INSTITUTO TÉCNICO RICALDONE

CÓMODOS: MANUAL TÉCNICO

DESARROLLADORES:

Gerson Alexander Echeverría Hernández (ID:20220175)

Teresa Yessenia Ruiz González (ID: 20220651)

Verónica Alejandra Sánchez Rosales (ID: 20220505)

Allan Gabriel Fuentes Galdámez (ID: 20220406)

Alejandro Alberto Fernández Robles (ID: 20220096)

DOCENTE:

Antonio Samuel Benavides

FECHA DE

PUBLICACIÓN:

02/10/2024

ÍNDICE

Contenido

INTRODUCCIÓN

Bienvenido al manual técnico de Cómodo$, una solución móvil innovadora diseñada para revolucionar la experiencia de compra y gestión de inventario en el emprendimiento Cómodo$.

Cómodo es una empresa de venta de calzado que actualmente enfrenta limitaciones significativas en su modelo de negocio debido a la falta de una plataforma digital integrada. Hasta ahora, la gestión de inventario y las interacciones con los clientes se han realizado exclusivamente a través de WhatsApp, lo que ha resultado en ineficiencias operativas y restricciones en el crecimiento del negocio.

El proyecto Cómodos tiene como objetivo principal superar estas limitaciones mediante el desarrollo de una solución tecnológica integral. Esta solución busca optimizar los procesos de venta, mejorar la experiencia del cliente y facilitar la gestión interna del negocio.

La solución Cómodos se compone de tres elementos fundamentales:

1. **Aplicación móvil para clientes**:
    - Permite a los usuarios explorar el catálogo de productos
    - Facilita la realización de reservas y compras
    - Ofrece herramientas para la gestión de cuentas de usuario
2. **Sitio web administrativo**:
    - Proporciona a los administradores herramientas para la gestión eficiente de ventas
    - Permite la administración dinámica del inventario (adición, eliminación y actualización de productos)
3. **Sistema de gestión de bases de datos**:

- Almacena y gestiona de manera eficiente toda la información necesaria para el funcionamiento de la aplicación

Este manual técnico está diseñado para proporcionar una comprensión detallada de la arquitectura, funcionamiento y mantenimiento de la solución Cómodos. Está dirigido a desarrolladores, administradores de sistemas y personal técnico involucrado en la implementación, mantenimiento y futuras actualizaciones del sistema.

En las siguientes secciones, encontrará información detallada sobre la arquitectura del sistema, especificaciones técnicas, guías de instalación y configuración, así como procedimientos de mantenimiento y resolución de problemas.

REQUISITOS DE SISTEMA

Requisitos básicos

1. **Sistema operativo:**
    - Windows 10 o Windows 11 (para compatibilidad óptima con Visual Studio).
    - Linux o macOS si planeas usar herramientas de desarrollo específicas o contenedores como Docker.
2. **Procesador:**
    - Intel Core i5 o AMD Ryzen 5 (o superior) para un rendimiento fluido.
3. **Memoria RAM:**
    - 8 GB como mínimo, pero 16 GB o más es recomendable para un desarrollo más cómodo y para manejar varias aplicaciones a la vez.
4. **Espacio en disco:**
    - 256 GB de almacenamiento en SSD (unidad de estado sólido) como mínimo.
5. **Conexión a internet:**
    - Para poder observar el sitio web, las actualizaciones, bibliotecas y para trabajar con repositorios en la nube.

INSTALACIÓN Y CONFIGURACIÓN

Instalación de Visual Studio

1. **Descargar el instalador de Visual Studio:**
    - Ve al sitio web oficial de Visual Studio.
    - Haz clic en “Download Visual Studio” y selecciona la versión Community es gratuita para uso personal y académico.

2. **Ejecutar el instalador:**
    - Una vez descargado el archivo, ejecútalo
    - Si se te pide permiso para realizar cambios en tu dispositivo, haz clic en “Sí”.

3. **Seleccionar componentes:**
    - Aparecerá la ventana de instalador de Visual Studio.
    - Selecciona los componentes que necesitas. Por ejemplo:
        - **Desarrollo de aplicaciones de escritorio con C#**
        - **Desarrollo web y de la nube**
    - Puedes ver una descripción de cada componente y desmarcar los que no necesites.

4. **Elegir el directorio de instalación:**
    - Puedes cambiar el directorio de instalación si lo deseas, pero la configuración por defecto suele ser adecuada.

5. **Instalar:**
    - Haz clic en el botón “Instalar”.
    - Espera a que la instalación se complete. Esto puede tardar varios minutos.

6. **Reiniciar el sistema (si es necesario):**
    - Una vez completada la instalación, es posible que necesites reiniciar tu computadora.

INSTALACIÓN DE XAMPP

1. **Descargar XAMPP:**
    - Ve al sitio web oficial de XAMPP.
    - Descarga el instalador correspondiente para tu sistema operativo (Windows en este caso).

2. **Ejecutar el instalador:**
    - Abre el archivo descargado
    - Si se te pide permiso para realizar cambios en tu dispositivo, haz clic en “Sí”.

3. **Seleccionar componentes:**
    - En la ventana de instalación, puedes seleccionar los componentes que deseas instalar:
        - **Apache** (servidor web)
        - **MySQL** (servidor de base de datos)
        - **PHP** (lenguaje de programación)
        - **phpMyAdmin** (gestor de bases de datos)

4. **Elegir el directorio de instalación:**
    - Selecciona el directorio donde quieres instalar XAMPP

5. **Instalar:**
    - Haz clic en “Next” y luego en “Install”.
    - Espera a que se complete la instalación.

6. Iniciar XAMPP:

- Una vez instalado, abre el Panel de Control de XAMPP desde el menú de inicio o desde el directorio de instalación.
- Inicia los servicios necesarios (Apache y MySQL) haciendo clic en “Start” junto a cada servicio.

INSTALACIÓN DE MYSQL WORKBENCH

1. Descargar MySQL Workbench:
2. Al cual tendríamos que ir al sitio oficial de workbench.
3. Selecciona la versión de MySQL Workbench que corresponda con tu sistema operativo (Windows en este caso).
4. Haz clic en el botón “Download” y luego en “No, gracias, Iniciar mi descarga” para obtener el archivo de instalación directamente.

5. Ejecutar el Instalador:
    - Abre el archivo descargado
    - Si se te pide permiso para realizar cambios en tu dispositivo, haz clic en “Sí”.

6. Instalación:
    - Se abrirá el asistente de instalación de MySQL Workbench.
    - Haz clic en “Next” para continuar.

7. Aceptar el Acuerdo de Licencia:
    - Lee el acuerdo de licencia, acepta los términos, y haz clic en “Next”.

8. Elegir el directorio de instalación:
    - Elige el directorio en el que quieres instalar MySQL Workbench o deja la ubicación predeterminada.
    - Haz clic en “Next”.

9. Seleccionar componentes:
    - Asegúrate de que los componentes necesarios estén seleccionados (la configuración predeterminada suele estar bien).
    - Haz clic en “Next”.

10. Instalar:
    - Haz clic en “Install” para comenzar la instalación.
    - Espera a que la instalación se complete y luego haz clic en “Finish”.

CONFIGURACIÓN INICIAL

1. Configuración Inicial de Visual Studio

1.1 Instalación y primer Inicio:

- Una vez instalado, abre Visual Studio.
- Si es la primera vez que lo usas, se te pedirá que elijas una carga de trabajo. Puedes elegir según tus necesidades:
    - **Desarrollo en .NET**: Para aplicaciones web, de escritorio y móviles con C#.
    - **Desarrollo de aplicaciones de escritorio con C++**: Para aplicaciones de escritorio usando C++.
    - **Desarrollo web y en la nube**: Para aplicaciones web y servicios en la nube.

1.2 Configuración del entorno:

- Puedes personalizar el entorno de desarrollo según tus preferencias. Ve a **Herramientas > Opciones** para ajustar la configuración.
- En **Entorno > General**, puedes elegir el tema de color (claro o oscuro) y otras configuraciones generales.
- Configura el control de versiones si planeas usar Git u otro sistema de control de versiones. Esto se puede hacer en **Herramientas > Opciones > Control de fuente**.

1.3 Crear un nuevo proyecto:

- Selecciona **Archivo > Nuevo > Proyecto**.
- Elige el tipo de proyecto que deseas crear según la carga de trabajo que seleccionaste (por ejemplo, una aplicación de consola, aplicación web, etc.).
- Completa los detalles del proyecto y haz clic en **Crear**.

2. CONFIGURACIÓN INICIAL DE XAMPP

2.1 Iniciar Servicios:

- Abre el **panel de control de XAMPP**.
- Inicia los servicios que necesitas:
  - **Apache**: Para servir tus aplicaciones web.
  - **MySQL**: Para gestionar bases de datos.

2.2 Configuración de Apache:

- Si Apache no inicia debido a conflictos de puerto, cambia el puerto:
  - Abre el archivo de configuración httpd.conf ubicado en C:\xampp\apache\conf\.
  - Busca Listen 80 y cámbialo a otro puerto, como Listen 8080.
  - Guarda el archivo y reinicia Apache desde el Panel de Control de XAMPP.

2.3 Configuración de MySQL:

- Si necesitas cambiar la configuración del puerto de MySQL, edita el archivo my.ini ubicado en C:\xampp\mysql\bin\.
- Busca la línea port=3306 y cámbiala si es necesario.
- Guarda los cambios y reinicia MySQL desde el Panel de Control de XAMPP.

2.4 Configuración de phpMyAdmin:

- Accede a phpMyAdmin a través de http://localhost/phpmyadmin.
- Puedes crear nuevas bases de datos y gestionar usuarios desde la interfaz web.

3. CONFIGURACIÓN INICIAL DE MYSQL WORKBENCH

3.1 Crear una nueva conexión:

- Abre MySQL Workbench.
- En la pantalla principal, haz clic en el icono + al lado de “MySQL Connections” para crear una nueva conexión.

3.2 Configuración de conexión:

- **Connection Name:** Dale un nombre descriptivo a la conexión (por ejemplo, Localhost).
- **Connection Method:** Selecciona “Standard (TCP/IP)”.
- **Hostname:** Introduce localhost o 127.0.0.1.
- **Port:** Deja el puerto predeterminado 3306.
- **Username:** Introduce root (o el nombre de usuario que uses).
- **Password:** Introduce la contraseña de MySQL o deja en blanco para ingresarla cada vez.

3.3 Probar la conexión:

- Haz clic en **Test Connection** para asegurarte de que la configuración es correcta.
- Si la prueba es exitosa, haz clic en **OK** para guardar la configuración de la conexión.

3.4 Uso inicial:

- Haz doble clic en la conexión creada para abrir una ventana de consulta.
- Puedes usar el editor SQL para ejecutar comandos y gestionar bases de datos.

