total. 4 Si lo requiere el administrador actualiza el estado del pedido. Usuario: 1 El sistema muestra la vista de pedidos. 2 El usuario selecciona el pedido a visualizar a detalle. 3 El sistema muestra la vista de detalle de pedido con productos, estado y total.
Encadenamientos Alternativos:
Encadenamientos de error: E1: Lista de pedidos vacía. El encadenamiento puede iniciar después de que el sistema muestre la vista de pedidos, punto 1 del encadenamiento nominal 1. 2 El sistema muestra notificación de lista de pedidos vacía. 3 El usuario no puede seleccionar pedidos.

Tabla 4-5 Descripción del Caso de uso Gestión de pedidos.

La interfaz y la experiencia de usuario son aspectos fundamentales para el éxito de una aplicación, buscando ofrecer a los usuarios más que una aplicación funcional, para ello en esta sección serán descritas las técnicas que se realizaron para llegar al diseño final, entre estas técnicas se encuentran teoría del color, auditoria de aplicaciones similares, mapa heurístico, maquetado y presentación de recursos empleados.

Auditoria de aplicaciones similares

La auditoría de aplicaciones similares tuvo como objetivo consultar y analizar aplicaciones existentes con fines similares a los del proyecto de esta tesis. Estas auditorias permitirán obtener algo preliminar al resultado esperado desde la distribución del espacio, colores enfocados a su área de servicios o bien el manejo o cumplimiento de sus casos de uso.

Las aplicaciones seleccionadas para realizar su análisis son Mercado Libre y Walmart siendo plataformas con objetivo en la venta de artículos de diversas categorías. De entrada inicial al acceder a la raíz de la plataforma sin iniciar sesión como usuario del sistema nos presenta secciones similares por lo que se toman como referencia para estar presentes en el proyecto las cuales son de importancia en los sitios, pueden presentarse con estilos diferentes pero siempre presentes y estás son un header o encabezado el cual presenta los logos de la marca, un buscador, acceso al carrito de compra, inicio se sesión para identificarse y menú de acceso a diferentes categorías principales dentro del sistema lo que permite la navegación a las diferentes paginas dentro del sitio. También se identifica un carrusel en esta página de inicio en la cual se presentan principalmente ofertas de temporada o eventos próximos como el buen fin, hot sale o evento de la propia empresa. Esto se puede identificar en la Figura 4-2 y Figura 4-3

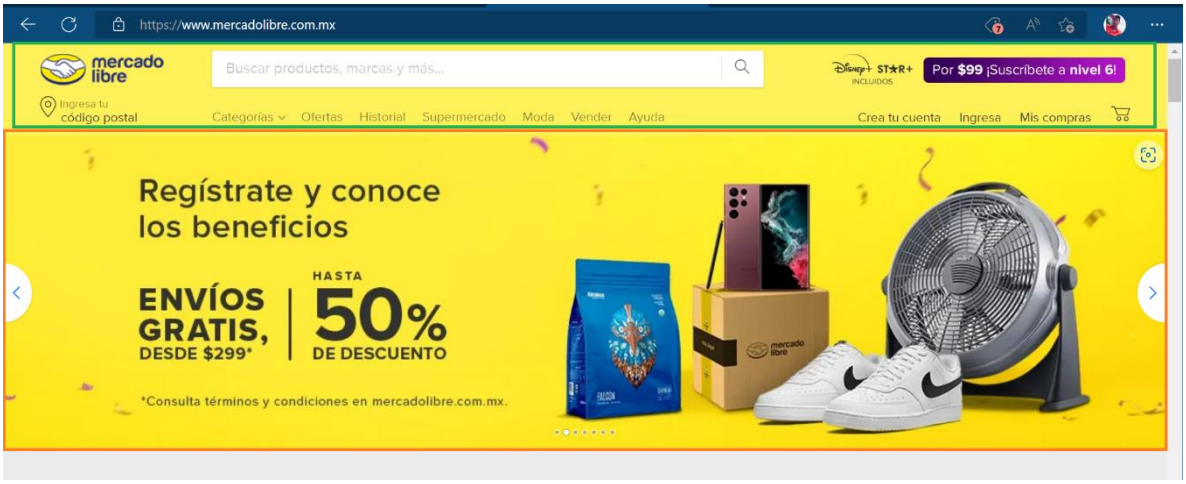

Figura 4-2 Header y Carrusel de Mercado Libre

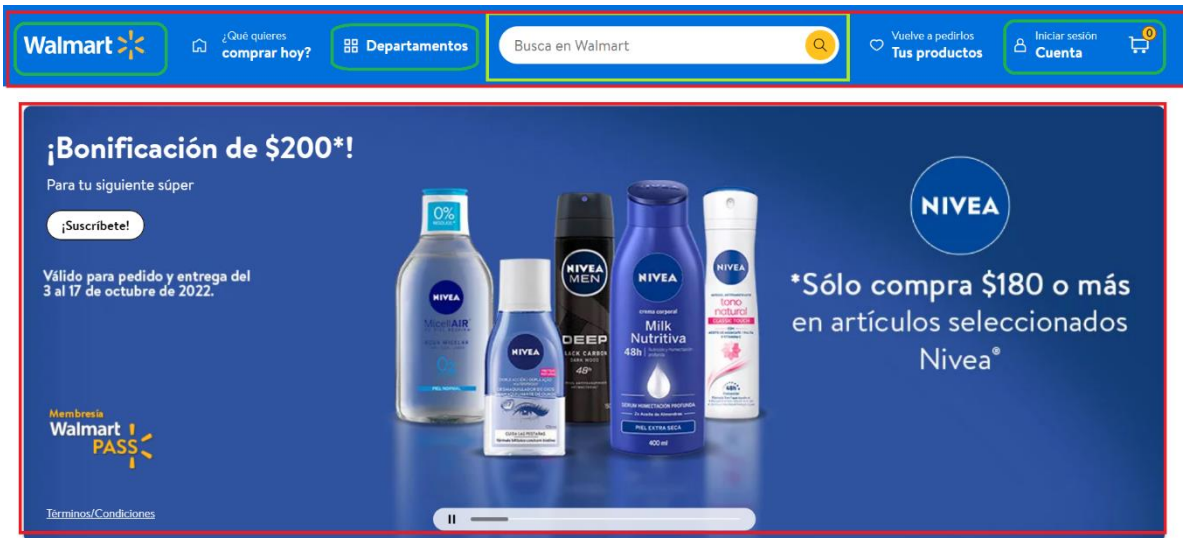

Figura 4-3 Header y Carrusel Walmart

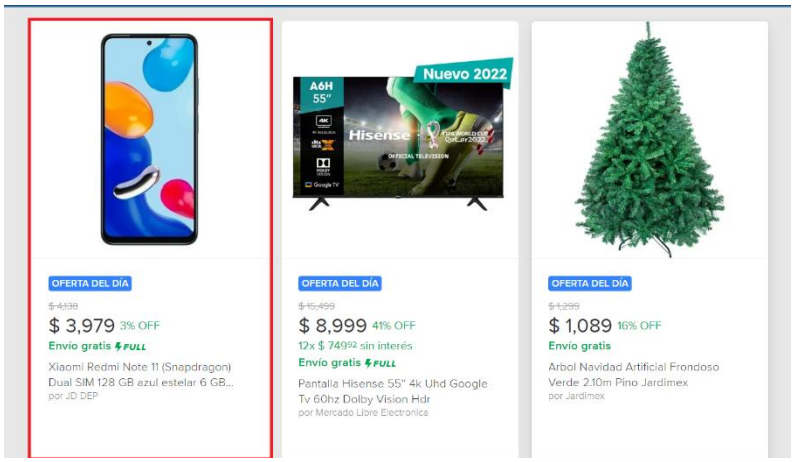

Figura 4-4 Card de Producto

La cards de los productos son las que atraen a los usuarios a realizar la compra en primera vista, en este caso las cards cuentan con imagen para identificar el producto, una etiqueta en este caso de detalle a resaltar como el caso de formar parte de las ofertas muestra el costo original del producto, su costo actual, nombre del producto, tipo de envió y el vendedor que oferta el producto.

