Nombre de la materia:

Ingeniería en software

Nombre:

Carrasco Medina Carlos Ivan

Fecha:

31/05/2024

Unidad:

4

No. De Actividad:

Documento final

índice:

Contenido

Introducción:

En este documento encontraremos el primer avance del proyecto que se realizará a lo largo de este semestre en la materia de ingeniería en software en esta primera parte encontraremos el documento análisis de mi proyecto sobre la realización de una app que ayude al usuario a administrar el inventario de su propio negocio en esta parte nos enfocaremos en los requisitos que este tendrá que tener para su correcto desarrollo e uso.

Resumen del proyecto:

Este proyecto busca brindarle al usuario una función para administrar el inventario de su negocio ofreciéndole la posibilidad de agregar producto a un catálogo, eliminarlo de este, un filtro de búsqueda, un sistema de notificaciones para informar al usuario sobre el sistema y un pequeño apartado de ajustes para personalizar el uso al gusto del usuario.

Objetivos del proyecto:

- Fácil uso para usuarios nuevos de aplicaciones
- Facilidad de búsqueda de los productos
- Versatilidad de uso para cualquier tipo de negocio

Alcance del proyecto:

La app contará con los espacios para agregar el producto y darlo de baja al realizar la venta, contará con un pequeño tutorial que especificará que hará cada componente de este.

Análisis del requerimiento:

1. Introducción

En este apartado encontraremos la Especificación de Requisitos Software (ERS) sobre el proyecto de una aplicación Android sobre un inventario y sus derivados.

1.1. Propósito

Este documento se centrará en explicar que es lo que se necesitara para la creación de software de forma que pueda ser presentado tanto a un posible cliente como a un compañero de trabajo.

1.2. ámbito del Sistema

El sistema actualmente denominado “Mi Inventario” este se encargará del almacenamiento de distintos productos que serán agregados por el usuario estos contara con los siguientes datos: Fotografía, nombre, precio, cantidad, descripción y etiquetas.

También contará con la opción de dar de baja un producto y realizar una venta colocando los distintos productos con un sistema de búsqueda que contendrá filtro por nombre o etiquetas y dará el precio de este conjunto después de finalizar la venta se descontarán los productos de la cantidad de estos.

Este no contara con base de datos ya que al ser de uso particular no podemos saber si el cliente contara con recursos para mantenerla (aunque si el usuario lo requiere se puede agregar), no contara con un sistema de ubicación de productos y no contara con animaciones la mayoría de las veces.

Con esto queremos llegar a crear una aplicación para brindar apoyo a los negocios que busque una manera más sencilla de manejar el inventario de este facilitando el uso de la app por parte cualquier persona sea o no usuaria frecuente de apps.

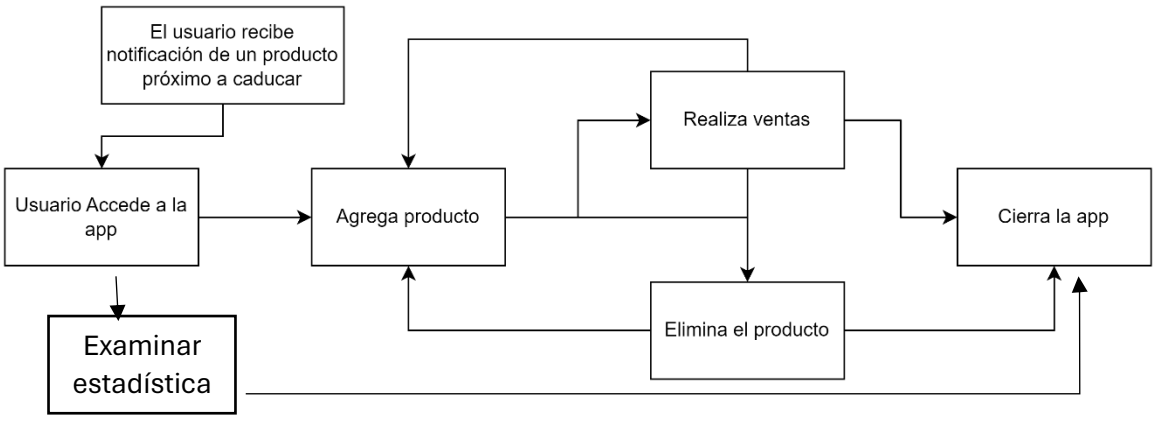
2. Descripción General

En este proyecto nos enfrentamos a distintos factores que pueden cambiar el rumbo del desarrollo de la aplicación como que versión de Android debe estar disponible la aplicación, que dispositivo se tiene que usar de referencia y que subsecciones requiere esta.

2.1. Perspectiva del Producto

El producto es independiente a cualquier otro producto no necesita de un servicio después de la instalación mas que el propio que le pueda proporcionar el sistema operativo, busca brindarle al usuario un sistema sencillo para administra su empresa pudiendo agregar, eliminar y realizar ventas.

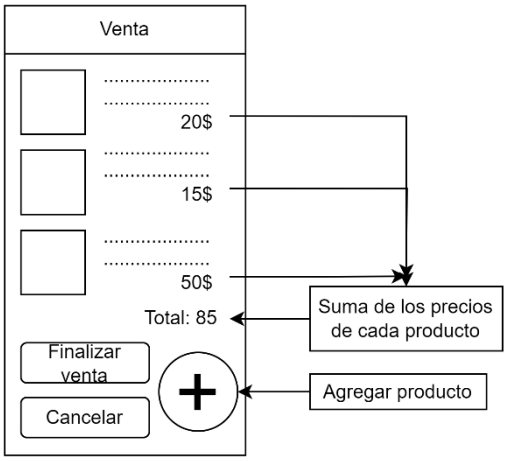
Diagrama para ejemplificar el uso esperado de la app.



2.2. Funciones del Producto:

- **Agregar productos:** se utilizará para dar de alta un nuevo producto.

- **Eliminar producto:** se utilizará para dar de baja definitivamente un producto.
- **Notificar al usuario:** recibirá notificaciones cada que un producto este próximo a caducar.
- **El sistema contara con modo oscuro y claro:** este ayudara a que el usuario se sienta cómodo usando el producto.
- **EL sistemas proporcionara una serie de estadísticas que se mostraran con gráficos y otros detalles que ayuden a la comparación.**
- **Venta de productos:**
  Se seleccionarán los productos y se les dará el precio de la venta a su vez:
  - **Búsqueda con filtros:** cuando se quiera buscar un producto en específico se le proporcionara un filtro que puede ser por nombre o etiquetas.
  - **Suma de precios:** cada producto tiene un precio, estos se sumarán automáticamente
  - **Descuento de producto:** al terminar la venta estos productos desaparecerán del conteo.