RESOLUCIÓN DE PROBLEMAS COMUNES

Problemas comunes con Visual Studio

1. **Problema: Instalación no se completa o se congela.**
   - **Solución:** Asegúrate de tener suficiente espacio en disco y una conexión a Internet estable. Intenta ejecutar el instalador como administrador (clic derecho en el archivo del instalador y selecciona “Ejecutar como administrador”). También puedes intentar desactivar temporalmente el antivirus o el firewall, ya que a veces estos pueden interferir con la instalación.

2. **Problema: Error al descargar componentes.**
   - **Solución:** Verifica tu conexión a Internet y asegúrate de que no haya restricciones de red que bloqueen las descargas. También puedes intentar ejecutar el instalador en modo de compatibilidad (clic derecho en el archivo del instalador, Propiedades, Compatibilidad, Ejecutar este programa en modo de compatibilidad para).

Problemas Comunes con XAMPP

3. **Problema: XAMPP no se inicia o Apache/MySQL no se inicia.**
   - **Solución:** Asegúrate de que los puertos utilizados por Apache (80) y MySQL (3306) no estén ocupados por otros programas. Puedes cambiar el puerto en la configuración de XAMPP si es necesario (para Apache, edita httpd.conf y cambia la línea Listen 80 a otro puerto como Listen 8080). Para MySQL, edita my.ini y cambia el puerto si hay un conflicto.

4. **Problema: Error al iniciar Apache debido a conflictos de puerto.**
    - **Solución:** Si el puerto 80 está en uso, cambia el puerto de Apache en el archivo de configuración. Abre C:\xampp\apache\conf\extra\httpd-vhosts.conf y cambia el puerto de Listen 80 a Listen 8080. Luego, actualiza la configuración en el archivo httpd.conf de Apache (busca Listen 80 y cámbialo a Listen 8080).

Problemas comunes con MySQL Workbench

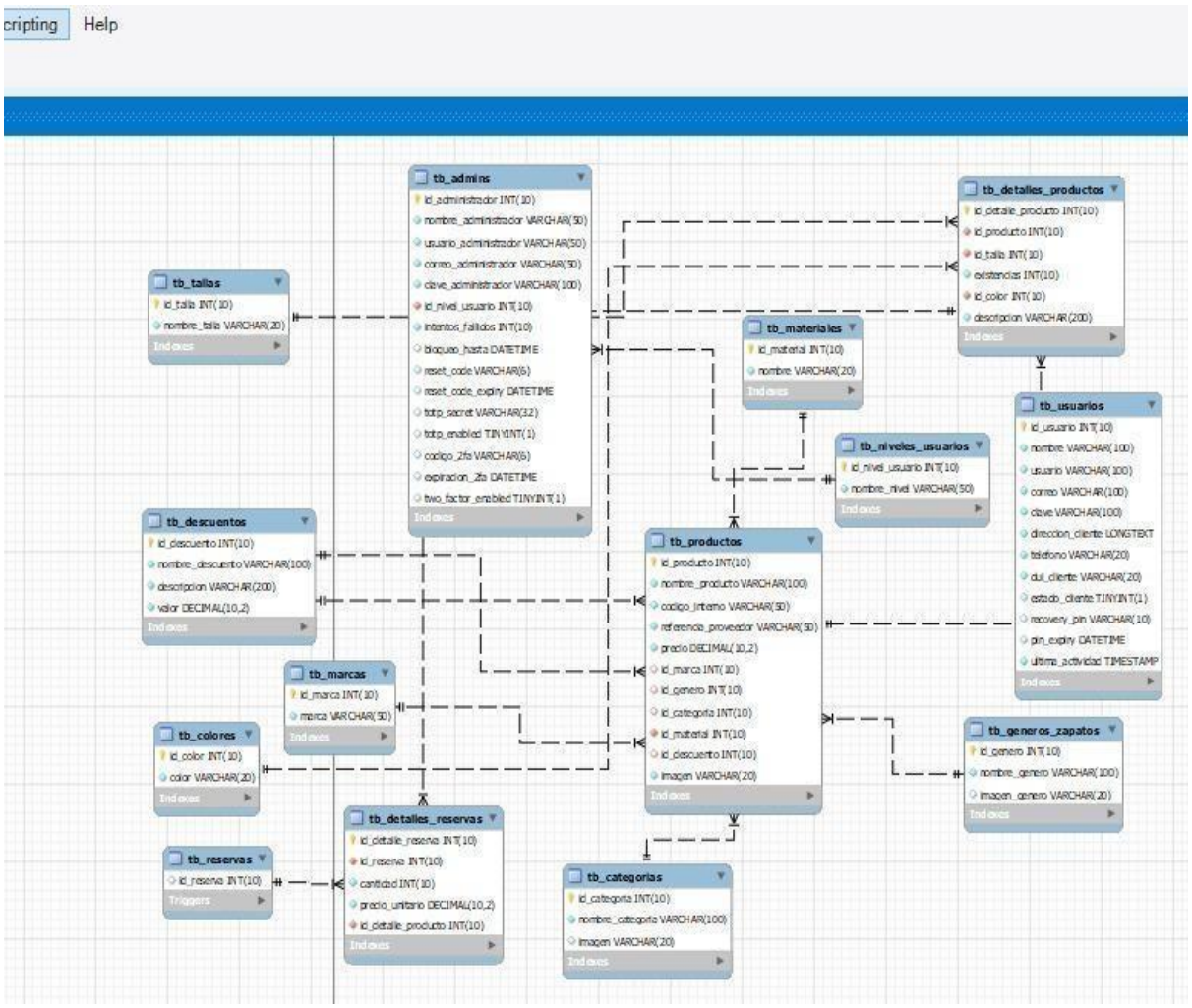
1. Problema: No se puede conectar a MySQL desde MySQL Workbench:
    - **Solución:** Verifica la configuración de la conexión (host, puerto, usuario y contraseña). Asegúrate de que el servidor MySQL esté en ejecución. Puedes probar la conexión usando la línea de comandos para verificar si el problema es específico de MySQL Workbench o de la configuración general de MySQL.

2. Problema: Error de autenticación de usuario.
    - Solución: Asegúrate de que el usuario y la contraseña sean correctos. Si has olvidado la contraseña del usuario root, puedes restablecerla siguiendo la documentación de MySQL sobre cómo cambiar la contraseña de root.

ARQUITECTURA DE LA APLICACIÓN

A continuación, se mostrarán los diferentes tipos de diagrama de flujo de datos para las acciones las cuales se realizan en el programa también el diagrama de la base de datos.