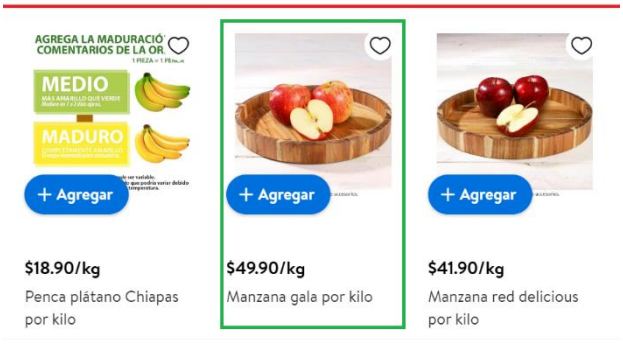

Figura 4-5 Card de producto Walmart

En el caso de la Figura 4-5 que muestra el manejo de Walmart podemos ver que es un estilo más simple, únicamente presentando la imagen, nombre y precio del producto.

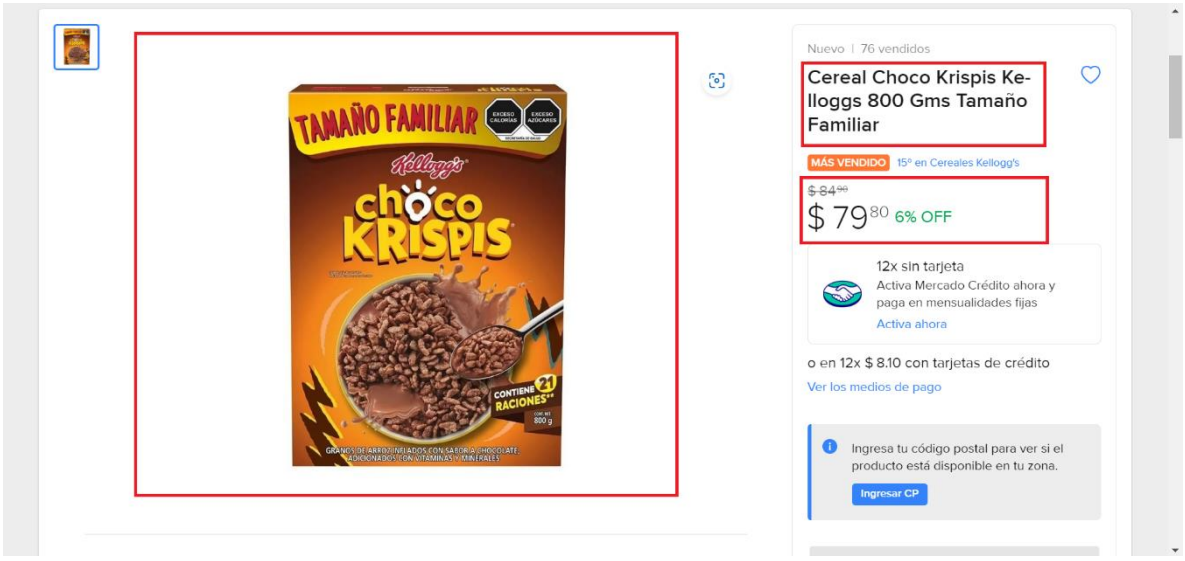

Figura 4-6 Detalles de Producto Mercado Libre

En los detalles de un producto (Figura 4-6) podemos observar características principales a considerar para la venta del producto que permiten al usuario identificar si es el producto adecuado. Entre estas características nos encontramos con el nombre o título de publicación del producto, precio a la venta, precio anterior a algún descuento actual si así fuera necesario e imágenes del producto. Otra parte importante es la Descripción del producto donde a mayor detalle se puede informar el usuario sobre detalles más precisos del producto como la presentación del producto, algunos de los aportes nutricionales y la cantidad de porciones que podemos obtener.

Esta vista de detalles de producto permite agregar la cantidad de producto que requeríamos a nuestro carrito de compras para posteriormente realizar la compra.

Finalmente tenemos en la Figura 4-7 el carrito de compra en esta vista se pueden observar 3 valores que ayudan en esta vista, el primero contiene una imagen del producto, nombre de producto, podemos incrementar la cantidad a comprar desde esta vista y el costo del producto. también se nos dan acciones como eliminar del carrito, comprar ahora o bien guardar para después y finalmente tenemos un total que es de nuestro carrito con ello sabremos el total de nuestra compra.

Figura 4-7 Carrito de Compra

Entre otros métodos que nos proporciona Mercado libre se encuentra la opción de poder dar seguimiento a nuestros pedidos y así saber en qué estado del proceso de envió se encuentra lo que nos da un panorama de cuando llega nuestro pedido.

Análisis de usuario

Se presenta un análisis realizado para identificar el público al que se dirige la aplicación con el fin de identificar características generales.

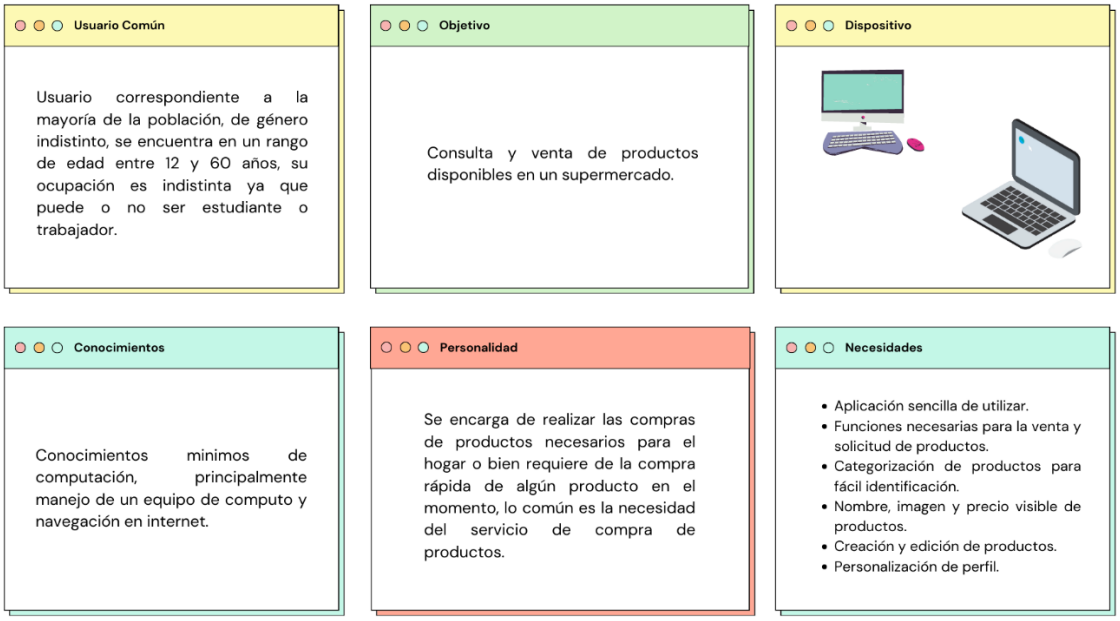

Figura 4-8 Arquetipo de personas.

Mapa heurístico para el intercambio de ideas

A través del mapa en la Figura 4-9 se busca identificar una representación visual del objetivo que se busca conseguir al desarrollar la aplicación.

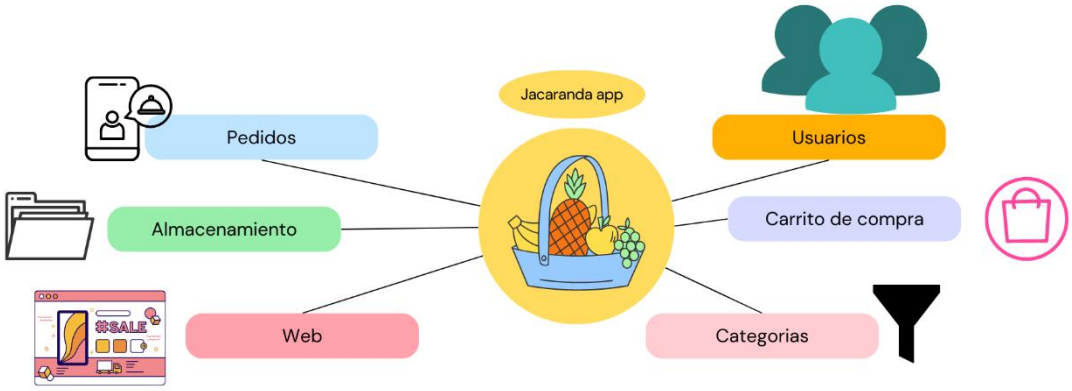

Figura 4-9 Mapa heurístico para el diseño de aplicación

Maquetado de interfaces

La vista de Inicio se definió como se muestra en la Figura 4-10 teniendo en la parte superior un header que contendrá el logo de la tienda, una barra de búsqueda para la consulta de productos mediante el nombre y en la esquina superior derecha se cuenta con el carrito de compra y el icono de perfil el cual tendrá el acceso a las funciones de cada tipo de usuario y así mismo el acceso a su cuenta.