2.3. Características de los Usuarios

Se espera que el usuario tenga la capacidad de lectura básica y escritura mínima no requiere otro tipo de conocimiento ya que esta aplicación está pensada para ser tanto simple de usar como eficiente y capaz.

2.4. Restricciones

No se buscarán crear intricadas opciones para el usuario.

Políticas de la empresa: sencillo y eficaz

Limitaciones del hardware: se espera que funcione solo de Android 5.0 lolipop y versiones posteriores.

Operaciones paralelas: Al realizar ventas también se descuentan del inventario y siempre que un producto se acerque a su fecha de caducidad se notificara al usuario.

2.5. Suposiciones y Dependencias

Se tiene supuesto que el sistema operativo que se usará será Android 5.0 en adelante si se quiere agregar se tendrá que restructurar ciertas funciones, lo mismo si se piensa expandir a ios se expandirá el tiempo.

2.6. Requisitos Futuros

Se espera agregar una opción para poder trasladar la información de un dispositivo a otro por lo que se requerirá de una aplicación externa como bluetooth o Google drive.

3. Requisitos Específicos

- App para dispositivos Android: se usará un motor como Android studio o godot engine 3.5. (por limitaciones técnicas se cambió a flutter)
- Android 5.0 lolipop en adelante.
- La app debe contener 5 modulo:
    - El módulo principal(main):
        - contará con 3 apartados principales que se dividirán a los otros 3 módulos (Agregar productos, Realizar venta y dar de baja un producto) y hasta arriba a la derecha el apartado de notificaciones.
        - También como notificaciones contara con un submenú que mostrara distintas configuraciones que afectaran al resto de módulos.
            - Ajustes:
                - Diseño oscuro: se mostrará una opción de activar o no este modo para cambiar el diseño
                    - Ejemplo:

                      Desactivado Activado

                      
                - Información: se muestra detalles de la app como año de desarrollo, nombre del programador, etc.
    - Modulo con un menú para agregar productos este recolectara los siguientes datos:

        - Nombre: Una lista de caracteres (String, varchar)
        - Cantidad: numérico entero (integer)
        - Etiquetas: Un arreglo de listas de caracteres
        - Fotografía: imagen formato (jpeg, png)
        - Fecha de caducidad: formato de fecha (datatime)

        - Después de rellenar los campos al usuario se le mostrara un mensaje o no dependiendo de los siguientes casos:
            - Caso 1, si no relleno todos los campos: se le mostrara un mensaje de que se asegure de rellenarlos o seguir adelante.
            - Caso 2, si relleno todos los campos o siguió adelante: se guardará el producto el catálogo.
    - Menú para vender producto:
        - Selección de productos se mostrará el catálogo de los productos que están disponibles (la cantidad de estos debe ser mayor a 0) se podrá deslizar hacia abajo para ver más productos y al seleccionar se sumará al precio total.
        - Contará con un sistema de búsqueda por filtros (este se ubicará en la parte de arriba) este será por nombre o por etiquetas.
        - En la parte de abajo estará el apartado que mostrara el precio total de la venta que como se mencionó en el primer punto se sumará cada que se seleccione un producto.
    - Menú para dar de baja:
        - Este menú será parecido al de realizar ventas
        - Selección de productos se mostrará el catálogo completo de los productos sin importar la cantidad, como en el apartado vender se podrá deslizar hacia abajo para ver más productos.
        - Contará con un sistema de búsqueda por filtros (este se ubicará en la parte de arriba) este será por nombre o por etiquetas.
        - En la parte de abajo se mostrará en rojo muy llamativo una opción que diga eliminar al dar clics, se mostrará un mensaje para asegurarse de que quiere eliminar esos productos.
            - Caso 1, cancelar: se vuelve a la pantalla anterior.
            - Caso 2, Eliminar: se borrará el producto del catálogo permanentemente a su vez volverá a la pantalla anterior actualizada para no mostrar los productos anteriores.
    - Apartado de notificaciones: (se excluyó por razones de lógica)

        - En la esquina superior abra un pequeño circulo con un símbolo de campanita al dar clics se mostrará un historial de las ultimas notificaciones.
        - Estas notificaciones llegaran automáticamente al detectar que un producto tiene una fecha cercana a una semana de diferencia.
        - Para este apartado se requiere tener acceso al reloj del sistema.
    - Apartado para mostrar estadísticas:
        - Productos más vendidos
        - Productos menos vendidos
        - Productos próximos a vencer (se excluyó por razones de lógica)
        - Entre otros…

3.1. Funciones

- El módulo principal (pantalla de inicio de la aplicación) solo podrá acceder a los otros módulos y contara con un apartado de ajuste que afectaran los otros módulos.
- El módulo para agregar producto solo recogerá datos del formato de un producto y lo agregará a una lista que se usará en los otros dos módulos
- El módulo para realizar una venta se seleccionar los productos y te dará el precio final y les descontará el número de productos vendidos de la cantidad de cada producto.
- El módulo para dar de baja un producto, selecciona los productos que quiera eliminar para siempre.
- El módulo para administrar las notificaciones se mostrarán las notificaciones anteriormente recibidas.
- El sistema enviara notificaciones de los productos cercanos a caducar.
- **El módulo para que se muestren las estadísticas**

3.2. Restricciones de Diseño

Simplista se buscarán formas geométricas básicas como círculos, elipses o rectángulos con puntas redondeadas, se buscará evitar figuras con algunos rectos como rectángulos y cuadrados, los colores seguirán una gama de azules y verdes.

3.3. Atributos del Sistema

- Android 5.0 lolipop en adelante.
- La gama baja será suficiente.
- Como será una aplicación mayormente offline no necesitará un usuario ni contraseña.