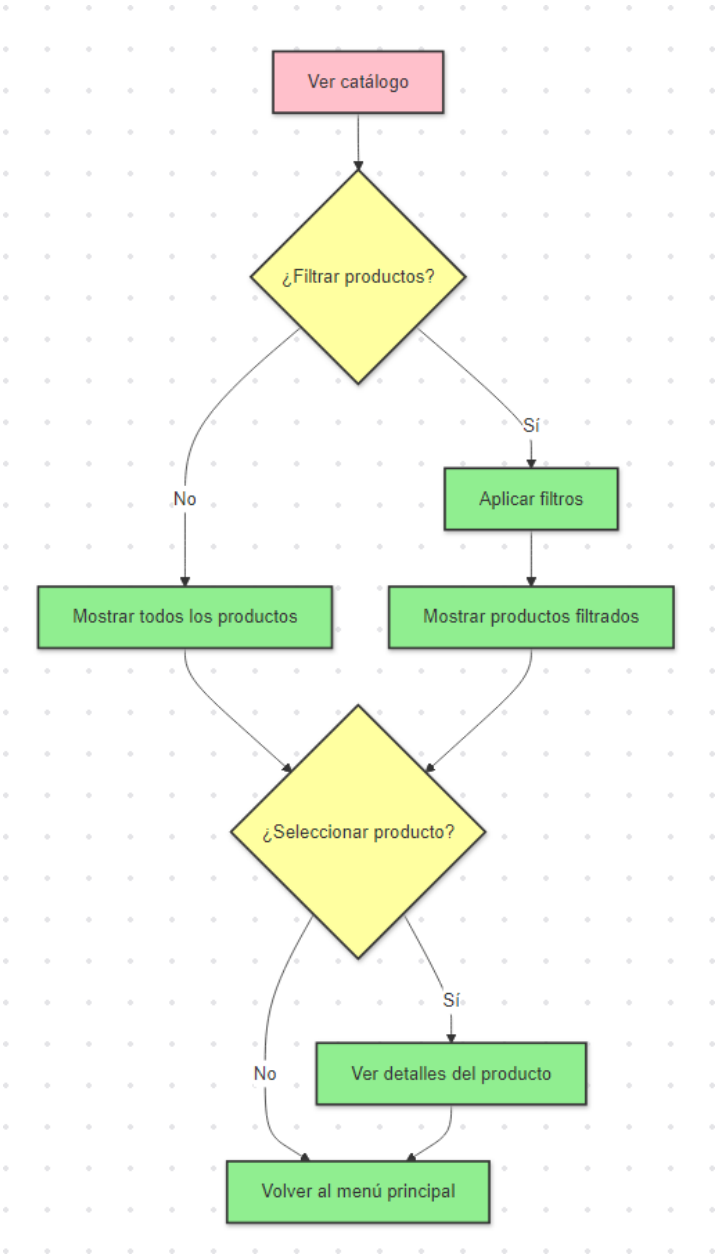
Modelo relacional de base de los datos



Modelo relacional de la aplicación 1.1 Base de Datos

Proceso de ver catalogo

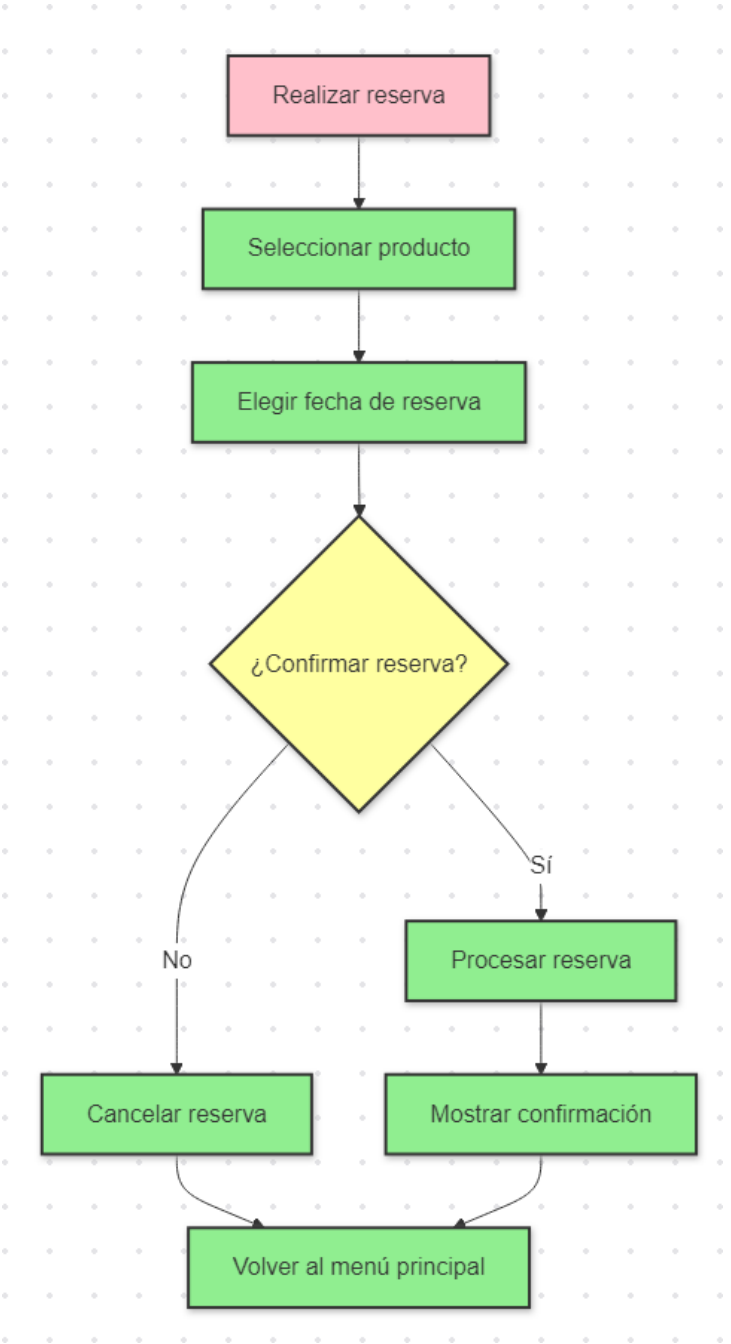
En este proceso se puede observar las funciones que el usuario haría al momento de ver e catalogo



Diagramas de la aplicación 2.2 Catálogo

Realizar una reserva

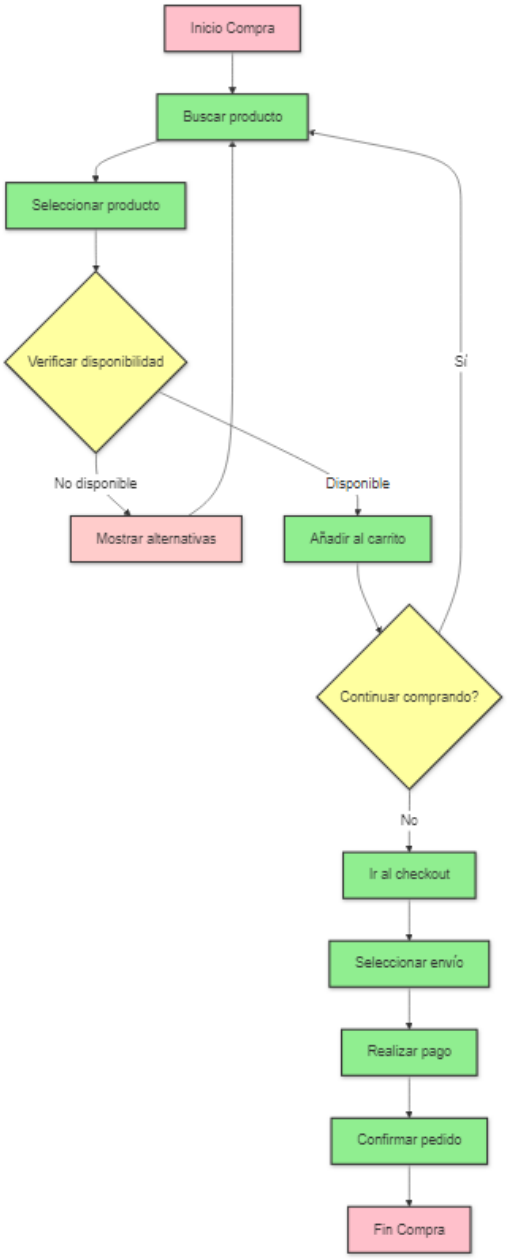
Ahora se verían los procesos los cuales el usuario haría cuando realice una reserva de producto



Diagramas de la aplicación 3.3 Reserva

Compra de productos

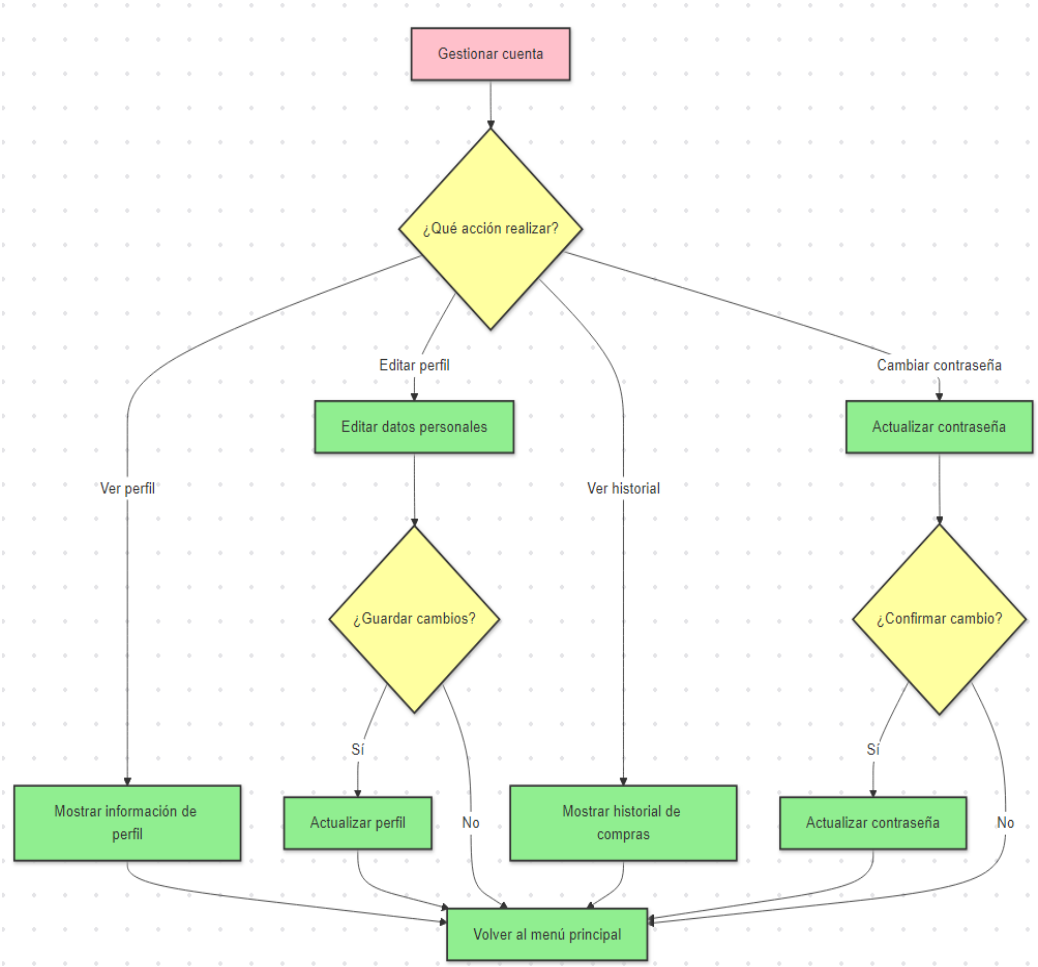
Ahora el proceso el cual haria al momento de la compra del producto que desee



Diagramas de la aplicación 4.4 Compra de productos

Gestionar una cuenta

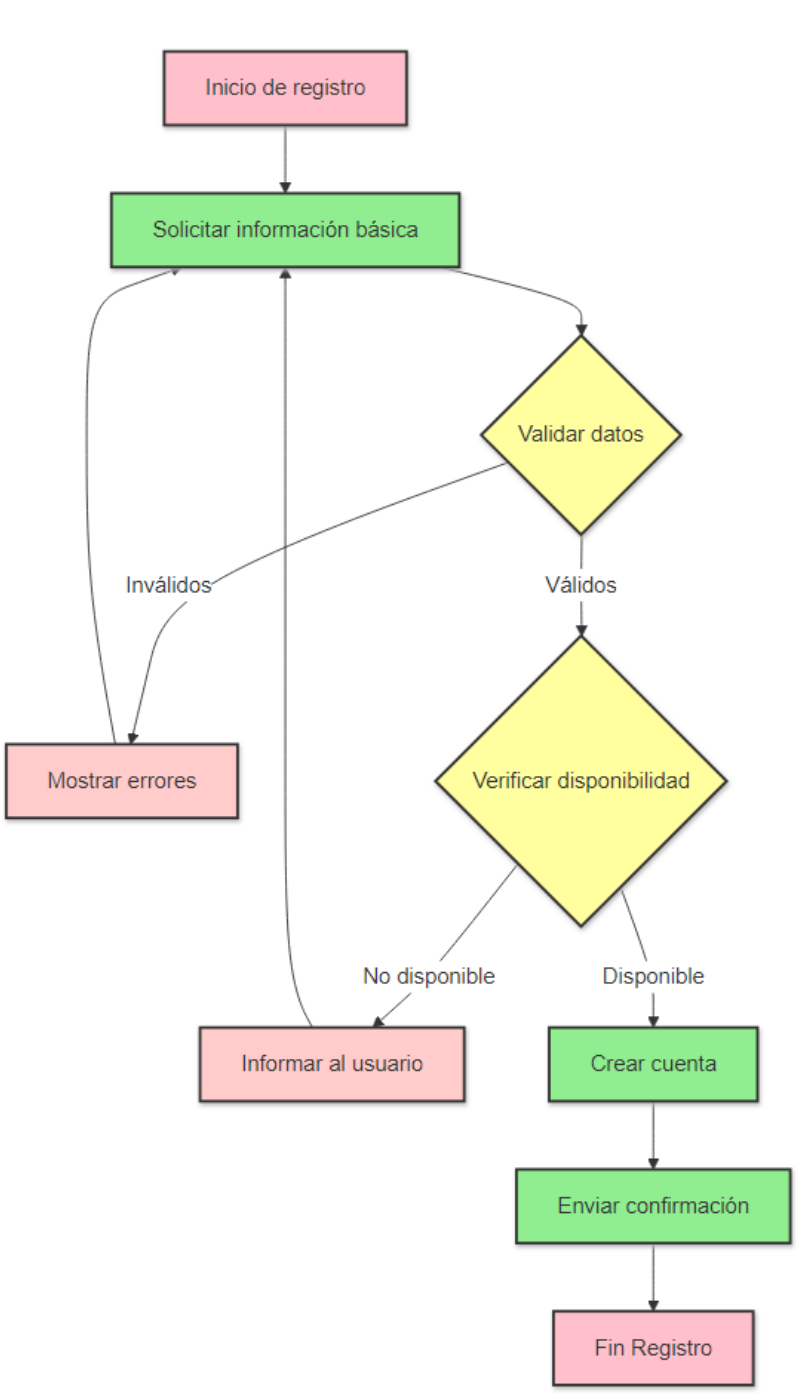
Ahora los procesos para gestionar la cuenta del usuario