Seguido del header se implementa un banner de imágenes el dónde se podrá presentar las ofertas de temporada mediante ilustraciones, así mismo ofertas de alguna categoría de productos o bien próximas ofertas.

En la parte central se integran el listado de categorías con productos correspondientes a las mismas presentados por card que contendrán una imagen, nombre de productos y precio. Las categorías que se presentan son las que define la tienda y en ellas son distribuidos los productos para que los clientes puedan identificar de forma más rápida los productos que requiere comprar.

Si se activa el evento de clic mediante las card de producto se podrá acceder a su vista de detalle, esta vista se comportará de acuerdo con el tipo de usuario sea cliente o bien el administrador. El administrador tendrá acceso a la vista de edición del producto mientras que el cliente tendrá una vista de detalle para consultar la información que le corresponde al producto.

En la parte inferior de la vista de inicio se integra un footer donde se incluyen algunos puntos de interés como: términos y condiciones, redes sociales, categorías, etc. Esto con el fin de tener mayor comunicación e información con la tienda en caso de alguna duda o aclaración.

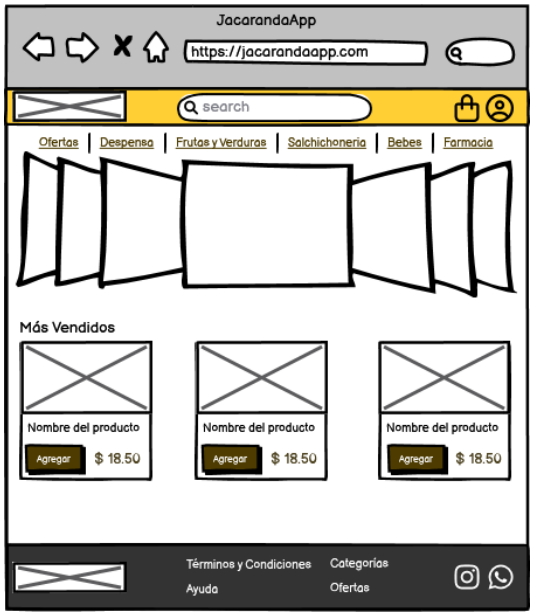

Figura 4-10 Vista "Inicio" de aplicación

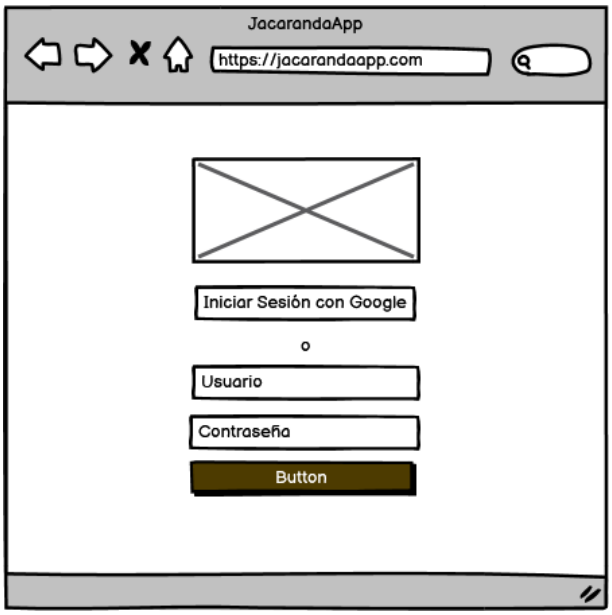

Figura 4-11 Vista "Inicio de Sesión"

La vista de inicio de sesión se realizó buscando lo más simple posible en este caso al centro se integran elementos como el logo de la aplicación, dos maneras de iniciar sesión como son:

- Inicio de sesión mediante cuenta de Google
- Inicio de sesión mediante correo electrónico

En ambos casos el usuario debe de ingresar su cuenta, aunque el proceso en diferente, en el caso de autenticación por cuenta de Google se requiere iniciar sesión lo que accederá a una vista fuera de la aplicación que permitirá acceder a su cuenta. En el caso del correo electrónico debe de coincidir el email ingresado con la contraseña proporcionada y dar clic en el botón de inicio de sesión para corroborar la autenticidad y así entrar a la aplicación de manera correcta reconociendo su usuario.

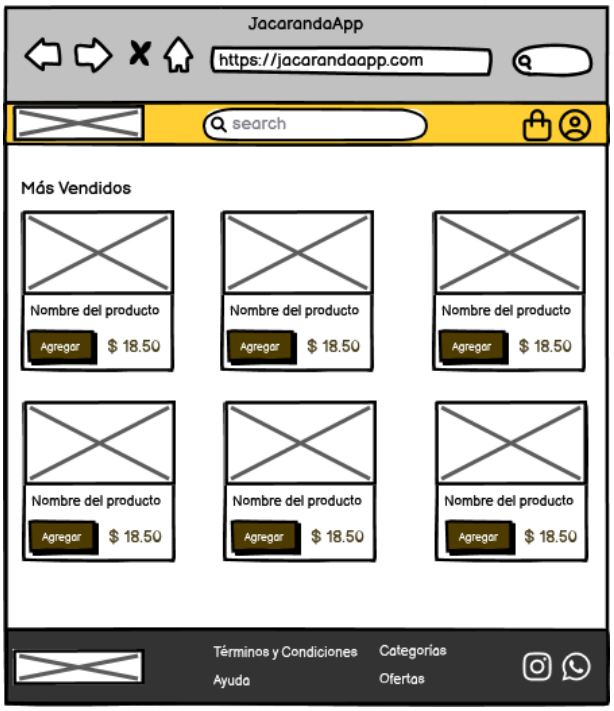

Figura 4-12 Vista "Categoría "

La vista Categoría de la Figura 4-12 su principal enfoque se encuentra en el centro donde se presentan todos los productos que encuentran disponibles dentro de cada categoría, esta vista será renderizada de acuerdo con la categoría a la que requiera acceder el usuario.

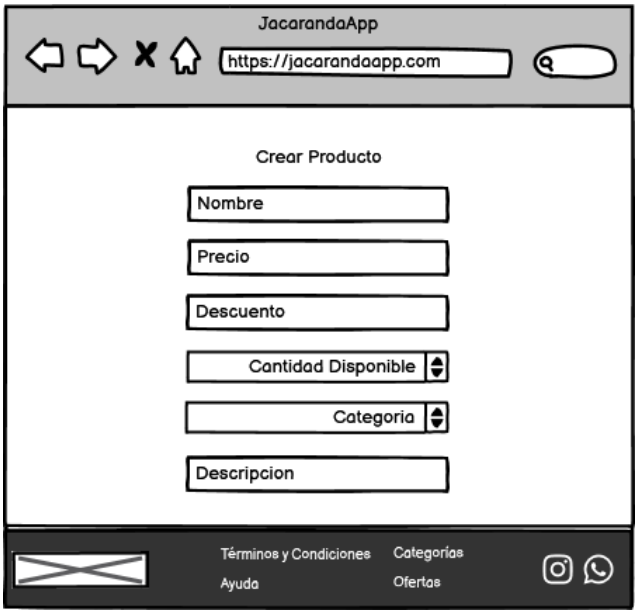

Figura 4-13 Vista "Nuevo Producto"

La Figura 4-13 muestra la vista que presentara al administrador para realizar la creación de un nuevo producto para disposición de los pedidos de los clientes. El acceso a esta vista es exclusivo del Administrador y en ella se presenta un formulario el cual debe ser llenado con los datos del producto: nombre, precio, descuento aplicado, cantidad disponible de producto, descripción, categoría, imagen de producto y disponibilidad. Una vez completa la información necesaria el administrador podrá agregar el producto y verlo disponible en cuanto se tenga en base de datos.