4. Apéndices

4.1 Trazabilidad de requerimientos:

a. Matriz de trazabilidad de requerimientos:

Id	Requisito	Tipo	prio	estado	objetivo	Funcionalidad(es)	Estado
1	El sistema permitirá dar de alta productos con nombre, descripción, foto, etiquetas y cantidad.	Administrativo	Alto	Activo	Crear una lista personalizada por el usuario	Detectar si el producto ya existe si no permite agregarlo, también permite aumenta la cantidad de otro producto.	Pendiente
2	El sistema permitirá dar de baja productos.	Administrativo	Alto	Activo	Permite liberar memoria y facilita la búsqueda.	Selecciona un producto y lo borra para siempre	Pendiente
3	El sistema permitirá realizar ventas	Venta	Medio	Activo	Facilita realizar ventas al usuario.	Permite seleccionar una lista de productos que y dará el precio total.	Pendiente
4	El sistema descontara los productos que se seleccionaron en la venta	Venta	Medio	Activo	Garantizara que los productos vendidos no sigan en el inventario.	Realiza una acción que al usuario le daría problema	Pendiente
5	El sistema contara con un buscador por filtros	Venta/ Administrativo	Baja	Activo	Facilitará la selección de productos	Permite encontrar productos al realizar la venta o dar de baja aun producto	Pendiente
6	El sistema enviara una notificación al usuario cada que un producto este próximo a caducar.	Administrativo	Bajo	Activo	Evitar el desperdicio de producto, pudiendo el usuario realizar ofertas para no perder dinero	El usuario podrá reducir la cargar de recordar ver la fecha de cada producto.	Pendiente
7	El sistema permitirá cambiar entre modo oscuro o modo claro.	Comodidad	Bajo	Activo	Permite que el usuario descanse la vista al usar la app	Selecciona modo de visualización del programa para mayor comodidad	Pendiente

b. Casos de prueba:

i. **Caso de prueba registrar un producto:**

1. Objetivo: verificar que el usuario pueda agregar un producto correctamente.
2. Precondiciones: el usuario ha abierto la app
3. Pasos:
    a. Ingresar al menú agregar producto
    b. rellenar cada uno de los campos
    c. dar a agregar producto
    d. y aceptar

ii. **Caso de prueba dar de baja un producto:**

1. Objetivo: verificar que el usuario pueda dar de baja un producto correctamente.
2. Precondiciones: el usuario ha abierto la app
3. Pasos:
    a. Ingresar al menú dar de baja/eliminar producto

b. Seleccionar los productos a eliminar
c. dar a eliminar
d. y aceptar

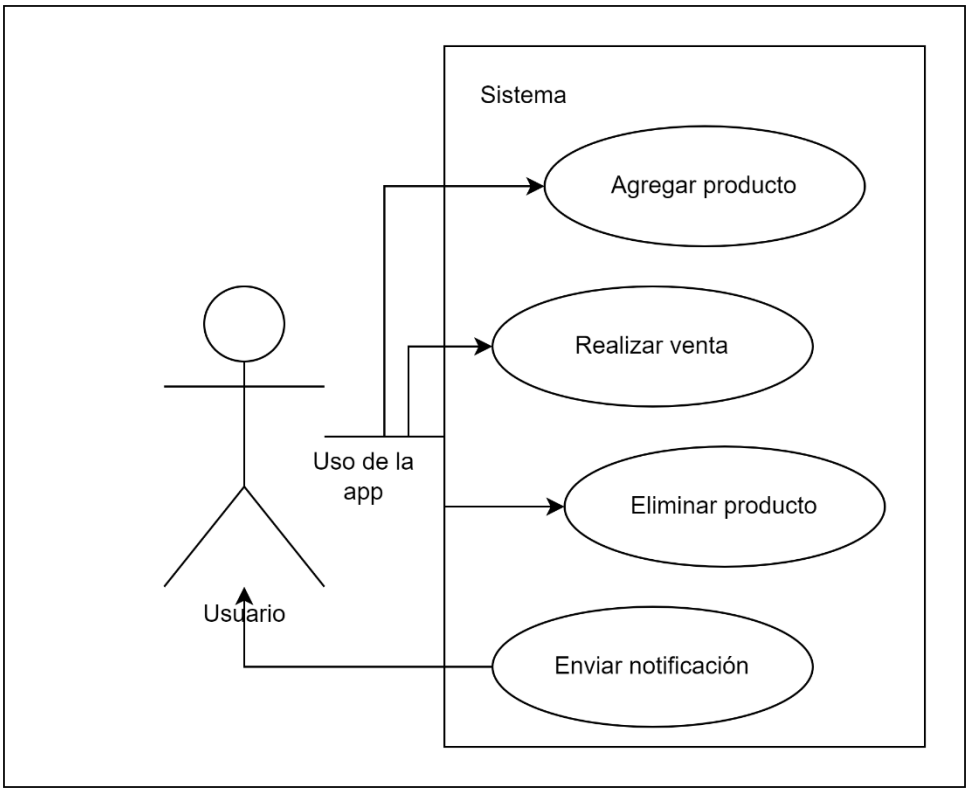
iii. Caso de prueba realizar una venta.

1. Objetivo: verificar que el usuario pueda realizar una venta correctamente.
2. Precondiciones: el usuario ha abierto la app
3. Pasos:
   a. Ingresar al menú realizar venta
   b. Seleccionar los productos que tendrá la venta
   c. dar a finalizar venta
   d. y aceptar

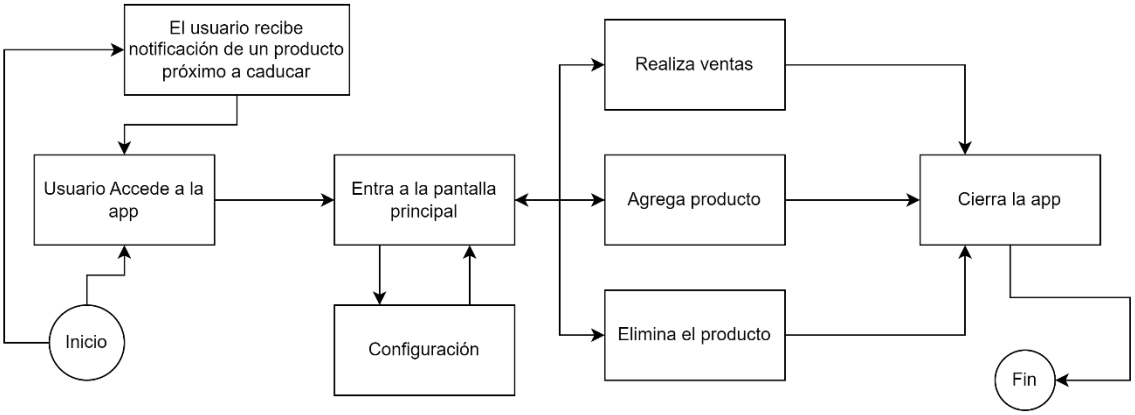
iv. Caso de prueba recibir notificaciones

1. Objetivo: verificar que el usuario reciba las notificaciones correctamente.
2. Precondiciones: el usuario debe haber ingresado una app próxima a caducar
3. Pasos:
   a. Revisar si la notificación apareció en el teléfono.