Diagramas de la aplicación 5.5 Gestionar cuenta de usuario

Registro de usuarios

Ahora el proceso el cual el usuario tiene que realizar al momento de registrarse



Diagramas de la aplicación 6.6 Registro del usuario

ESTRUCTURA DE LA BASE DE DATOS:

Script de la base:

```
-- Eliminar la base de datos si ya existe
DROP DATABASE IF EXISTS expo_comodos;

-- Crear la base de datos
CREATE DATABASE expo_comodos;

-- Seleccionar la base de datos para su uso
USE expo_comodos;

-- Tabla de usuarios
-- Almacena información de los clientes registrados
CREATE TABLE tb_usuarios (
  id_usuario INT UNSIGNED AUTO_INCREMENT NOT NULL,
  nombre VARCHAR(100) NOT NULL,
  usuario VARCHAR(100) NOT NULL,
  correo VARCHAR(100) NOT NULL,
  clave VARCHAR(100) NOT NULL,
  direccion_cliente LONGTEXT NOT NULL,
  telefono VARCHAR(20) NOT NULL,
  dui_cliente VARCHAR(20) NOT NULL, -- Documento Único de Identidad
  estado_cliente TINYINT(1) DEFAULT TRUE,
  PRIMARY KEY (id_usuario),
  CONSTRAINT uc_usuario UNIQUE (usuario),
  CONSTRAINT uc_correo UNIQUE (correo),
  CONSTRAINT uc_telefono UNIQUE (telefono),
  CONSTRAINT uc_dui_cliente UNIQUE (dui_cliente)
);

-- Tabla de niveles de usuarios
-- Define los diferentes niveles de acceso para los administradores
CREATE TABLE tb_niveles_usuarios (
  id_nivel_usuario INT UNSIGNED AUTO_INCREMENT NOT NULL,
  nombre_nivel VARCHAR (50) NOT NULL,
  PRIMARY KEY (id_nivel_usuario)
);

-- Tabla de administradores
-- Almacena información de los usuarios administrativos del sistema
CREATE TABLE tb_admins (
  id_administrador INT UNSIGNED AUTO_INCREMENT NOT NULL,
  nombre_administrador VARCHAR(50) NOT NULL,
  usuario_administrador VARCHAR(50) NOT NULL,
  correo_administrador VARCHAR(50) NOT NULL,
  clave_administrador VARCHAR(100) NOT NULL,
  id_nivel_usuario INT UNSIGNED NOT NULL,
```

```
  PRIMARY KEY (id_administrador),
  CONSTRAINT fk_nivel_usuario FOREIGN KEY (id_nivel_usuario) REFERENCES
tb_niveles_usuarios(id_nivel_usuario),
  CONSTRAINT uc_usuario_administrador UNIQUE (usuario_administrador),
  CONSTRAINT uc_correo_administrador UNIQUE (correo_administrador)
);

-- Tabla de géneros de zapatos
-- Clasifica los productos por género
CREATE TABLE tb_generos_zapatos (
  id_genero INT UNSIGNED AUTO_INCREMENT NOT NULL,
  nombre_genero VARCHAR(100) NOT NULL,
  imagen_genero VARCHAR(20) NULL,
  PRIMARY KEY (id_genero)
);

-- Tabla de categorías
-- Define las diferentes categorías de productos
CREATE TABLE tb_categorias (
  id_categoria INT UNSIGNED AUTO_INCREMENT NOT NULL,
  nombre_categoria VARCHAR(100) NOT NULL,
  imagen VARCHAR(20) NULL,
  PRIMARY KEY (id_categoria)
);

-- Tabla de tallas
-- Almacena las diferentes tallas disponibles para los productos
CREATE TABLE tb_tallas (
  id_talla INT UNSIGNED AUTO_INCREMENT NOT NULL,
  nombre_talla VARCHAR(20) NOT NULL,
  PRIMARY KEY (id_talla)
);

-- Tabla de marcas
-- Almacena las diferentes marcas de productos
CREATE TABLE tb_marcas (
  id_marca INT UNSIGNED AUTO_INCREMENT NOT NULL,
  marca VARCHAR(50) NOT NULL,
  PRIMARY KEY (id_marca)
);

-- Tabla de colores
-- Define los colores disponibles para los productos
CREATE TABLE tb_colores (
  id_color INT UNSIGNED AUTO_INCREMENT NOT NULL,
  color VARCHAR(20) NOT NULL,
  PRIMARY KEY (id_color)
);
```

```
-- Tabla de descuentos
-- Almacena información sobre los descuentos aplicables a los productos
CREATE TABLE tb_descuentos (
  id_descuento INT UNSIGNED AUTO_INCREMENT NOT NULL,
  nombre_descuento VARCHAR(100) NOT NULL,
  descripcion VARCHAR(200) NOT NULL,
  valor DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (id_descuento),
  UNIQUE (nombre_descuento),
  CONSTRAINT ck_valor CHECK (valor >= 0)
);

-- Tabla de materiales
-- Define los materiales utilizados en los productos
CREATE TABLE tb_materiales (
  id_material INT UNSIGNED AUTO_INCREMENT NOT NULL,
  nombre VARCHAR(20) NOT NULL,
  PRIMARY KEY (id_material)
);

-- Tabla de productos
-- Almacena la información principal de los productos
CREATE TABLE tb_productos (
  id_producto INT UNSIGNED AUTO_INCREMENT NOT NULL,
  nombre_producto VARCHAR(100) NOT NULL,
  codigo_interno VARCHAR(50) NOT NULL,
  referencia_proveedor VARCHAR(50) NOT NULL,
  precio DECIMAL(10,2) NOT NULL,
  id_marca INT UNSIGNED,
  id_genero INT UNSIGNED,
  id_categoria INT UNSIGNED,
  id_material INT UNSIGNED NOT NULL,
  id_descuento INT UNSIGNED NOT NULL,
  imagen VARCHAR(20) NOT NULL,
  PRIMARY KEY (id_producto),
  CONSTRAINT fk_material FOREIGN KEY (id_material) REFERENCES
tb_materiales(id_material),
  CONSTRAINT fk_marcas FOREIGN KEY (id_marca) REFERENCES tb_marcas(id_marca),
  CONSTRAINT fk_generos FOREIGN KEY (id_genero) REFERENCES
tb_generos_zapatos(id_genero),
  CONSTRAINT fk_descuento FOREIGN KEY (id_descuento) REFERENCES
tb_descuentos(id_descuento),
  CONSTRAINT fk_categorias FOREIGN KEY (id_categoria) REFERENCES
tb_categorias(id_categoria),
  CONSTRAINT ck_precio CHECK (precio >= 0),
  CONSTRAINT uc_nombre_producto UNIQUE (nombre_producto),
  CONSTRAINT uc_codigo_interno UNIQUE (codigo_interno),
  CONSTRAINT uc_referencia_proveedor UNIQUE (referencia_proveedor)
);
```

```
-- Tabla de detalles de productos
-- Almacena información específica de cada variante de producto (por talla, color, etc.)
CREATE TABLE tb_detalles_productos (
  id_detalle_producto INT UNSIGNED AUTO_INCREMENT NOT NULL,
  id_producto INT UNSIGNED NOT NULL,
  id_talla INT UNSIGNED NOT NULL,
  existencias INT UNSIGNED NOT NULL,
  id_color INT UNSIGNED NOT NULL,
  descripcion VARCHAR(200) NOT NULL,
  PRIMARY KEY (id_detalle_producto),
  CONSTRAINT fk_producto FOREIGN KEY (id_producto) REFERENCES
tb_productos(id_producto),
  CONSTRAINT fk_talla FOREIGN KEY (id_talla) REFERENCES tb_tallas(id_talla),
  CONSTRAINT fk_color FOREIGN KEY (id_color) REFERENCES tb_colores(id_color),
  CONSTRAINT ck_existencias  CHECK (existencias >= 0)
);

-- Tabla de reservas
-- Almacena información sobre las reservas realizadas por los usuarios
CREATE TABLE tb_reservas (
  id_reserva INT UNSIGNED AUTO_INCREMENT NOT NULL,
  id_usuario INT UNSIGNED NOT NULL,
  fecha_reserva DATETIME DEFAULT CURRENT_DATE() NOT NULL,
  estado_reserva ENUM ('Aceptado', 'Pendiente', 'Cancelado') NOT NULL,
  PRIMARY KEY (id_reserva),
  CONSTRAINT fk_reserva_usuario FOREIGN KEY (id_usuario) REFERENCES tb_usuarios
(id_usuario)
);

-- Tabla de detalles de reservas
-- Almacena los productos específicos incluidos en cada reserva
CREATE TABLE tb_detalles_reservas (
  id_detalle_reserva INT UNSIGNED AUTO_INCREMENT NOT NULL,
  id_reserva INT UNSIGNED NOT NULL,
  cantidad INT UNSIGNED NOT NULL,
  precio_unitario DECIMAL(10,2) NOT NULL,
  id_detalle_producto INT UNSIGNED NOT NULL,
  PRIMARY KEY (id_detalle_reserva),
  CONSTRAINT fk_reserva FOREIGN KEY (id_reserva) REFERENCES
tb_reservas(id_reserva),
  CONSTRAINT fk_detalle_producto FOREIGN KEY (id_detalle_producto) REFERENCES
tb_detalles_productos(id_detalle_producto),
  CONSTRAINT ck_cantidad  CHECK (cantidad >= 0),
  CONSTRAINT ck_precio_unitario CHECK (precio_unitario >= 0)
);

-- Agregar columnas para la recuperación de contraseña en la tabla de usuarios
ALTER TABLE tb_usuarios
ADD COLUMN recovery_pin VARCHAR(10) NULL,
ADD COLUMN pin_expiry DATETIME NULL;
```

```
-- Agregar columnas para la seguridad de la cuenta en la tabla de administradores
ALTER TABLE tb_admins
ADD COLUMN intentos_fallidos INT UNSIGNED DEFAULT 0 NOT NULL,
ADD COLUMN fecha_clave DATETIME NULL DEFAULT NOW(),
ADD COLUMN bloqueo_hasta DATETIME NULL;

-- Agregar columnas para el restablecimiento de contraseña en la tabla de administradores
ALTER TABLE tb_admins
ADD COLUMN reset_code VARCHAR(6) DEFAULT NULL,
ADD COLUMN reset_code_expiry DATETIME DEFAULT NULL;

-- Agregar columnas para la autenticación de dos factores en la tabla de administradores
ALTER TABLE tb_admins ADD COLUMN totp_secret VARCHAR(32);
ALTER TABLE tb_admins ADD COLUMN totp_enabled BOOLEAN DEFAULT FALSE;

-- Agregar columna para el seguimiento de la última actividad del usuario
ALTER TABLE tb_usuarios ADD COLUMN ultima_actividad TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP;

-- Agregar columnas para la autenticación de dos factores en la tabla de administradores
ALTER TABLE tb_admins
ADD COLUMN codigo_2fa VARCHAR(6),
ADD COLUMN expiracion_2fa DATETIME;

-- Insertar niveles de usuario predefinidos
INSERT INTO tb_niveles_usuarios (id_nivel_usuario, nombre_nivel)
VALUES
(1, 'administrador'),
(2, 'inventaristas'),
(3, 'vendedoras');

-- Crear una función para generar un saludo personalizado
DELIMITER //

CREATE FUNCTION generar_saludo(nombre_usuario VARCHAR(100))
RETURNS VARCHAR(255)
BEGIN
   DECLARE saludo VARCHAR(255);
   SET saludo = CONCAT('¡Hola ', nombre_usuario, '! Bienvenido/a.');
   RETURN saludo;
END;
//

DELIMITER ;

-- Modificar la tabla de productos para permitir descuentos nulos
ALTER TABLE tb_productos
MODIFY COLUMN id_descuento INT UNSIGNED DEFAULT NULL,
ADD CONSTRAINT ck_descuento
   FOREIGN KEY (id_descuento)
   REFERENCES tb_descuentos(id_descuento)
   ON DELETE CASCADE
   ON UPDATE CASCADE;
```

```
-- Crear un trigger para actualizar existencias después de aceptar una reserva
DELIMITER //

CREATE TRIGGER actualizar_existencias
AFTER UPDATE ON tb_reservas
FOR EACH ROW
BEGIN
   IF NEW.estado_reserva = 'Aceptado' THEN
      UPDATE tb_detalles_productos dp
      INNER JOIN tb_detalles_reservas dr ON dp.id_detalle_producto = dr.id_detalle_producto
      SET dp.existencias = dp.existencias - dr.cantidad
      WHERE dr.id_reserva = NEW.id_reserva;
   END IF;
END //

DELIMITER ;

-- Crear un procedimiento almacenado para eliminar reservas pendientes antiguas
DELIMITER //

CREATE PROCEDURE eliminar_reservas_pendientes()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE usuario_id INT;
   DECLARE reserva_id INT;
   DECLARE usuario_correo VARCHAR(100);

   DECLARE cur CURSOR FOR
      SELECT r.id_usuario, r.id_reserva, u.correo
      FROM tb_reservas r
      JOIN tb_usuarios u ON r.id_usuario = u.id_usuario
      WHERE r.estado_reserva = 'Pendiente'
        AND r.fecha_reserva < NOW() - INTERVAL 72 HOUR;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN cur;

   read_loop: LOOP
      FETCH cur INTO usuario_id, reserva_id, usuario_correo;
      IF done THEN
         LEAVE read_loop;
      END IF;

      -- Eliminar todos los detalles de la reserva
      DELETE FROM tb_detalles_reservas WHERE id_reserva = reserva_id;

      -- Eliminar la reserva
      DELETE FROM tb_reservas WHERE id_reserva = reserva_id;

      -- Llamar a la función para enviar correo (asume que existe una función 'enviar_correo')
      CALL enviar_correo(usuario_correo, 'Su pedido ha sido eliminado por políticas de la
app.');
```

```
    END LOOP;

    CLOSE cur;
END //

DELIMITER ;

-- Crear un evento programado para ejecutar el procedimiento de eliminación de reservas
DELIMITER //

CREATE EVENT eliminar_reservas_event
ON SCHEDULE EVERY 1 HOUR
DO
BEGIN
    CALL eliminar_reservas_pendientes();
END //

DELIMITER ;

-- Activar el programador de eventos
SET GLOBAL event_scheduler = ON;
```