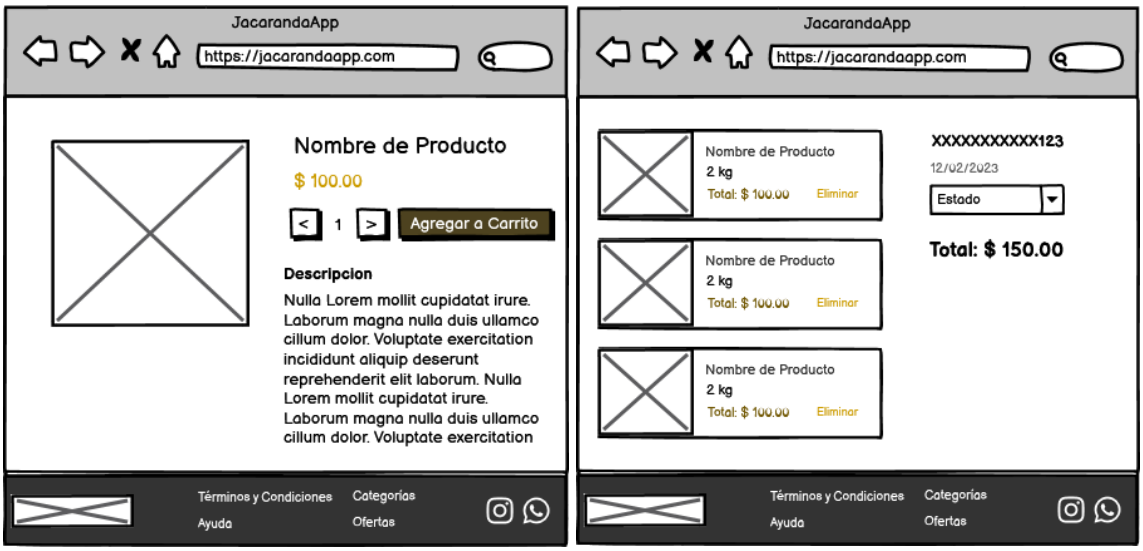

Figura 4-14 Izquierda: Vista "Detalle de Producto", Derecha: Vista "Detalle de pedido"

La Figura 4-14 presenta las vistas de detalle de Producto (Lado Izquierdo) esta vista es la que le permite al usuario lograr recabar mayor detalle del producto, en ella se le presenta el nombre completo del producto, el precio, se le permite seleccionar la cantidad de producto x que requiera, la descripción completa y su imagen de producto. La cantidad de producto que indique el cliente será la que se agregue al carrito de compra, cuando el cliente se encuentre seguro de su cantidad podrá agregar mediante un clic en el botón “Agregar a Carrito” y vista detalle de Pedido (Lado Derecho) es la vista que presenta los detalles del pedido realizado en ella se despliegan los productos que se solicitaron, con nombre, imagen, precio el código de pedido, la fecha en que se realizó, el estado en el que se encuentra y el total por el pedido.

Estas son algunas de las vistas de mayor importancia para el proyecto, en capítulos más adelante se presentarán las vistas finales donde se podrá ver el resultado final de este diseño planteado en este capítulo.

Paleta de colores

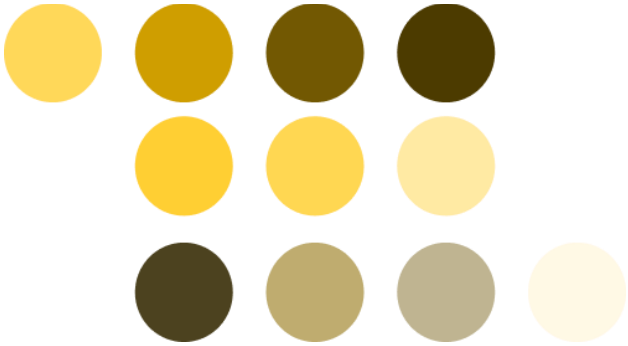

Figura 4-15 Paleta de colores.

De acuerdo con la teoría del color el amarillo es un color brillante y vibrante que se asocia comúnmente con la felicidad, la energía y la tranquilidad mental. En el contexto de las aplicaciones de venta de productos en supermercados, el color amarillo se puede utilizar para atraer la atención del cliente y crear una sensación de frescura y vitalidad por ello es que se tomó la decisión de emplear el color amarillo como color base y de generar la paleta de colores de la Figura 4-15 explorando tonos que van del blanco al negro, pero con tonalidad amarrilla de modo que sea lo más accesible dentro de la aplicación.

Interactividad

La interactividad dentro de una aplicación nos identifica las maneras en que los usuarios interactúan con la aplicación, es decir las maneras con la que se puede accionar la aplicación y dar paso a un flujo de trabajo al usuario. La interactividad nos permite considerar puntos al momento de diseñar la aplicación para obtener una funcionalidad intuitiva y sencilla. A continuación, se presentan algunos puntos considerados para el diseño de la aplicación.

La aplicación requiere tener a la vista y de mayor acceso para el usuario **zonas de prioridad** donde se mantengan recursos para su interacción del usuario con la aplicación, en el caso de la aplicación web se identifican como zonas prioritarias:

- Header que es la zona superior, esta zona es importante porque ayuda a los usuarios a orientarse en la aplicación y encontrar lo que están buscando, siempre contiene menús de navegación y barras de búsqueda.
- Contenido principal: zona central de la aplicación donde se presentan los productos, esta zona es donde el usuario mantiene la interacción por ser donde se encuentra su razón de entrar a la aplicación, la consulta de categorías y productos para solicitar a la tienda.
- Barra lateral derecha: esta barra lateral solo se presenta cuando el usuario accede a su carrito de compras el cual se presenta del lado derecho. Aquí el usuario puede

visualizar el listado de productos contenido en él, el total por todos los productos y puede realizar eliminación de productos si así lo desea.

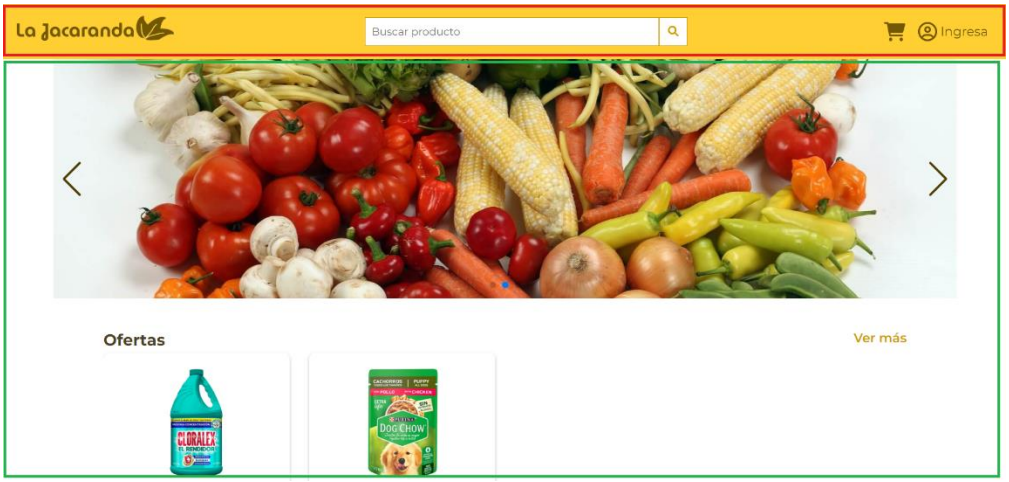

Figura 4-16 Zonas Header (Rojo) y Contenido Principal (Verde).