4.2 Casos de uso



4.3 Diagrama de estado



4.4 Resultados de análisis de costes.

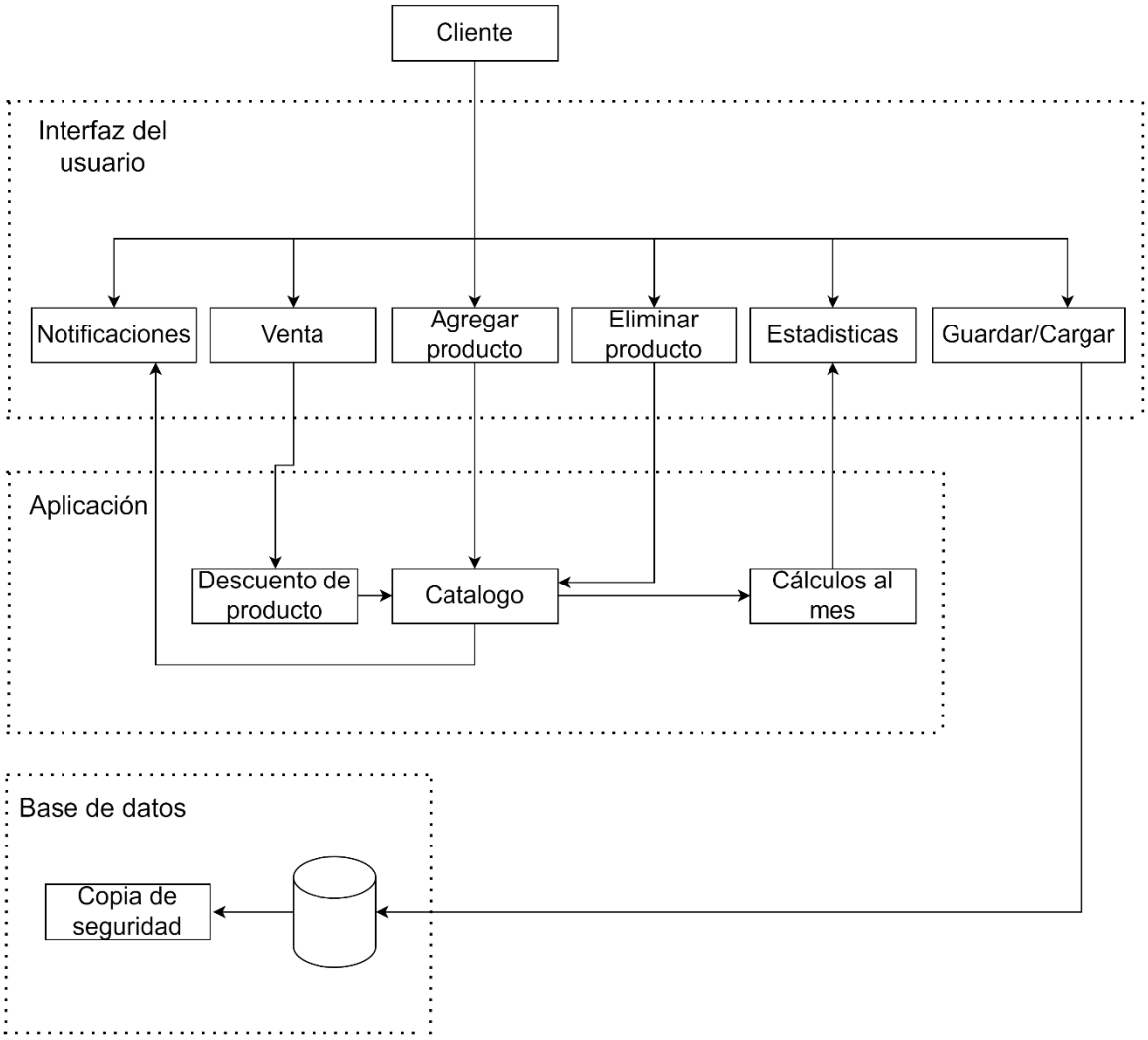
a. Costo-Beneficio:

- Según lo investigado se a llegado a un presupuesto estimado de cuánto tiempo tomara realizar dicha app el cual sería de 14-16 días a una tarifa de 600 pesos el día nos daría 8400$ a 9600$ la realización del software
- Según las investigaciones hemos concluido que en promedio se pierden cerca de 500$ mensuales y 50$ en producto desaparecido también mensuales
- Por lo que llego a la siguiente conclusión:

- Solo se obtendrán beneficios después de un año.

5.diseño

5.1 Modelo grafico de la arquitectura:



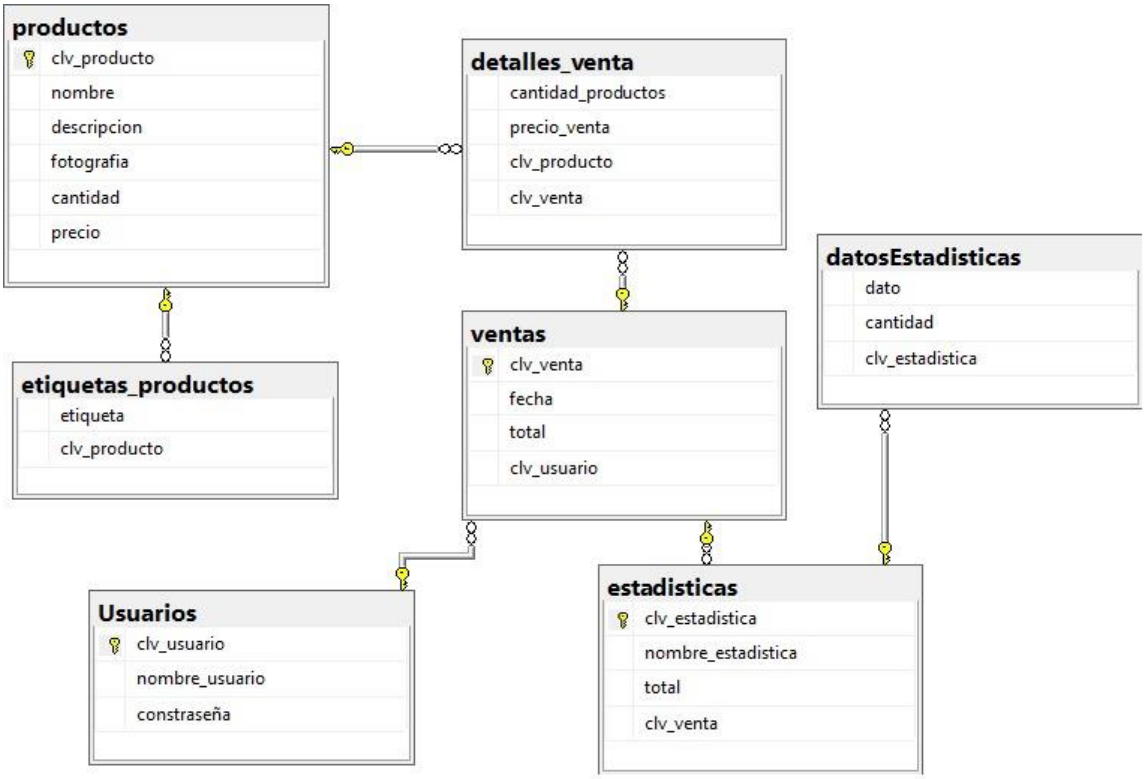
Tecnologías utilizadas en la arquitectura:

- Para la base de datos usare sql server.
- Firebase para alojar el servidor.
- Android studio:
    - Sdk 21
    - Api Level 21 Lolipop
    - Jdk 8

--por problemas técnicos se decidió cambiar

Android studio por flutter apoyado por visual studio code

5.2 Modelo y diseño de la base de datos:



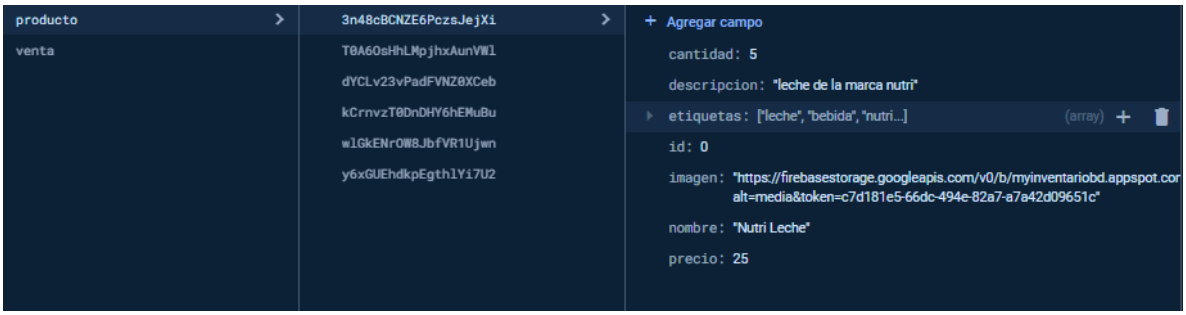
5.3 Sql:

```
create database MyInventario
go
use MyInventario
go
create table Usuarios
(
        clv_usuario int primary key not null,
        nombre_usuario nvarchar(20),
        constraseña nvarchar(50) not null
)

create table productos(
        clv_producto int primary key not null,
        nombre nvarchar(50),
```

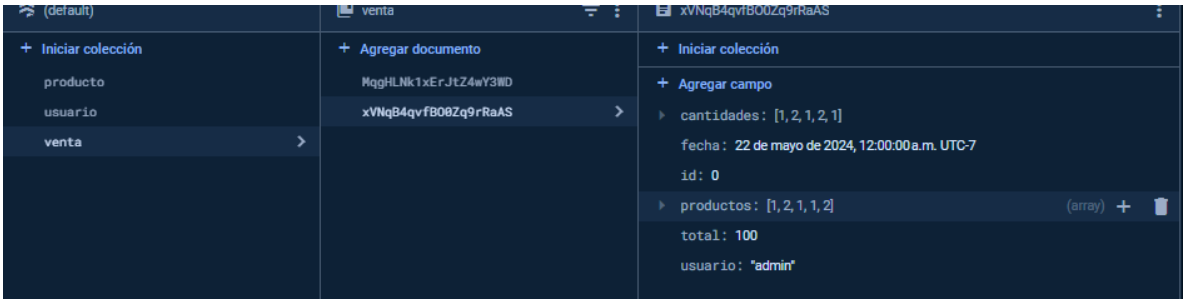
```
	descripcion nvarchar(150),
	fotografia varbinary (max),
	cantidad int,
	precio money
)

create table etiquetas_productos(
	etiqueta nvarchar,
	clv_producto int foreign key references productos(clv_producto)
)

create table ventas(
	clv_venta int primary key not null,
	fecha date,
	total money,
	clv_usuario  int foreign key references usuarios(clv_usuario)
)

create table detalles_venta(
	cantidad_productos int,
	precio_venta money,
	clv_producto int foreign key references productos(clv_producto),
	clv_venta int foreign key references ventas(clv_venta)
)


create table estadisticas(
	clv_estadistica int primary key not null,
	nombre_estadistica nvarchar(100),
	total int,
	clv_venta int foreign key references ventas(clv_venta)
)

create table datosEstadisticas(
	dato nvarchar,
	cantidad int,
	clv_estadistica int foreign key references estadisticas(clv_estadistica),
)
```

5.4 firebase

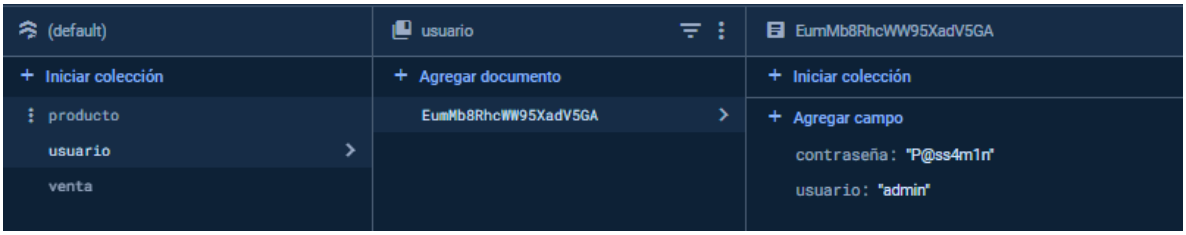
Producto:



Venta

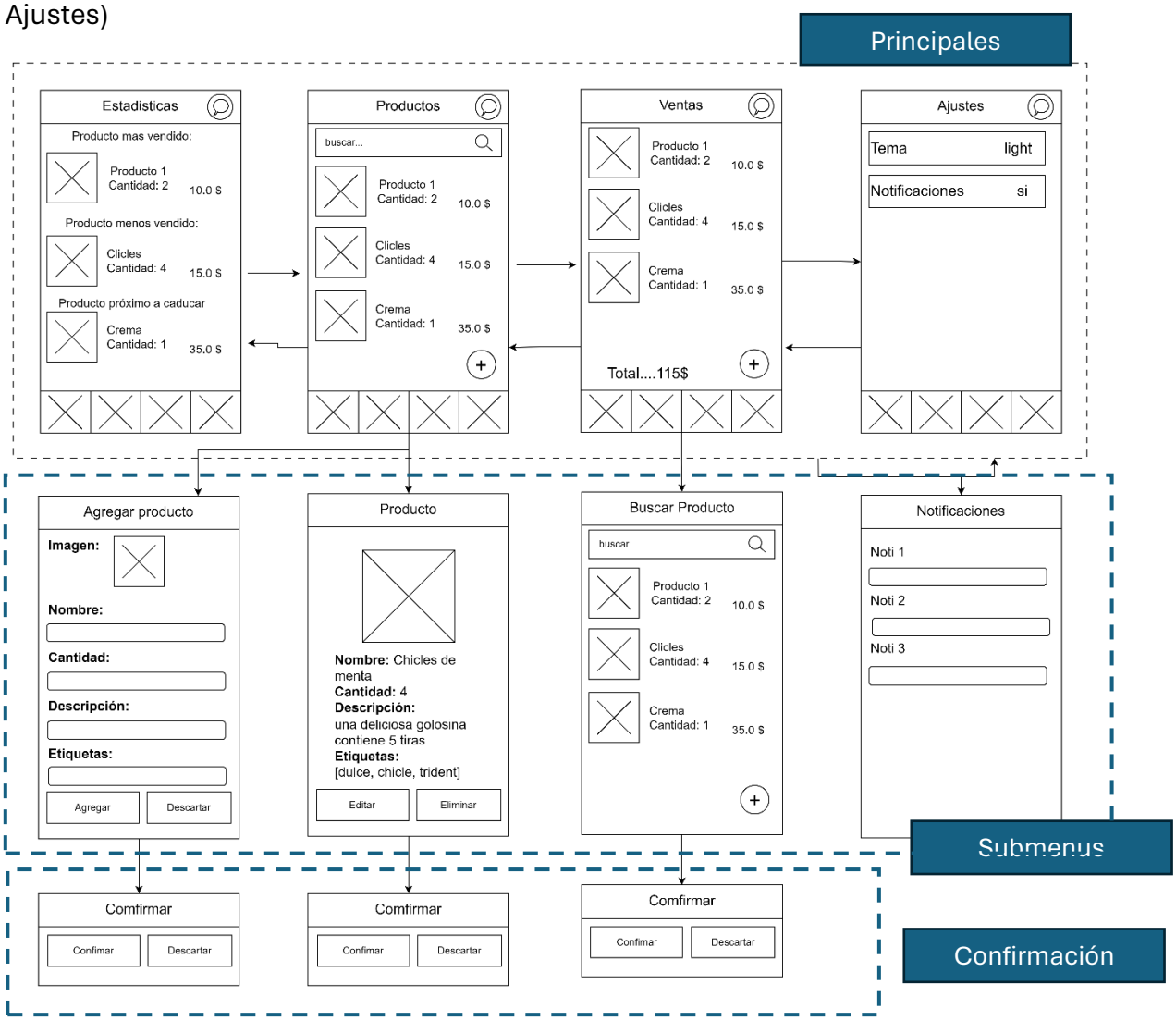


Usuario

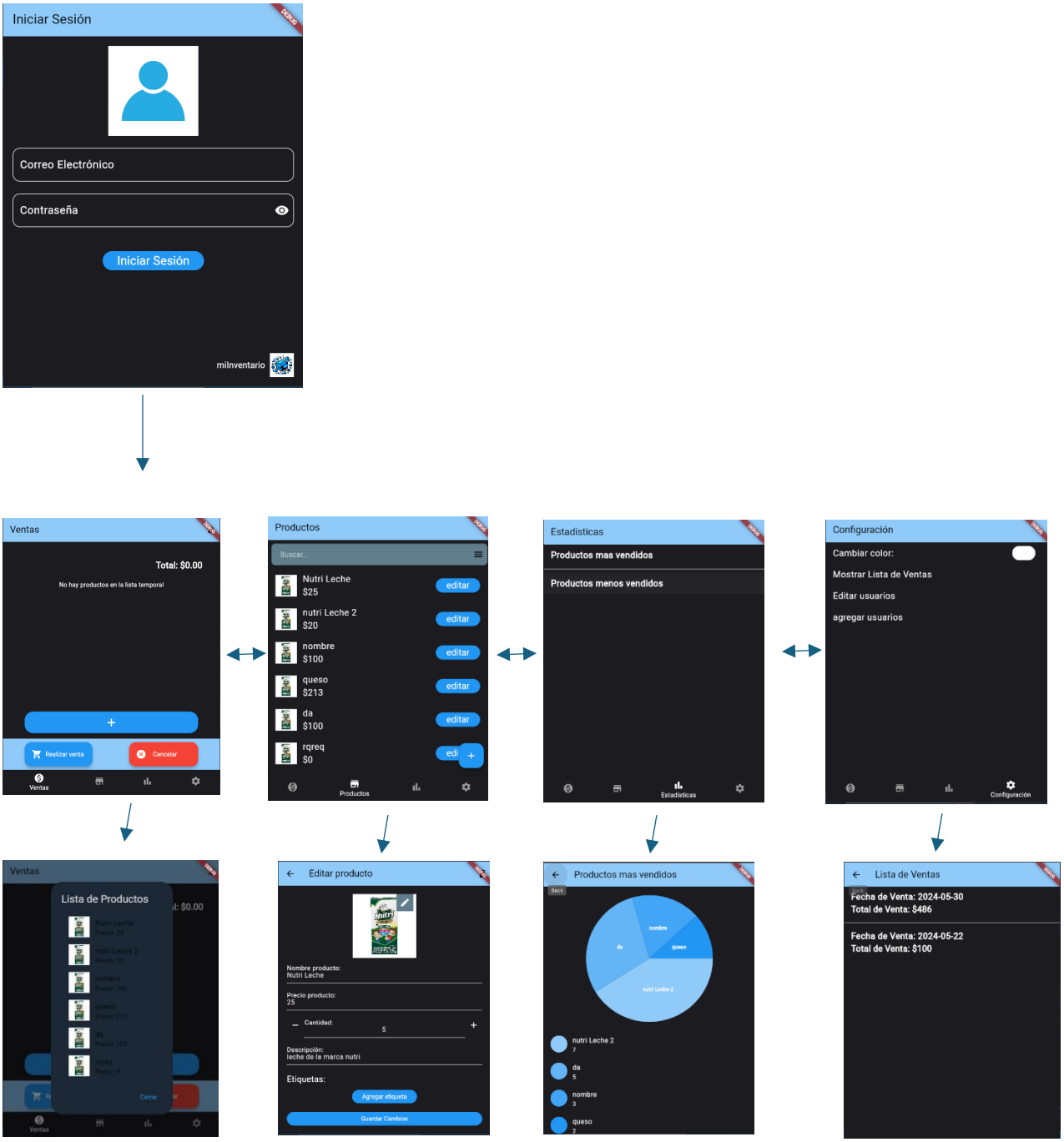


5.5 Interfaz de usuario (wareframes):

Nota: se puede deslizar entre las primeras cuatro (Estadísticas, Productos, Ventas, Ajustes)



5.6 warframe (alto nivel)



6. código

Ejemplo de clase

```
class Producto {
  int id, cantidad;
  String nombre;
  double precio;
  String imagen;
  String descripcion;
  List<String> etiquetas;

  Producto({
    required this.id,
    required this.cantidad,
    required this.nombre,
    required this.precio,
    required this.imagen,
    required this.descripcion,
    required this.etiquetas
  });

  // Método para convertir un documento Firestore en un Producto
  factory Producto.fromFirestore(DocumentSnapshot doc) {
    Map data = doc.data() as Map<String, dynamic>;
    return Producto(
      cantidad: data['cantidad'],
      descripcion: data['descripcion'],
      etiquetas: List<String>.from(data['etiquetas']),
      id: data['id'],
      //imagen: data['imagen'],
      imagen: 'assets/nutrileche.jpg',
      nombre: data['nombre'],
      precio: data['precio'],
    );
  }
}
```