DICCIONARIO DE DATOS:

Tabla 1

tb_usuarios

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_usuario	INT UNSIGNED	10	No	Llave primaria	Auto-increment
nombre	VARCHAR	100	No		
usuario	VARCHAR	100	No	Único	
correo	VARCHAR	100	No	Único	
clave	VARCHAR	100	No		
direccion_cliente	LONGTEXT	-	No		
telefono	VARCHAR	20	No	Único	
dui_cliente	VARCHAR	20	No	Único	
estado_cliente	TINYINT(1)	-	No		TRUE
recovery_pin	VARCHAR	10	Sí		NULL
pin_expiry	DATETIME	-	Sí		NULL
ultima_actividad	TIMESTAMP	-	No		CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

Tabla 2

tb_niveles_usuarios

Columna	**Tipo**	**Longitud**	**Nulo**	**Índice**	**Predeterminado**
id_nivel_usuario	INT UNSIGNED	10	No	Llave primaria	Auto-increment
nombre_nivel	VARCHAR	50	No		
id_nivel_usuario	INT UNSIGNED	10	No	Llave primaria	Auto-increment

Tabla 3

tb_admins

Columna	**Tipo**	**Longitud**	**Nulo**	**Índice**	**Predeterminado**
id_administrador	INT UNSIGNED	10	No	Llave primaria	Auto-increment
nombre_administrador	VARCHAR	50	No		
usuario_administrador	VARCHAR	50	No	Único	
correo_administrador	VARCHAR	50	No	Único	
clave_administrador	VARCHAR	100	No		
id_nivel_usuario	INT UNSIGNED	10	No	Llave foránea	
intentos_fallidos	INT UNSIGNED	-	No		0
fecha_clave	DATETIME	-	No		NOW()

bloqueo_hasta	DATETIME	-	Sí	NULL
reset_code	VARCHAR	6	Sí	NULL
reset_code_expiry	DATETIME	-	Sí	NULL
totp_secret	VARCHAR	32	Sí	NULL
totp_enabled	BOOLEAN	-	No	FALSE

Tabla 4

tb_generos_zapatos

Columna	**Tipo**	**Longitud**	**Nulo**	**Índice**	**Predeterminado**
id_genero	INT UNSIGNED	10	No	Llave primaria	Auto-increment
nombre_genero	VARCHAR	100	No		
imagen_genero	VARCHAR	20	Sí		NULL

Tabla 5

tb_categorias

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_categoria	INT UNSIGNED	10	No	Llave primaria	Auto-increment
nombre_categoria	VARCHAR	100	No		
imagen	VARCHAR	20	Sí		NULL

Tabla 6

tb_tallas

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_talla	INT UNSIGNED	10	No	Llave primaria	Auto-increment
nombre_talla	VARCHAR	20	No		

Tabla 7

tb_marcas

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_marca	INT UNSIGNED	10	No	Llave primaria	Auto-increment
marca	VARCHAR	50	No		

Tabla 8

tb_colores

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_color	INT UNSIGNED	10	No	Llave primaria	Auto-increment
color	VARCHAR	20	No		

Tabla 9

tb_descuentos

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_descuento	INT UNSIGNED	10	No	Llave primaria	Auto-increment
nombre_descuento	VARCHAR	100	No	Único	
descripcion	VARCHAR	200	No		
valor	DECIMAL(10, 2)	-	No		>= 0

Tabla 10

tb_materiales

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_material	INT UNSIGNED	10	No	Llave primaria	Auto-increment

nombre	VARCHAR	20	No

Tabla 11

tb_productos

Columna	**Tipo**	**Longitud**	**Nulo**	**Índice**	**Predeterminado**
id_producto	INT UNSIGNED	10	No	Llave primaria	Auto-increment
nombre_producto	VARCHAR	100	No	Único	
codigo_interno	VARCHAR	50	No	Único	
referencia_proveedor	VARCHAR	50	No	Único	
precio	DECIMAL(10,2)	-	No		>= 0
id_marca	INT UNSIGNED	10	Sí	Llave foránea	
id_genero	INT UNSIGNED	10	Sí	Llave foránea	
id_categoria	INT UNSIGNED	10	Sí	Llave foránea	
id_material	INT UNSIGNED	10	No	Llave foránea	
id_descuento	INT UNSIGNED	10	No	Llave foránea	
imagen	VARCHAR	20	No		

Tabla 12

tb_detalles_productos

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_detalle_producto	INT UNSIGNED	10	No	Llave primaria	Auto-increment
id_producto	INT UNSIGNED	10	No	Llave foránea	
id_talla	INT UNSIGNED	10	No	Llave foránea	
existencias	INT UNSIGNED	-	No		>= 0
id_color	INT UNSIGNED	10	No	Llave foránea	
descripcion	TEXT	-	No		

Tabla 13

tb_reservas

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_reserva	INT UNSIGNED	10	No	Llave primaria	Auto-increment
id_usuario	INT UNSIGNED	10	No	Llave foránea	
fecha_reserva	DATETIME	-	No		current_date()

estado_reserva	ENUM	-	No		'Aceptado', 'Pendiente', 'Cancelado'

Tabla 14

tb_detalles_reservas

Columna	**Tipo**	**Longitud**	**Nulo**	**Índice**	**Predeterminado**
id_detalle_reserva	INT UNSIGNED	10	No	Llave primaria	Auto-increment
id_reserva	INT UNSIGNED	10	No	Llave foránea	
cantidad	INT UNSIGNED	-	No		>= 0
precio_unitario	DECIMAL(10, 2)	-	No		>= 0
id_detalle_producto	INT UNSIGNED	10	No	Llave foránea	

ARQUITECTURA DE SOFTWARE

1. Capa de presentación (Vistas)

- Ubicación: Expo_Comodo/views/admin/
- Componentes: Archivos HTML (ej. administrador.html, categoria.html, colores.html, etc.)
- Propósito: Interfaz de usuario y presentación de datos
- Interacción: Se comunica con la capa de aplicación para mostrar datos y capturar entradas del usuario

2. Capa de lógica de negocio (Controladores)

- Ubicación: Expo_Comodo/controllers/admin/
- Componentes: Archivos controladores JavaScript
- Propósito: Maneja las solicitudes del usuario, procesa datos y gestiona la lógica de la aplicación
- Interacción:
    - Recibe solicitudes de la capa de presentación
    - Se comunica con la capa de dominio para recuperar y manipular datos
    - Envía datos procesados de vuelta a la capa de presentación

3. Capa de dominio (Modelos)

- Ubicación: Expo_Comodo/api/models/
- Componentes: Archivos de modelo PHP (ej. estructuras de datos y lógica de negocio)

- Propósito: Representa la lógica de negocio central y las estructuras de datos
- Interacción:
    - Recibe solicitudes de la capa de aplicación
    - Implementa reglas de negocio y manipulación de datos
    - Se comunica con la capa de acceso a datos para la persistencia de datos

4. Capa de acceso a datos

- Ubicación: Expo_Comodo/api/
- Componentes: Conexión a la base de datos y manejadores de consultas
- Propósito: Gestiona las interacciones con la base de datos y la persistencia de datos
- Interacción:
    - Recibe solicitudes de manipulación de datos de la capa de dominio
    - Ejecuta consultas a la base de datos y devuelve resultados