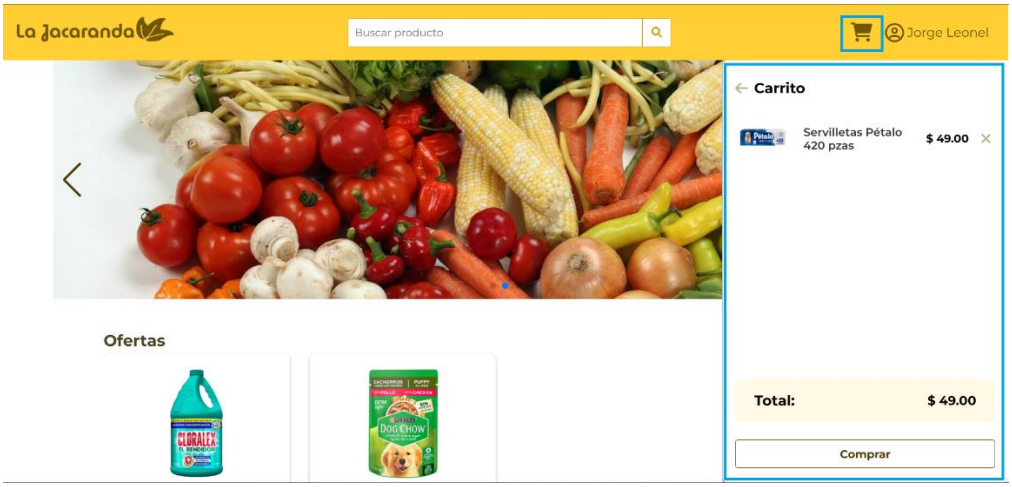

Figura 4-17 Zona Barra lateral (Azul)

Dentro de la aplicación las maneras de interaccionar que se pueden identificar dependerán del dispositivo del cual se conecte el usuario ya sea una computadora de escritorio o una laptop por el momento, una vez agregada la parte responsive se podrá también considerar los dispositivos móviles.

Para el caso de la computadora de escritorio la interacción se logra exclusivamente mediante clic por medio del ratón y scroll que puede ser por el ratón o bien mediante las flechas de desplazamiento del teclado. En el caso de una laptop se pueden emplear los métodos mencionados anteriormente más el uso de touch a través del panel táctil que refuerza el clic (tap) y scroll.

Estas interacciones son las más sencillas y tradicionales dentro de la web permitiendo el control dentro de aplicaciones y sitios a los que acceda el usuario.

- **Tap/Clic**: Inicia acciones mediante los botones, abre menús, accede a los enlaces y cambia las vistas de la aplicación.
- **Scroll**: Permite al usuario desplazarle por la aplicación en el contenido central que se encuentra dentro del viewport logrando consumir más elementos.

Gestión del espacio

La identificación de **puntos focales** es una parte importante para mantener al usuario activo dentro de la aplicación de modo que sea inmediata su atención a los elementos que le permiten realizar acciones o el modo en que se logran ejecutar, para ello es que, mediante el color, pequeñas animaciones y aplicación de sombras permiten resaltar estos elementos a la vista del usuario como son los botones, cards de productos e incluso los iconos que representan acciones (Figura 4-18).

Figura 4-18 Puntos focales

Los **llamados a la acción** serán los encargados de persuadir a los usuarios a que realicen acciones dentro de la aplicación es por ello por lo que su diseño debe ser claro, atractivo y conciso con su acción. Algunas de las llamadas a la acción dentro de la aplicación son:

- Botones con texto que reflejen su acción de manera clara, alternando diseño en caso de existir dos botones. Es posible integrar hover como herramienta para identificar un botón secundario. En la Figura 4-19 se ilustran dos botones el principal para el registro con correo electrónico y el secundario de momento para registrase con Google.

Figura 4-19 Llamado a la acción en Botón

- Botones con iconografía especifica a su función logrando también con ello el **Affordance** haciendo predecibles los cambios que se visualizaran.

Figura 4-20 Llamado a la acción en botones con iconos

- Enlaces identificados con color de texto distinto de forma que se resalte.

Seguimiento de Pedidos

Id	Estado	Tipo Pedido	Fecha	Total
BFiQ0K2ECg3ASIgFPfJC	Finalizado	En Tienda	14/3/2023	$ 18.00
ZzVPoym2zX3ss9BVuEZ8	Para recoger	En Tienda	15/3/2023	$ 133.00
c707ipVrtUFjjLBlzpuO	En espera	En Tienda	13/3/2023	$ 42.00
j06VykPN5volPDqoXMfq	Finalizado	En Tienda	2/3/2023	$ 22.00
l5WkbgL1kbFbjc3WqH2M	En espera	En Tienda	13/3/2023	$ 129.00
pf6UH5QdnU5Sy4I3JryQ	En espera	En Tienda	1/3/2023	$ 87.00

Figura 4-21 Llamado a la acción a través de enlaces.

Capítulo 5

Antes de comenzar se presenta información de las herramientas empleadas para el desarrollo del proyecto, entre las que destacan React, Firebase, TypeScript, Javascript y Github Pages

React

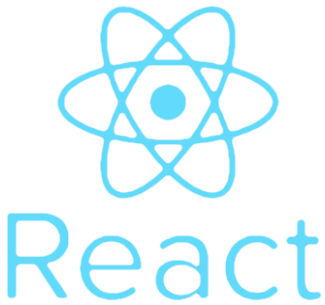

Figura 5-1 React

React se define como una librería de JavaScript, aunque con ciertas adaptaciones de elementos se puede lograr convertirlo a un framework. React permite construir interfaces de usuario de una manera sencilla, con la creación de diferentes vistas que representen el estado de la aplicación, se puede obtener los cambios que resulten ante la interacción que realicen los usuarios dentro de la aplicación.

Su base es el manejo de componentes encapsulados los cuales manejan su estado de forma independiente de esta manera se logra controlar y renderizar los cambios sin necesidad de reescribir código y esto gracias a el manejo que se realiza a través de hooks que proporciona como: useState(), useEffect(), en el caso de la navegación nos apoya con react-router-dom y otros recursos para el funcionamiento adecuado.

Firebase

Firebase es una plataforma proporcionada por Google que permite la compilación y desarrollo de aplicaciones tanto móviles como web mediante diferentes servicios que proporciona, lo que permite lograr aplicaciones con datos en tiempo real. Algunos de los servicios que proporciona Firebase permiten el almacenamiento de información, notificaciones, autenticación, base de datos, análisis y hospedaje incluso a través de su hosting [4].

Firebase Authentication se encarga de realizar autenticación de manera segura, logrando mejorar la experiencia de integración y acceso para los usuarios finales, permite integrar autenticación a través de email y contraseña, Facebook, Twitter, Google, Github e incluso

número telefónico. Este servicio es integrado a el proyecto para cumplir el objetivo de registrar e iniciar sesión a los usuarios.

Cloud Firestore es una base de datos NoSQL que se rige por colecciones y documentos siendo los documentos los que se encargan de contener la información de cada registro de objetos que se requieran, mientras que las colecciones son quienes almacenan a estos documentos, es posible crecer nuestra aplicación sin problema pues se logra crear jerarquías que ayuden a mantener un control de la información.

Como tercer y último recurso de Firebase se empleó **Cloud Storage** quien es el encargado del almacenamiento de archivos para asegurar rapidez y seguridad al almacenar y obtener recursos que se emplean dentro de la aplicación como son las imágenes de producto.

TypeScript

Es un lenguaje de programación de código abierto desarrollado por Microsoft, este lenguaje de programación no es más que un superset de JavaScript, es decir, proporciona una solución a las limitaciones de las cuales carece JavaScript principalmente el tipado de datos dando una sugerencia de tipo al parámetro o variable que se defina, logrando identificar que sea ejecutado de manera adecuada obteniendo un resultado satisfactorio [5]. De igual manera se logra una mejor documentación que permite a desarrolladores futuros entender nuestro código sin problema alguno.

Como aporte adicional TypeScript brinda el manejo de interfaces, Clases abstractas, Sobrecarga de Funciones, Valores opcionales, entre otros elementos.

Github Pages

Github Pages es un servicio de alojamiento proporcionado por GitHub el cual toma archivos HTML, CSS y JavaScript que se encuentran en un repositorio dentro del mismo GitHub, para ser visible ejecuta un proceso de compilación de manera que se publique adecuadamente el sitio. En cuanto al dominio puede ser el proporcionado de GitHub el cual no tiene ningún costo o bien un dominio personalizado que se adquiere con algún proveedor y después se configura dentro de GitHub [1].

El beneficio de emplear GitHub Pages es que es gratuito y no tiene un límite de alojamiento de sitios, cada sitio cuenta con su propio repositorio al cual a través de comando de Git y GitHub podemos subir cambios de manera inmediata, aunque como limitante seria la cantidad de solicitudes que se realizan al api de GitHub para realizar estos cambios.