Ejemplo de clases y pantallas

```
import 'package:flutter/material.dart';
import 'package:fl_chart/fl_chart.dart';
import 'package:myinventory/catalogo.dart';
import 'package:myinventory/main.dart';
import 'package:myinventory/Ventas.dart';
```

```dart
class Estadisticas extends StatefulWidget {
  final Color colorLetra;
  final Function(Color) onColorChanged;
  final List<Venta> listaVenta;
  const Estadisticas({super.key, required this.colorLetra, required
this.onColorChanged, required this.listaVenta});

  @override
  // ignore: library_private_types_in_public_api
  _EstadisticasLista createState() => _EstadisticasLista();
}

class _EstadisticasLista extends State<Estadisticas> {
  late Color colorLetra = widget.colorLetra;
  final PageController _pageController = PageController();

  @override
  void initState() {
    super.initState();
    selectedIndex = 2;
  }

  void _onItemTapped(int index) {
    setState(() {
      selectedIndex = index;
      _cambiarPantalla();
    });
  }

  void _cambiarPantalla(){
    if (selectedIndex == 0) { // Verifica si se seleccionó la opción de
"Productos"
      Navigator.popUntil(context, ModalRoute.withName('/')); // Cierra todas
las rutas hasta llegar a la inicial

      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => MyHomePage(title: 'Ventas',
colorLetra: colorLetra)),
      );
    }
    else if (selectedIndex == 1) { // Verifica si se seleccionó la opción de
"Estadisticas"
      Navigator.popUntil(context, ModalRoute.withName('/')); // Cierra todas
las rutas hasta llegar a la inicial
```

```
      Navigator.push(
        context,
        MaterialPageRoute(builder: (context) => Catalogo(productos:
ListaDeProductos, colorLetra: colorLetra, onColorChanged:
widget.onColorChanged)),
      );
    }
    else if (selectedIndex == 3) { // Verifica si se seleccionó la opción de
"Estadisticas"
      Navigator.popUntil(context, ModalRoute.withName('/')); // Cierra todas
las rutas hasta llegar a la inicial

      Navigator.pushNamed(context, '/configuraciones', arguments:
ListaDeProductos);
    }
  }

  @override
  void dispose() {
    _pageController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.background,
        title: const Text('Estadisticas'),
        automaticallyImplyLeading: false,
      ),
      body: ListView(
        children: [
          ListTile(
            title: Text('Productos mas vendidos', style: TextStyle(
                        fontSize: 20,
                        fontWeight: FontWeight.bold,
                        color: colorLetra,
                      ),),
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(builder: (context) =>
_topVentasPantalla(colorLetra: widget.colorLetra)),
```

```
            );
          },
        ),
        const Divider(),
        ListTile(
          title: Text('Productos menos vendidos', style: TextStyle(
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                      color: colorLetra,
                    ),),
          onTap: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) =>
_menosVentasPantalla(colorLetra: widget.colorLetra)),
            );
          },
        ),
        // Agrega más opciones según sea necesario
      ],
    ),
    //=========================================================================
=========================
    //      BARRA DE NAVEGACION
    //=========================================================================
=========================
    bottomNavigationBar: BottomNavigationBar(
      //colores
      selectedItemColor: widget.colorLetra,
      unselectedItemColor: Colors.grey,
      selectedLabelStyle: TextStyle(color: widget.colorLetra),
      unselectedLabelStyle: const TextStyle(color: Colors.grey),
      selectedIconTheme: IconThemeData(color: widget.colorLetra),
      unselectedIconTheme: const IconThemeData(color: Colors.grey),
      //botones e iconos
      items: const <BottomNavigationBarItem>[
        BottomNavigationBarItem(
          icon: Icon(Icons.monetization_on),
          label: 'Ventas',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.store),
          label: 'Productos',
        ),
        BottomNavigationBarItem(
```

```
            icon: Icon(Icons.bar_chart),
            label: 'Estadísticas',
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.settings),
            label: 'Configuración',
          ),
        ],
        currentIndex: selectedIndex,
        onTap: _onItemTapped,
      ),
    );
  }
}

// ignore: must_be_immutable, camel_case_types
class _topVentasPantalla extends StatelessWidget {
  final Color colorLetra;
  _topVentasPantalla({required this.colorLetra});

  List<Venta> listaVenta = List.of(ListaDeVentas);
  List<PieChartSectionData> sectionsChart = [];
  List<Color> coloresGrafica = [
    Colors.blue[100]!, // Azul 100
    Colors.blue[200]!, // Azul 200
    Colors.blue[300]!, // Azul 300
    Colors.blue[400]!, // Azul 400
    Colors.blue[500]!, // Azul 500];
  ];

  int contador = 0;
  Color _regresaColor(){
    contador == 4? contador = 0:contador++;
    return coloresGrafica[contador];
  }

  // Método para ordenar el mapa por valor (cantidad del producto)
  Map<String, int> ordenarPorCantidad(Map<String, int> map) {
    var sortedEntries = map.entries.toList()
      ..sort((a, b) => b.value.compareTo(a.value)); // Ordena por valor