5. Capa de a yuda/utilidades

- Ubicación: Expo_Comodo/api/helpers/
- Componentes: Funciones de utilidad y clases auxiliares
- Propósito: Proporciona funcionalidades comunes utilizadas en toda la aplicación
- Interacción: Puede ser utilizada por cualquier otra capa cuando sea necesario

6. Capa de servicios

- Ubicación: Expo_Comodo/api/services/
- Componentes: Clases de servicio para operaciones complejas
- Propósito: Encapsula operaciones de negocio complejas que pueden involucrar múltiples modelos
- Interacción:
    - Es llamada por la Capa de aplicación para tareas complejas
    - Puede interactuar con múltiples componentes de la Capa de Dominio

7. Capa de recursos

- Ubicación: Expo_Comodo/resources/
- Componentes: Archivos CSS, manejo de errores, fuentes, imágenes, JS y SQL
- Propósito: Almacena recursos estáticos y activos utilizados por la aplicación
- Interacción: Utilizada por la Capa de Presentación y potencialmente otras capas para estilos, scripts del lado del cliente y esquema de base de datos

Flujo de Interacción

1. El usuario interactúa con la capa de presentación (Vistas)
2. Las vistas envían solicitudes a la capa de lógica de negocio (Controladores)
3. Los Controladores procesan la solicitud e interactúan con la capa de dominio (Modelos) según sea necesario

4. Los modelos pueden usar la capa de acceso a datos para persistir o recuperar datos

5. Los resultados fluyen de vuelta a través de las capas: Acceso a Datos → Dominio → Aplicación / Lógica de negocio → Presentación

6. Las Capas de Ayuda/Utilidades y Servicios proporcionan soporte a otras capas según se requiera

7. La Capa de Recursos proporciona los activos necesarios durante todo el proceso

Esta arquitectura sigue un patrón típico MVC (Modelo-Vista-Controlador) con capas adicionales para la separación de preocupaciones y una mejor mantenibilidad

DESCRIPCIÓN DE COMPONENTES APLICACIÓN MÓVIL

1. Frontend (Aplicación móvil)

La aplicación está desarrollada utilizando React, un framework de JavaScript para construir interfaces de usuario. La estructura del proyecto incluye:

- **Screens**: Contiene los componentes principales de la interfaz de usuario, incluyendo:
  - LoginScreen
  - DashboardScreen
  - PerfilScreen
  - ProductosScreen
  - OfertasScreen
  - CarritoScreen
  - DetalleProductoScreen
  - HistorialScreen
  - entre otros
- **Components**: Alberga componentes reutilizables como:
  - Buttons
  - Cards
  - Inputs
  - Hooks personalizados
- **Navigation**: Maneja la navegación entre las diferentes pantallas de la aplicación.
- **Assets**: Contiene recursos estáticos como imágenes y fuentes.
- **Estilos**: Archivos de estilo para mantener una apariencia consistente en toda la aplicación.

- **Utils**: Funciones de utilidad para tareas comunes.

2. Backend (API)

El backend está implementado como una API, utilizando PHP, dado el uso de composer. La estructura incluye:

- **Controllers**: Manejan las peticiones HTTP y la lógica de negocio.
- **Models**: Representan las entidades de datos y la lógica de acceso a la base de datos.
- **Services**: Contienen la lógica de negocio compleja y operaciones.
- **Helpers**: Funciones de utilidad para tareas comunes.
- **Libraries**: Código reutilizable y integraciones con servicios externos.
- **Reports**: Generación de reportes y análisis de datos.
- **Vendor**: Dependencias de terceros gestionadas por Composer.

3. Base de Datos

- **MySQL**: Se utiliza como sistema de gestión de base de datos relacional.

4. Estructura de archivos del Backend

- **api/**
  - **helpers/**: Funciones auxiliares
  - **images/**: Almacenamiento de imágenes
  - **libraries/**: Bibliotecas y código reutilizable
  - **models/**: Modelos de datos
  - **reports/**: Generación de reportes.

    - **services/**: Servicios de la aplicación
    - **vendor/**: Dependencias gestionadas por Composer
- **controllers/**: Controladores de la API
- **resources/**: Recursos de la aplicación
- **views/**: Vistas de las páginas.
- **.htaccess**: Configuración del servidor web
- **README.md**: Documentación del proyecto
- **composer.json**: Definición de dependencias
- **composer.lock**: Versiones exactas de las dependencias

5. Flujo de datos

1. La aplicación React realiza peticiones HTTP a la API.
2. Los controladores de la API procesan estas peticiones.
3. Los modelos interactúan con la base de datos MySQL según sea necesario.
4. Los servicios implementan la lógica de negocio compleja.
5. La API devuelve las respuestas a la aplicación React.
6. La interfaz de usuario se actualiza con los datos recibidos.

Esta arquitectura proporciona una clara separación entre el frontend y el backend, facilitando el mantenimiento y la escalabilidad del proyecto.

ESTRUCTURA DEL PROYECTO:

Organización general

El proyecto Expo_Comodo está organizado en una estructura de directorios que separa claramente las diferentes partes de la aplicación web. A continuación, se detalla cada componente principal:

Directorio raíz

- api/: Contiene la lógica del backend y la API.
- controllers/: Aloja los scripts de control para las páginas web.
- resources/: Almacena archivos estáticos de uso general.
- views/: Contiene las páginas HTML de la aplicación.
- .htaccess: Archivo de configuración para el servidor web Apache.
- README.md: Documentación principal del proyecto.
- composer.json y composer.lock: Archivos de configuración de Composer para PHP.

Desglose detallado

api/

- helpers/: Funciones auxiliares para la API.
- images/: Almacenamiento de imágenes relacionadas con la API.
- libraries/: Bibliotecas externas utilizadas por la API.
- models/: Modelos de datos para la API.
- reports/: Generación de informes.
- services/: Servicios web de la API.
- vendor/: Dependencias de terceros gestionadas por Composer.

controllers/

Organizado en subcarpetas para facilitar el mantenimiento:

- admin/: Scripts para el panel de administración.
- public/: Scripts para el sitio público.
- utils/: Scripts de utilidad general y plantillas.

Recomendación: Crear un archivo .js por cada página web, incluso si no se necesita inmediatamente.

resources/

Archivos estáticos organizados en:

- css/: Hojas de estilo (propias y de terceros como frameworks).
- error/: Páginas de error personalizadas.
- fonts/: Fuentes tipográficas.
- img/: Imágenes estáticas (como el logo).
- js/: Scripts JavaScript de terceros (frameworks y bibliotecas).

views/

Páginas HTML organizadas por sitio:

- admin/: Páginas del panel de administración.

Las carpetas deben tener una página principal index.html.

DISEÑO DE LA APLICACIÓN

1. Paleta de colores

Colores principales

- **Morado principal**: #6C5DD3
  - Utilizado en la barra lateral de navegación
  - Botones de acción primaria
- **Blanco**: #FFFFFF
  - Fondo principal de la aplicación
  - Texto sobre fondos oscuros
- **Negro**: #000000
  - Texto principal

Colores Secundarios

- **Rojo**: #FF0000
  - Utilizado para indicadores de tiempo (ej: "La sesión expirará en: 5 min 0 sec")
- **Gris claro**: #F5F5F5
  - Fondo de los campos de entrada

2. Tipografía

Fuentes

- **Fuente Principal**: Poppins
  - Utilizada para todos los textos de la aplicación

Tamaños de fuente

- **Títulos Principales**: 24px
  - Ejemplo: "Buenas tardes, bienvenido"
- **Subtítulos**: 18px
  - Ejemplo: "Proyección de ventas"
- **Texto de navegación**: 16px
  - Elementos del menú lateral
- **Texto regular**: 14px
  - Texto de campos de entrada

3. Elementos de interfaz

Botones

- **Botón Primario**
  - Fondo: Morado (#6C5DD3)
  - Texto: Blanco
  - Bordes redondeados: 8px
  - Padding: 12px 24px

Campos de entrada

- Fondo: Gris claro
- Bordes redondeados: 8px
- Padding: 12px
- Placeholder en gris medio

Iconos

- Tamaño consistente: 24x24 pixels
- Color adaptado al contexto (blanco en barra lateral, gris en contenido)

4. Diseño responsivo

Breakpoints

- **Móvil**: < 768px
- **Tablet**: 768px - 1024px
- **Desktop**: > 1024px

Adaptaciones responsivas

- Barra lateral se convierte en menú hamburguesa en móvil
- Elementos del dashboard se reorganizan en columna única en móvil
- Fuentes se reducen en tamaños móviles

5. Estructura de pagina

Barra lateral

- Ancho: 250px en desktop
- Fondo: Morado principal
- Elementos:
    - Logo/Avatar del administrador
    - Enlaces de navegación con iconos
    - Botón de cerrar sesión en la parte inferior

Contenido principal

- Margen izquierdo: 250px (alineado con barra lateral)
- Padding: 24px
- Elementos:
    - Encabezado con título de bienvenida
    - Secciones de contenido con títulos descriptivos

6. Imágenes y gráficos

Ilustraciones

- Estilo: Flat design con colores complementarios
- Formato preferido: JPG o SVG para escalabilidad
- Tamaños:
    - Ilustración de login: 400x400px
    - Iconos de navegación: 24x24px

BUENAS PRÁCTICAS DE DESARROLLO

1. Introducción

Este documento tiene como objetivo establecer una serie de estándares y buenas prácticas para la programación tanto del lado del cliente como del servidor. Su propósito es garantizar la calidad, legibilidad y mantenibilidad del código, facilitando la colaboración entre desarrolladores.

2. Estructura del código

2.1 Organización de archivos y directorios

- Lado del servidor (PHP):

- Modelos y clases: Se organizan en el directorio /modelos/, subdividido en:
  - /modelos/handler/: Lógica de negocio.
  - /modelos/data/: Gestión de datos.
- Servicios: Los controladores de solicitudes y respuestas se almacenan en /servicios/.

- Lado del cliente (JavaScript):
  - Los componentes y scripts se organizan por funcionalidad en directorios específicos como /componentes/ o /scripts/.

3. Nombres de clases, métodos y variables

3.1 Clases

- PHP: Las clases deben usar PascalCase y ser descriptivas. Ejemplo: UsuarioHandler, ProductoData.
- JavaScript: Usar PascalCase para nombres de clases o componentes de React. Ejemplo: UserForm, ProductList.

3.2 Métodos y funciones

- PHP y JavaScript: Deben usar camelCase para nombrar métodos o funciones, siendo claros respecto a la acción que realizan.

- PHP: createUser(), getProductDetails()

- JavaScript: loadComponent(), fetchData()

3.3 Variables

- PHP:

- Se deben utilizar nombres descriptivos y en camelCase.

- Las constantes usan mayúsculas y guiones bajos. Ejemplo: ADMIN_API_URL.