Desarrollo

El proceso de desarrollo se puso en marcha una vez fue completada la fase de análisis y diseño de la aplicación para tener una clara idea de la interacción y funcionamiento que se requiere, así como la manera en la que se implementaran a código los requerimientos señalados en el capítulo 3 donde se menciona la descripción del proyecto. En este capítulo se presentan las partes importantes del proyecto en la fase de desarrollo, así como algunas capturas de código que permitan entender a mayor detalle lo que se obtiene con el tema principal de esta tesis, logrando implementar el proyecto.

El proyecto Jacaranda App se realizará mediante la librería descrita anteriormente (React) en conjunto con el superset de JavaScript para el manejo de tipado de datos lo cual se obtiene con TypeScript y la implementación de los servicios de Firebase apoyando conforme a sus beneficios. No se presentará todo código o vista creada en el proyecto únicamente se describen códigos que muestran la implementación de la Programación Reactiva y Funcional para identificar el beneficio o aplicación de este dentro del proyecto.

Funciones Puras

Una función pura es aquella que siempre devuelve el mismo valor con los mismos argumentos, llamando en un mejor termino lo anterior como *transparencia referencial* y su segunda regla es que no debe tener efectos colaterales o secundarios.

```
const createOrder = (payload: IOrder) => {

    let createdOrder: IOrder = {
        id: payload.id,
        idUser: payload.idUser,
        nameUser: payload.nameUser,
        email: payload.email,
        phone: payload.phone,
        status: payload.status,
        order: payload.order,
        date: payload.date
    }

    return createdOrder;

}
```

Figura 5-2 Función pura de nombre createOrder

En la Figura 5-2 se presenta la función createOrder siendo un ejemplo de Función Pura, esta función de acuerdo con el Objeto de tipo IOrder que reciba será el valor de retorno, si se ingresan los mismos datos se retorna un nuevo Objeto con la información y en este caso no muta un valor de tipo global solo se enfoca en las variables que contiene.

Programación Declarativa

La programación declarativa nos explica el que hay que hacer sin importar el cómo y para ello JavaScript nos aporta de funciones para recorrer elementos de un arreglo logrando tener un código que es más entendible y claro para su comprensión además de formar parte de la Programación funcional en Js. Estas funciones aportar a nuestro proyecto tener código más testeable y mantenible.

Las siguientes funciones cumplen con este objetivo:

- Map(): permite recorrer el arreglo original para retornar uno nuevo, se aplica una función determinada a cada elemento.
- Filter(): filtra los elementos a partir de una prueba devolviendo un nuevo arreglo
- Reduce(): retorna un único valor, que se obtiene de la sumatoria de alguna propiedad de los elementos del array
- Find() : retorna un elemento del arreglo que cumple cierta condición, se devuelve la primer coincidencia
- Sort(): Ordena los elementos de acuerdo con la regla de comparación proporcionada
- Some (): devuelve true si un elemento cumple la función proporcionada
- Every(): lo contrario a some() pues devuelve true si todos los elementos cumplen la función

Están función suelen ser integradas para facilitar el trabajo con los arreglos en Js, además de ello reducen el código, son más claras y entendibles para conocer lo que realizan.

Ahora bien, dentro del código del proyecto se presentan los siguientes ejemplos de código.

```
1  const productosCategory:IProduct[] = productos.filter((product:IProduct) => product.category === name) ;
2
```

Figura 5-3 Función Declarativa Filter()

En la Figura 5-3 se obtiene un array de Productos mediante la función filter() aplicada a él array de productos completo con el que se cuenta en tienda, este filter recorre los elementos del array buscando coincidencia con base en la categoría a la que pertenece el producto cada vez que se encuentra una coincidencia este es considerado al valor a devolver. Empleando esta función únicamente se observa que cada producto su atributo category es comparado con una variable name, toda coincidencia es devuelta en un arreglo.

```
const totalPrice:number = props.carrito.reduce((acumulador: number, producto: IProduct) => {
    return acumulador + parseFloat(producto.price.toString());
}, 0);
```

Figura 5-4 Función Declarativa reduce()

La función reduce() de la Figura 5-4 se encarga de obtener el total de precio por todos los productos contenidos en el carrito, en decir el costo total por la compra. En este caso se define acumulador como variable que almacena la sumatoria, se define el valor inicial que es 0 y recorre producto a productos sumando el parseo a el string Price de producto con el acumulador. Y finalmente se retorna este valor a la variable totalPrice.

Programación Reactiva

```tsx
const ProductDetail = () =>{
    const {productos,cart,setCart} = UserAuth();

    //definicion de estado de el contador
    const [count,setCount] = useState(0);

    //Identificacion de producto a mostrar
    const location = useLocation();
    const searchParams = new URLSearchParams(location.search);
    const productId = searchParams.get('id');

    //Obtencion de producto
    const product:IProduct = productos.find((product:IProduct) => product.id === productId);

    //Funcion que actualiza el contador
    const actualizarCount = (valorNuevo:any) =>{
        setCount(valorNuevo);
    };

    //Función que agrega el producto en la cantidad indicada en count
    const handleAddCart = () => {
        const newItems = Array.from({ length: count }, () => product);
        setCount(0);
        setCart([...cart, ...newItems]);
    }

    return (
        <div>
            <Header carrito={cart}/>
            <section className='container-productDetail'>
                <div>
                    <Gallery
                        urlImg={product.image.toString()}
                        tipo={'detalle'}/>
                </div>
                <div className='details'>
                    <h1 className='detail-title'>{product.name}</h1>
                    <h3 className='detail-price'>$ {product.price}</h3>
                    <div className='section-button'>
                        <Counter
                            stock={product.stock}
                            actualizaCount={actualizarCount}
                            contador={count}/>
                        <button
                            className='btn-primary detail-btn'
                            onClick={handleAddCart}>
                                Agregar carrito
                        </button>
                    </div>
                    <h1>Descripción</h1>
                    <p className='detail-description'>{product.description}</p>
                </div>
            </section>
            <Footer />
        </div>
    );
}

export default ProductDetail;
```

Figura 5-5 Implementación de Programación Reactiva en Página Detalle del Producto

```jsx
import React, { useState } from 'react';

import '../styles/Counter.css';

const Counter = ({stock, actualizaCount,contador}:any) => {

    //Funcion que resta a el contador -1
    const handleRemove = ()=>{
        let cont = contador-1;
        if(contador>0){
            actualizaCount(cont);
        }
    }

    //Funcion que incrementa contrador en +1
    const handleAdd = ()=>{
        let cont = contador +1;
        if(contador<stock){
            actualizaCount(cont);
        }
    }
    return (
        <div className='container-counter'>
            <button
                className='btn-counter counter-remove'
                onClick={handleRemove}
                arial-label='Remover'>
                -
            </button>
            <input
                className='input-counter'
                type="text" name="countProduct"
                id="countProduct" placeholder='0 pz'
                value={contador} readOnly/>
            <button
                className='btn-counter counter-add'
                onClick={handleAdd}
                arial-label='Agregar'>
                +
            </button>
        </div>
    );
}

export default Counter;
```

Figura 5-6 Implementación de Programación Reactiva en Componente Counter

La Programación Reactiva se centra en los flujos de datos y eventos asíncronos con el objetivo de construir aplicaciones más eficientes y escalables, dentro de este proyecto se emplea mediante hook´s de React siendo funciones que reaccionan ante los eventos ocurridos y ayudan en la gestión del estado.

En la Figura 5-6 se presenta el componente Counter el cual recibe tres variables como argumentos stock siendo la cantidad disponible de producto, actualizaCount función que permite actualizar el valor de count y la variable count valor actual de la variable la cual se muestra al usuario identificando la cantidad a solicitar del producto. Esta actualización de la variable count se realiza cada vez que los usuarios activan el evento den los botones de + y – que permiten el incremento y decremento del valor, el evento se ejecuta y se invoca la función para actualizar el estado.

Las tres variables recibidas en el componente Counter provienen de la página ProductDetail que se presenta en la Figura 5-5 donde se gestiona el estado actual del contador de productos a solicitar (Agregar al carrito). Además de count se gestiona de forma reactiva la obtención de id de producto a mostrar mediante *useLocation* y la gestión del usuario que se encuentra en sesión mediante *UserAuth* siendo el contexto de la Aplicación en donde se encuentran los productos de la tienda, la sesión activa y el carrito de compras actual.