- JavaScript:

- Variables: Usar let y const en lugar de var.

- Constantes: Nombres en mayúsculas con guiones bajos. Ejemplo: const API_ENDPOINT = '/api/users';

4. Comentarios en el código

Los comentarios son esenciales para mantener la claridad en el código. Se deben utilizar para explicar bloques complejos de código, pero evitar comentarios excesivos o innecesarios.

- PHP: Usar comentarios multilínea (/** ... */) para describir clases y funciones, y comentarios en línea (//) para aclarar acciones dentro de métodos.

- JavaScript: Similar a PHP, se usan comentarios multilínea (/** ... */) para funciones y comentarios en línea (//) para aclarar pasos en el código.

Ejemplo en PHP:

```
/**
 * Crea un nuevo usuario en la base de datos.
 * @param string $nombreUsuario
 * @param string $correo
 */
function createUser($nombreUsuario, $correo) {
    // Validar la entrada de usuario
}
```

Ejemplo en JavaScript:

```
/**
 * Función que carga un componente de manera asíncrona.
 * @param {string} path - Ruta del componente a cargar.
 */
async function loadComponent(path) {
    // Realizar la solicitud para obtener el componente
    const response = await fetch(path);
}
```

5. Manejo de errores

5.1 PHP:

Se recomienda utilizar try...catch para manejar excepciones. Además, se deben capturar los errores al interactuar con bases de datos y mostrar mensajes significativos.

```php
try {
    // Código que puede lanzar una excepción
} catch (Exception $e) {
    // Manejar el error
}
```

5.2 JavaScript:

Utilizar bloques try...catch para manejar errores en operaciones asíncronas (por ejemplo, con fetch).

```javascript
try {
    const data = await fetchData(API_URL);
} catch (error) {
    console.error('Error fetching data:', error);
}
```

6. Estándares para la base de datos

- Nombres de tablas: Deben ser en plural y en minúsculas, utilizando guiones bajos para separar palabras.

Ejemplo: tb_usuarios, tb_productos.

- Nombres de columnas: Deben ser descriptivos y en minúsculas. Ejemplo: id_usuario, nombre_producto.

- Llaves foráneas: Utilizar prefijos como fk_ para indicar relaciones entre tablas.

Ejemplo de creación de tabla:

```sql
CREATE TABLE tb_usuarios (
  id_usuario INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  correo VARCHAR(100) NOT NULL UNIQUE
);
```

7. Indentación y espacios

PHP y JavaScript: La indentación debe ser de 4 espacios para mejorar la legibilidad del código. No usar tabulaciones.

Ejemplo de indentación correcta en JavaScript:

```javascript
function openModal() {
    modal.show();
    modalTitle.textContent = 'Crear administrador';
}
```

8. Evitar el uso de variables globales

JavaScript: Minimizar el uso de variables globales para evitar colisiones. Encapsular el código en funciones o módulos.

9. Conclusión

Estos estándares aseguran que el código se mantenga claro, legible y fácil de mantener, promoviendo una colaboración eficiente y facilitando

TECNOLOGÍAS UTILIZADAS

1. Visual Studio

- **Definición**: Entorno de desarrollo integrado (IDE) utilizado para desarrollar la aplicación web y móvil. Visual Studio proporciona herramientas para la programación, pruebas, y despliegue de aplicaciones web y móviles.
- **Funcionalidades**:
    - Soporte para múltiples lenguajes de programación (C#, JavaScript, HTML, CSS, etc.).
    - Integración con sistemas de control de versiones como Git.
    - Depuración y pruebas avanzadas.

2. React (para la aplicación móvil)

- **Definición**: Librería de JavaScript utilizada para construir la interfaz de usuario de la aplicación móvil. React se enfoca en el desarrollo de componentes reutilizables para la creación de interfaces interactivas.
- **Funcionalidades**:
    - Creación de aplicaciones móviles con interfaz de usuario dinámica y eficiente.
    - Utilización del Virtual DOM para mejorar el rendimiento.
    - Compatible con React Native para el desarrollo multiplataforma (iOS y Android).

3. API REST

- **Definición**: Interfaz de programación de aplicaciones que permite la comunicación entre el frontend y el backend, utilizada para manejar solicitudes entre la aplicación web/móvil y la base de datos o servicios externos.
- **Funcionalidades**:
    - Envío y recepción de datos entre el cliente (web o móvil) y el servidor.
    - Utiliza métodos HTTP (GET, POST, PUT, DELETE) para realizar operaciones de CRUD (Crear, Leer, Actualizar, Eliminar).
    - Permite la integración de diferentes servicios y el acceso a bases de datos remotas.

4. Base de Datos (SQL/NoSQL)

- **Definición**: Sistema de gestión de bases de datos que almacena, gestiona y recupera la información de la aplicación.
- **Funcionalidades**:
    - Conexión a través de APIs para operaciones de lectura y escritura de datos.
    - Manejo de grandes volúmenes de datos.
    - Soporte para bases de datos relacionales (SQL) o no relacionales (NoSQL), según las necesidades de la aplicación.

5. Hosting Web

- **Definición**: Servicio de alojamiento utilizado para desplegar y publicar la aplicación web en línea. El hosting conecta la aplicación web con los usuarios y permite que sea accesible desde cualquier lugar.
- **Funcionalidades**:
    - Alojamiento de archivos de la aplicación web.
    - Manejo de bases de datos y APIs desde servidores remotos.
    - Seguridad y cifrado para proteger la información de la aplicación.

6. Generación de reportes

- **Definición**: Herramientas integradas para generar reportes basados en los datos de la aplicación. Estos reportes pueden ser exportados en formatos como PDF, Excel, etc.
- **Funcionalidades**:
    - Generación automática de reportes basados en la actividad de la base de datos.
    - Integración con herramientas de análisis y visualización de datos.
    - Personalización y exportación de reportes.

7. Frontend (HTML, CSS, JavaScript)

- **Definición**: Conjunto de tecnologías que se utilizan para el desarrollo de la interfaz de usuario de la aplicación web.
- **Funcionalidades**:
  - **HTML**: Estructura el contenido y define los elementos básicos de la página.
  - **CSS**: Aplica estilos y diseño visual a la interfaz.
  - **JavaScript**: Agrega interactividad y lógica dinámica a la página.

8. Backend (PHP, Node.js, u otros)

- **Definición**: La parte del servidor que maneja la lógica de negocio, la interacción con la base de datos y la autenticación de usuarios.
- **Funcionalidades**:
  - Procesamiento de datos y solicitudes desde la aplicación web y móvil.
  - Integración con APIs externas y bases de datos.
  - Manejo de la lógica de seguridad y autenticación de usuarios.

9. Servicios en la nube (Opcional)

- **Definición**: Plataformas como Azure, AWS o Google Cloud que permiten el almacenamiento y procesamiento de datos en la nube, lo que mejora la escalabilidad y accesibilidad de la aplicación.
- **Funcionalidades**:
  - Almacenamiento en la nube.

- Despliegue de aplicaciones y bases de datos en servidores remotos.
- Escalabilidad automática para manejar más tráfico o carga de trabajo.

10. Control de Versiones (Git)

- **Definición**: Herramienta de control de versiones utilizada para seguir el historial de cambios en el código fuente y colaborar con otros desarrolladores.
- **Funcionalidades**:
  - Gestión de versiones del código fuente.
  - Colaboración en equipo mediante ramas y solicitudes de extracción (pull requests).
  - Integración con plataformas como GitHub o GitLab.

INTERFACES DE USUARIO

Registro

1. Descripción:

La interfaz de registro permite a los usuarios ingresar la información necesaria para crear una cuenta.

2. Campos del Formulario

- Nombre Completo
  - Descripción: Ingreso del nombre completo del usuario.
  - Validación: No permite números.
- Correo Electrónico
  - Descripción: Se solicita un correo válido.

- Validación: Debe tener formato de email (ej. usuario@dominio.com).

- Teléfono
  - Descripción: Número de teléfono del usuario.
  - Validación: Solo números, formato internacional opcional.

- DUI
  - Descripción: Documento Único de Identidad
  - Validación: Debe seguir el formato de 9 dígitos (ej. 12345678-9).

- Ubicación
  - Descripción: Dirección o ubicación actual del usuario.
  - Validación: Texto, sin restricciones específicas.

- Nombre de Usuario
  - Descripción: Nombre único para la cuenta del usuario.
  - Validación: No permite caracteres especiales.

- Contraseña
  - Descripción: Contraseña segura para el usuario.
  - Validación: Debe contener al menos 8 caracteres, incluyendo letras y números.

Inicio de sesión (Login)

1. Descripción

La pantalla de inicio de sesión permite a los usuarios ingresar con sus credenciales (nombre de usuario y contraseña) previamente registrados en el sistema.

2. Campos del Formulario

- Nombre de Usuario
  - Descripción: Campo para ingresar el nombre de usuario registrado.
  - Validación: No permite caracteres especiales no válidos. Debe coincidir con un usuario existente.
- Contraseña

- Descripción: Campo para ingresar la contraseña vinculada al usuario.
- Validación: Debe ser la misma que la utilizada en el registro, y tener al menos 8 caracteres.

Interfaz de usuario de categorías

1. Descripción

La pantalla de **Categorías** permite al usuario visualizar una lista de las diferentes categorías disponibles en la aplicación. Estas categorías agrupan productos o servicios relacionados y facilitan la navegación dentro de la app.

2. Elementos de la Interfaz

- Listado de Categorías
  - Descripción: Muestra una lista con el nombre de cada categoría disponible. Las categorías se presentan en formato de lista o cuadrícula.

- Interacción: El usuario puede tocar una categoría para visualizar más detalles sobre los productos o servicios relacionados.

Interfaz de usuario de productos

1. Descripción

Esta pantalla muestra las diferentes opciones de calzado disponibles en la aplicación.

2. Elementos de la interfaz

- **Listado o cuadrícula de productos de calzado**
  - **Descripción**: Muestra una serie de productos de calzado, con elementos clave como:
    - **Imagen del calzado**: Una imagen representativa de cada producto.

- **Nombre del producto**: Nombre breve del tipo o marca del calzado.

- **Opción "Ver más"**
  - **Descripción**: Cada producto de calzado incluye un botón o enlace **"Ver más"** que lleva a una pantalla detallada donde se muestra más información del calzado (tallas disponibles, descripción detallada, etc.).

Interfaz de usuario de detalles de producto

1. Descripción

La **Interfaz de productos** muestra la información detallada de un producto seleccionado, proporcionando al usuario todo lo necesario para tomar una decisión de compra.