Vistas Finales de Aplicación

En esta sección se presentan capturas de algunas de las vistas más relevantes dentro del proyecto.

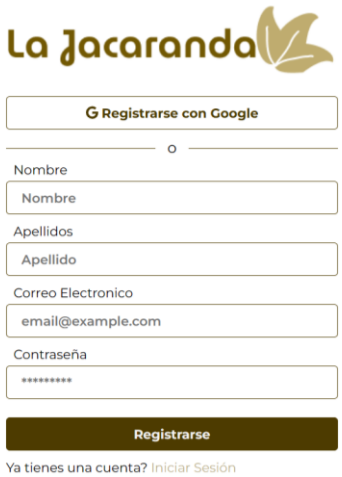

Figura 5-7 Vista de Registro

Figura 5-8 Vista Iniciar Sesión

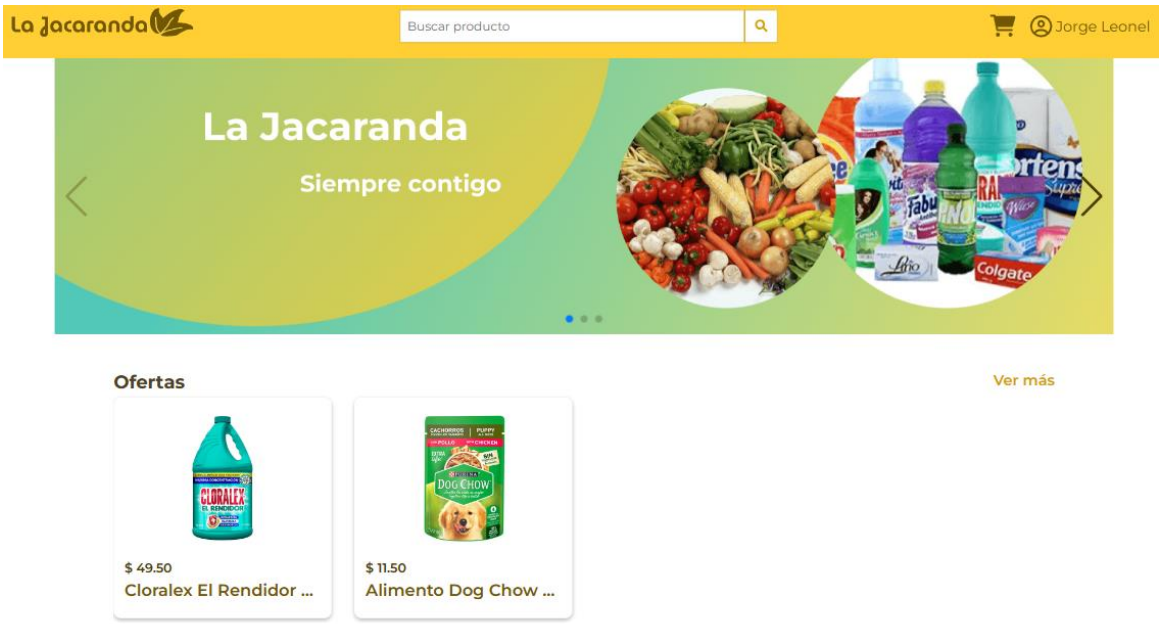

Figura 5-9 Vista de Inicio

Figura 5-10 Vista Detalle de Producto

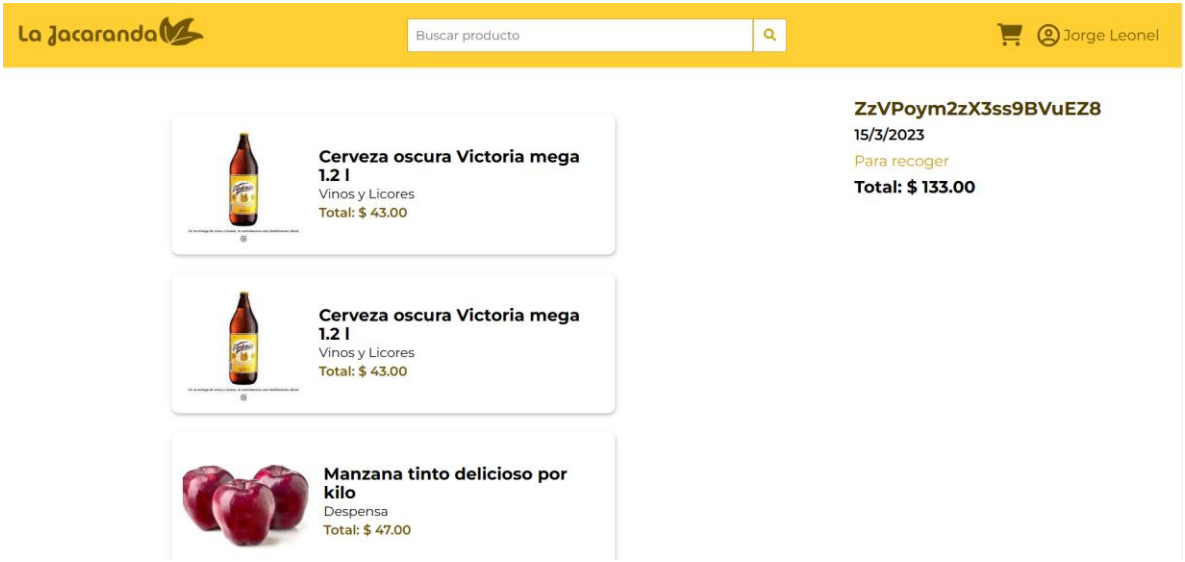

Figura 5-11 Vista Detalle de Pedido

Capítulo 6

Pruebas y Resultados

En estos capítulos se presentan algunas pruebas del código de se implementó con Programación Funcional de manera que se pueda comprender cuando se cumplen los puntos de la programación funcional y declarativa como se presentó en el capítulo anterior.

Para efectos de pruebas tenemos el siguiente código

```
const totalPrice:number = props.carrito.reduce((acumulador: number, producto: IProduct)
    => {
        return acumulador + parseFloat(producto.price.toString());
    }, 0);
```

Figura 6-1 Prueba 1 con función que obtiene el precio total del conjunto de productos en carrito de compra

Y se tendrá como arreglo props.carrito al siguiente conjunto de datos:

```
const props.carrito: IProduct[] = [
        { name: "Producto 1", description: "Descripción del producto 1", price: 10 },
        { name: "Producto 1", description: "Descripción del producto 1", price: 10 },
        { name: "Producto 2", description: "Descripción del producto 2", price: 20 },
        { name: "Producto 3", description: "Descripción del producto 3", price: 30 },
      ];
```

Figura 6-2 Ejemplo de conjunto de datos en props.carrito

De acuerdo al código en la Figura 6-1 la función reduce() recibe “acumulador” variable que se inicializa en 0 pero que lleva el valor acumulado a devolver de la suma de cada producto y toma producto a producto de props.carrito de tipo IProduct, con esos dos argumentos retorna el valor de una suma entre acumulador + el valor resultante del parseo de producto.price.toString(). Esto resultaría en lo siguiente

Index Producto	0	1	2	3	
Producto.Price	10	10	20	30	
acumulador	0	10	20	40	70

Figura 6-3 Tabla de Ejecución en código con valor en variables

Como se ilustra en la Figura 6-3 se puede seguir el valor de acumulador y el cambio de producto, en este caso el valor del producto en n índice del array, después de sumar acumulador con el precio de producto actual se retorna el valor.

En cuanto a las reglas de la programación funcional que anteriormente se presentaron (transparencia referencial y no tener efectos colaterales) en este caso podemos dar por hecho que si se ingresa una colección con los mismos productos se obtendrá el mismo resultado, no existen ningún daño colateral pues solo se retorna el valor obtenido en acumulador con el ultimo producto, pero no se muta ninguna variable ni se modifica ningún valor de ámbito global. Lo que concluye en satisfacer ambas condiciones.

La siguiente prueba va enfocada en el funcionamiento de la programación reactiva específicamente se trabaja sobre la vista de ProductDetail que se muestra en la Figura 6-4.

Figura 6-4 Vista detalle de Producto

La parte reactiva se encuentra en el componente Counter que se encuentra contenido en la vista que se muestra en la figura anterior debajo del precio y al lado opuesto al botón "Agregar carrito". Esta vista comparte como argumentos al componente la función que actualiza el contador mostrado que se encuentra en 0, el stock que es la cantidad de producto disponible siendo el máximo para comprar y el valor actual de count valor que se muestra.

El usuario decide ejecutar el botón de clic a el botón más (+, indicado en circulo color rojo) de counter el cual ejecuta el evento, realiza una suma del valor actual de count + 1, se compara con el stock disponible y si es menor se actualiza el estado de variable count lo que hace un cambio en la vista como se muestra en la Figura 6-5.

Figura 6-5 Cambio de estado a variable count

Cuando se llegue al tope no se verán cambios en la vista puesto que no habrá más producto disponible.

Figura 6-6 Componente Carrito vacío.

Ahora bien, cuando se tiene la cantidad requerida de producto el usuario lo agrega al carrito mediante un clic en el botón “Agregar carrito” y se actualiza tanto la vista ProductDetail en el componente counter a 0 y el componente de carrito. En la Figura 6-6 se muestra el componente carrito vacío y en la Figura 6-7 se muestra actualizado con el producto y total por todo su contenido, además del valor 0 en el componente counter.

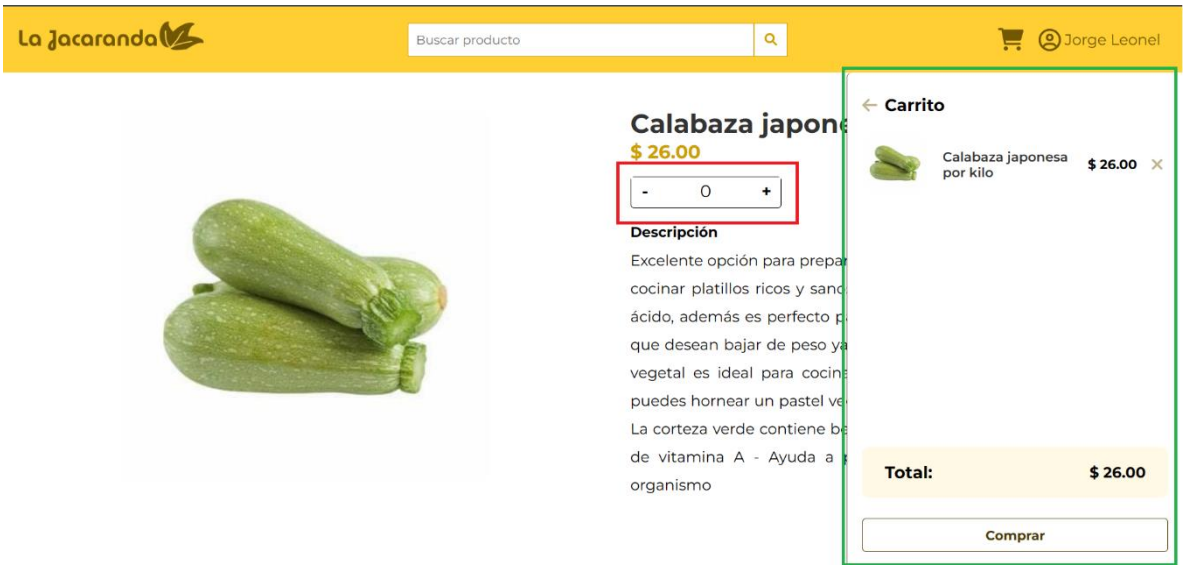

Figura 6-7 Cambio de estado.

El estado se actualiza al ejecutar el evento de clic al botón y se realiza el cambio de estado a los componentes necesarios sin tener que entrar a actualizar uno a uno de ellos.

Capítulo 7

Conclusiones y trabajos futuros

De momento se ha desarrollo el proyecto de una manera básica que a futuro se busca mejorar con la integración de más elementos y mejorar en una segunda versión para mejorar algunos otros temas que aporten más funcionalidades dentro de la aplicación entre ellas se tienen contemplado como trabajos futuros para este proyecto lo siguiente:

- La integración de APIs que permitan el cobro por medio de tarjeta
- Agregar una cartera de imágenes en la Vista de Detalle de Producto para no dejar a la idea nada del producto y sea lo más claro posible.
- Generación de tickets con envió a correo electrónico.
- Agregar estilos CSS para cumplir con la disponibilidad en la mayor cantidad de dispositivos.
- Agregar el módulo de buscador para determinar existencia de productos y direccionamiento rápido a la vista detalle de producto.

En cuanto a las conclusiones de esta tesis puedo concluir que la integración de la Programación Reactiva – Funcional aportan en la creación de sistemas mantenibles y escalables además de mejorar el tiempo de respuesta para nuestros usuarios, asegurando mantenerlos navegando dentro de nuestros sistemas. En cuanto a el mantenimiento permite tener un código más claro en cuanto a lo que se quiere lograr, centrándonos más en ello que en cómo hacerlo. Es claro que todo paradigma de programación es importante, pero va a depender su implementación de lo que se quiere lograr y del tipo de sistema que se va a realizar, pues esto será el punto por considerar al tomar una decisión en cuanto a que paradigma usar o bien paradigmas pues no es necesario centrarse en uno cuando es posible obtener mejores beneficios con la integración adecuada de diferente Paradigmas.

Capítulo 8

Referencias

[1] Acerca de GitHub Pages - Documentación de GitHub. (s. f.). GitHub Docs. https://docs.github.com/es/pages/getting-started-with-github-pages/about-github-pages

[2] Aprende qué es la Programación Funcional. (2022, 9 abril). Aprender BIG DATA. Recuperado 15 de agosto de 2022, de https://aprenderbigdata.com/programacion-funcional/

[3] Bonér, J., Farley, D., Kuhn, R., & Thompson, M. (2014, 16 septiembre). El Manifiesto de Sistemas Reactivos. El Manifiesto de Sistemas Reactivos. Recuperado 8 de agosto de 2022, de https://www.reactivemanifesto.org/es

[4] Firebase. (s. f.). Recuperado 29 de marzo de 2023, de https://firebase.google.com/?hl=es-419

[5] G. (s. f.-b). Información general de TypeScript - Training. Microsoft Learn. Recuperado 29 de marzo de 2023, de https://learn.microsoft.com/es-es/training/modules/typescript-get-started/2-typescript-overview

[6] Germain R., C. E. (2020). Desarrollo de aplicaciones web utilizando JavaScript [TFG]. Universidad Politécnica de Sinaloa.

[7] Mir G., P. (2015, diciembre). Aplicaciones Web Origen y Evolución [Diapositivas]. DOCPLAYER. https://docplayer.es/34083677-Aplicaciones-web-origen-y-evolucion.html

[8] Paradigma. (s. f.). En Real Academia Española. Recuperado 15 de junio de 2022, de https://dle.rae.es/paradigma?m=form

[9] Pulido, M. (2021, 2 noviembre). Todo lo que deberías saber sobre programación reactiva. SlashMobility | Soluciones mobile. Recuperado 16 de marzo de 2022, de https://slashmobility.com/blog/2019/09/programacion-reactiva/#:%7E:text=La%20programaci%C3%B3n%20reactiva%2C%20o%20Reactive,ejecutando%20una%20serie%20de%20eventos.

[10] Restrepo, F. (s. f.). Programación Reactiva. Paradigmas de Programación. Recuperado 14 de junio de 2022, de http://ferestrepoca.github.io/paradigmas-de-programacion/reactive/reactive_teoria/index.html

[11] Vaca, C. (2011, 11 febrero). Paradigmas de Programación [Diapositivas]. https://www.infor.uva.es/~cvaca/asigs/docpar/intro.pdf.

[12] Vega, R. M. (2022, 17 enero). Tipos de desarrollo de aplicaciones web: ejemplos y características. Profile Software Services. Recuperado 29 de Julio de 2022, de https://profile.es/blog/desarrollo-aplicaciones-web/

[13] Zayas, C. A. (s. f.). Paradigmas de la Programación con Python. Gitbooks. Recuperado 16 de junio de 2022, de https://czayas.gitbooks.io/paradigmas-de-la-programacion-con-python/content/