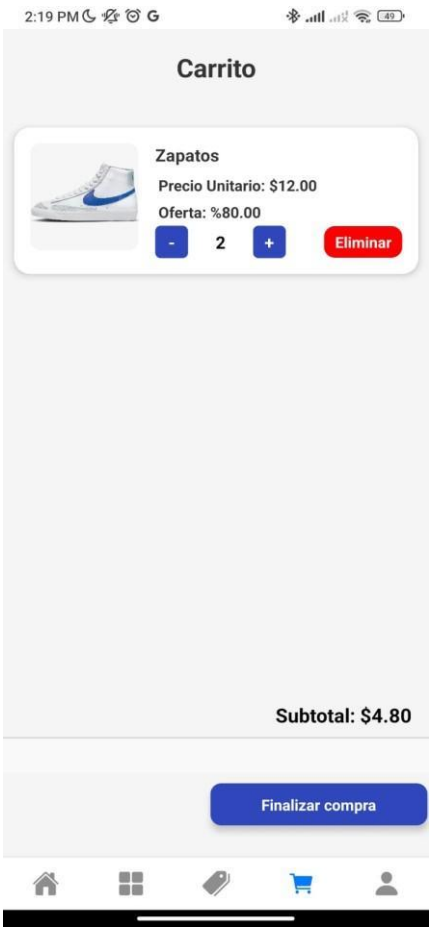
2. Elementos de la interfaz

- **Imagen del producto**
  - **Descripción**: Imagen de alta calidad que representa el producto. Puede incluir varias vistas o un carrusel de imágenes.
- **Nombre del producto**
  - **Descripción**: Nombre completo o título del producto, destacado en la parte superior de la pantalla.

- **Precio del producto**
  - **Descripción**: Precio actual del producto, mostrado de manera clara y destacada.

- **Descripción del producto**
  - **Descripción**: Texto que detalla las características, especificaciones y beneficios del producto.

- **Botón "Añadir al Carrito"**
  - **Descripción**: Permite al usuario añadir el producto a su carrito de compras para una posterior compra o procesamiento.
  - **Interacción**: Al tocar el botón, el producto se agrega al carrito y se actualiza el contador de productos en el carrito.



Interfaz de usuario de carrito de compra

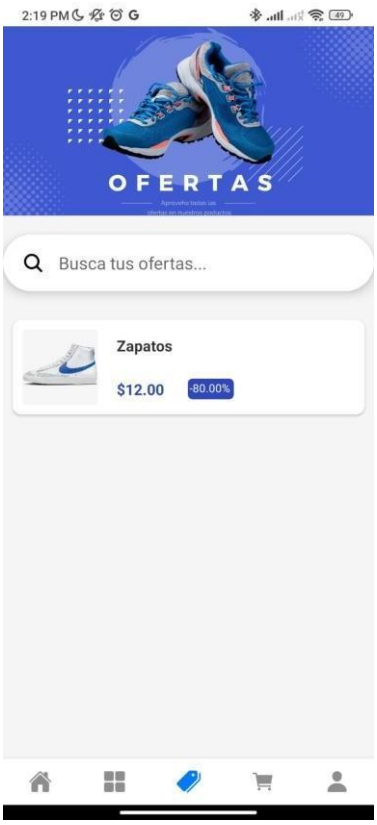
1. Descripción

La pantalla de Carrito de Compras permite a los usuarios revisar los productos que han agregado antes de finalizar su compra. Muestra un resumen de los productos seleccionados, el subtotal y una opción para proceder al pago.

2. Elementos de la Interfaz

- **Listado de Productos en el Carrito**
  - Descripción: Muestra los productos que el usuario ha agregado al carrito. Para cada producto, se muestra:
    - Nombre del Producto: Nombre del artículo.

- Imagen del producto: Imagen representativa del producto.
- Cantidad: Número de unidades del producto en el carrito.
- Precio: Precio por unidad del producto.
- Subtotal: Precio total por ese producto



Interfaz de usuario de ofertas

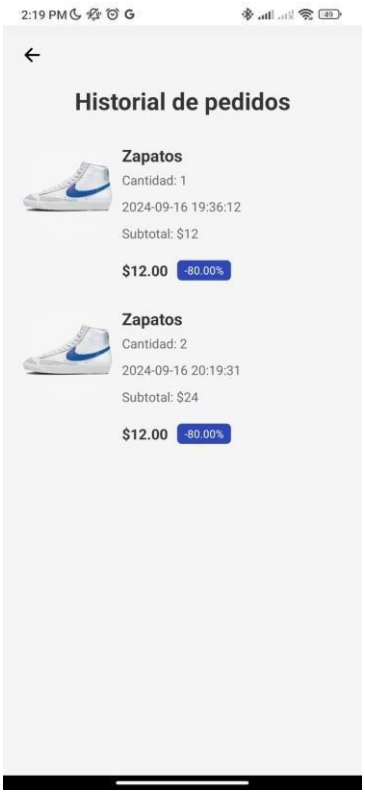
1. Descripción

La pantalla de **Ofertas** muestra productos o servicios que están en promoción o con descuento. Permite a los usuarios explorar las ofertas actuales y acceder a detalles de cada producto en promoción.

2. Elementos de la Interfaz

- **Listado de productos en Oferta**
  - Descripción: Se presenta una lista con los productos que tienen ofertas o descuentos especiales. Para cada producto se muestra:
    - Nombre del Producto: Nombre breve del producto.

- Imagen del Producto: Imagen representativa del producto.
- Precio Original: Precio anterior del producto (mostrado tachado o en un formato distinto para resaltar la oferta).
- Precio con Descuento: Precio actualizado con la oferta aplicada.
- Porcentaje de Descuento: Indicación visual del porcentaje de ahorro (ej. "-30%").



Interfaz de usuario de historial de pedidos

1. Descripción

La pantalla de Historial de Pedidos permite a los usuarios ver un registro de los pedidos realizados anteriormente. Proporciona detalles clave sobre cada pedido, como el nombre del producto, la cantidad, la fecha de la compra, el subtotal y el precio total.

2. Elementos de la Interfaz

- **Listado de pedidos anteriores**
  - **Descripción**: Muestra una lista de los pedidos realizados por el usuario. Para cada pedido se incluyen los siguientes elementos:
    - **Nombre del producto**: El nombre del producto o servicio adquirido.

- **Cantidad**: Número de unidades compradas de cada producto.
- **Fecha del pedido**: Fecha en la que se realizó la compra.
- **Subtotal**: Precio total por el producto o conjunto de productos antes de impuestos o descuentos.
- **Precio total**: Precio final pagado por el pedido, incluyendo impuestos y descuentos aplicados.

Interfaz de usuario de perfil (Vista externa)

1. Descripción

La pantalla de perfil (Vista externa) muestra información básica sobre la empresa, incluyendo enlaces a sus redes sociales y dos opciones clave que permiten al usuario acceder a su perfil personal y a los términos y condiciones de la aplicación.

2. Elementos de la interfaz

- Redes sociales de la empresa
    - Descripción: Se muestran iconos o enlaces directos a las redes sociales oficiales de la empresa (Facebook, Whatsapp, Instagram.).
    - Interacción: Al tocar un icono de red social, el usuario es redirigido al perfil oficial de la empresa en esa red social.
- Apartado "Perfil"

- Descripción: Enlace que lleva al usuario a su perfil personal donde puede modificar su información personal, como nombre, correo electrónico, contraseña, etc.
- Interacción: Al tocar este apartado, el usuario es redirigido a una pantalla donde puede editar sus datos personales y preferencias.

- Apartado "Términos y Condiciones"
  - Descripción: Enlace que lleva al usuario a una pantalla donde puede consultar los términos y condiciones del uso de la aplicación.
  - Interacción: Al tocar este apartado, el usuario es redirigido a una pantalla donde se muestran los términos y condiciones en formato de texto, para ser leídos y aceptados.

Interfaz de usuario de edición de perfil

1. Descripción

La pantalla de Edición de Perfil permite al usuario modificar su información personal dentro de la aplicación. Esta pantalla incluye campos editables como nombre, nombre de usuario, correo electrónico, contraseña y ubicación.

2. Elementos de la Interfaz

- **Campo "Nombre"**
    - **Descripción**: Campo donde el usuario puede modificar su nombre completo.
- **Campo "Usuario"**
    - **Descripción**: Campo para modificar el nombre de usuario (username) utilizado para iniciar sesión.
- **Campo "Correo Electrónico"**

- **Descripción**: Campo donde el usuario puede actualizar su dirección de correo electrónico.

- **Campo "Contraseña"**
  - **Descripción**: Campo para modificar la contraseña del usuario.

- **Campo "Ubicación"**
  - **Descripción**: Campo donde el usuario puede actualizar su dirección o ubicación.

Interfaz de usuario de términos y condiciones

1. Descripción

La pantalla de **Términos y Condiciones** permite a los usuarios leer las políticas y reglas de uso de la aplicación. Esta pantalla es informativa y está diseñada para que el usuario consulte y acepte los términos antes de utilizar los servicios de la aplicación.

GLOSARIO Y TÉRMINOS TÉCNICOS

1. Interfaz de usuario (UI)

- Definición: El conjunto de pantallas, páginas y elementos visuales con los que un usuario interactúa dentro de una aplicación o sistema. En este contexto, incluye las pantallas de registro, login, carrito de compras, entre otras.

2. Perfil

- Definición: Información personal del usuario almacenada en la aplicación, que puede incluir nombre, usuario, correo electrónico, contraseña, y ubicación. Los usuarios pueden editar su perfil para actualizar esta información.

3. Carrito de compras

- Definición: Una funcionalidad donde los usuarios pueden ver los productos que han seleccionado para comprar. Incluye información como nombre del producto, cantidad, precio y subtotal.

4. Subtotal

- Definición: La suma total de los productos agregados al carrito antes de aplicar impuestos o descuentos.

5. Ofertas

- Definición: Descuentos o promociones especiales aplicados a ciertos productos. En la interfaz de usuario, se muestra el precio original tachado, seguido del precio con descuento y el porcentaje de ahorro.

6. Historial de pedidos

- Definición: Una lista de compras anteriores realizadas por el usuario, que incluye detalles como el nombre del producto, la cantidad, la fecha de compra, el subtotal y el precio final pagado.

7. Redes Sociales

- Definición: Plataformas externas como Facebook, Instagram o Twitter, donde la empresa tiene presencia. Los enlaces a las redes sociales se incluyen en la pantalla de perfil para redirigir a los usuarios a las cuentas oficiales de la empresa

8. Términos y Condiciones

- Definición: Un conjunto de reglas y políticas que los usuarios deben aceptar para utilizar la aplicación. Esta sección cubre aspectos legales como privacidad, uso del servicio, derechos y obligaciones.

9. Navegación

- Definición: El proceso mediante el cual un usuario se mueve entre diferentes pantallas de la aplicación. Incluye acciones como tocar botones o enlaces que redirigen a otras secciones de la aplicación, como el perfil, historial de pedidos o el carrito de compras.

10. Validaciones

- Definición: Reglas que se aplican a los datos ingresados por el usuario para asegurar que cumplan con los requisitos del sistema (ej. formato de correo electrónico, longitud de la contraseña). En la interfaz de edición de perfil, cada campo tiene su propia validación.