    // Crea un nuevo mapa a partir de las entradas ordenadas
    var sortedMap = <String, int>{};
    for (var entry in sortedEntries) {
      sortedMap[entry.key] = entry.value;
```

```
    }
    return sortedMap;
  }

  // ignore: non_constant_identifier_names
  void _EncontrarTop(){
    Map<String, int> conteoProductos = {};

    for (var venta in listaVenta) {
      for (int i = 0; i < venta.productos.length; i++) {
        String nombreProducto = venta.productos[i].nombre;
        int cantidad = venta.cantidades[i];

        if (conteoProductos.containsKey(nombreProducto)) {
          conteoProductos[nombreProducto] = conteoProductos[nombreProducto]!
+ cantidad;
        } else {
          conteoProductos[nombreProducto] = cantidad;
        }
      }
    }

    // Ordenar los productos por cantidad vendida en orden descendente
    conteoProductos = ordenarPorCantidad(conteoProductos);

    if(conteoProductos.length > 5){
      // Crear un nuevo mapa con las claves originales y los primeros 5
valores
      var nuevoConteoProductos = Map.fromEntries(
        conteoProductos.entries.take(5),
      );

      conteoProductos = nuevoConteoProductos;
    }

    // Crear una lista de datos para el gráfico de pastel
    List<PieChartSectionData> generarSeccionesParaGrafico(Map<String, int>
conteo, Color colorLetra) {
      return conteo.entries.map((entry) {
        return PieChartSectionData(
          color: _regresaColor(), // Asegúrate de definir correctamente la
función _regresaColor()
          value: entry.value.toDouble(), // Convierte la cantidad a double
          title: entry.key, // Usa la clave como título
          radius: 150,
```

```
          titleStyle: TextStyle(fontSize: 12, fontWeight: FontWeight.bold,
color: colorLetra),
        );
      }).toList();
    }

    // Llamada a la función con tu mapa conteoProductos
    sectionsChart = generarSeccionesParaGrafico(conteoProductos,
colorLetra);
  }

  @override
  Widget build(BuildContext context) {
    _EncontrarTop();
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.background,
        title: const Text('Productos mas vendidos'),
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(20.0),
            child: SizedBox(
              width: MediaQuery.of(context).size.width,
              height: 300,
              child: PieChart(
                PieChartData(
                  borderData: FlBorderData(
                    show: false,
                  ),
                  sectionsSpace: 0,
                  centerSpaceRadius: 0,
                  sections: sectionsChart,
                ),
              ),
            ),
          ),
          Expanded(
            child: SingleChildScrollView(
              child: Column(
                children: [
                  for (int i = 0; i < sectionsChart.length; i++)
                    ListTile(
```

```
                      leading: CircleAvatar(backgroundColor:
sectionsChart[i].color),
                      title: Text(
                        sectionsChart[i].title.toString(),
                        style: TextStyle(color: colorLetra), // Cambia el
color del texto
                      ),
                      subtitle: Text(
                        sectionsChart[i].value.toStringAsFixed(0),
                        style: TextStyle(color: colorLetra), // Cambia el
color del texto
                      ),
                    ),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }
}

// ignore: must_be_immutable, camel_case_types, unused_element
class _menosVentasPantalla extends StatelessWidget {
  final Color colorLetra;
  _menosVentasPantalla({required this.colorLetra});

  List<Venta> listaVenta = List.of(ListaDeVentas);
  List<PieChartSectionData> sectionsChart = [];
  List<Color> coloresGrafica = [
    Colors.red[100]!, // Azul 100
    Colors.red[200]!, // Azul 200
    Colors.red[300]!, // Azul 300
    Colors.red[400]!, // Azul 400
    Colors.red[500]!, // Azul 500];
  ];

  int contador = 0;
  Color _regresaColor(){
    contador == 4? contador = 0:contador++;
    return coloresGrafica[contador];
  }

  // Método para ordenar el mapa por valor (cantidad del producto)
```

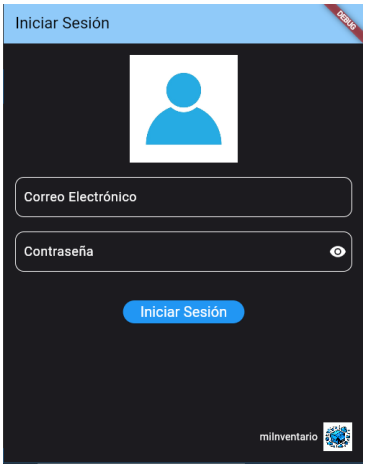
```
  Map<String, int> ordenarPorCantidad(Map<String, int> map) {
    var sortedEntries = map.entries.toList()
      ..sort((a, b) => a.value.compareTo(b.value)); // Ordena por valor

    // Crea un nuevo mapa a partir de las entradas ordenadas
    var sortedMap = <String, int>{};
    for (var entry in sortedEntries) {
      sortedMap[entry.key] = entry.value;
    }
    return sortedMap;
  }

  // ignore: non_constant_identifier_names
  void _EncontrarTop(){
    Map<String, int> conteoProductos = {};

    for (var venta in listaVenta) {
      for (int i = 0; i < venta.productos.length; i++) {
        String nombreProducto = venta.productos[i].nombre;
        int cantidad = venta.cantidades[i];

        if (conteoProductos.containsKey(nombreProducto)) {
          conteoProductos[nombreProducto] = conteoProductos[nombreProducto]!
+ cantidad;
        } else {
          conteoProductos[nombreProducto] = cantidad;
        }
      }
    }

    // Ordenar los productos por cantidad vendida en orden descendente
    conteoProductos = ordenarPorCantidad(conteoProductos);

    if(conteoProductos.length > 5){
      // Crear un nuevo mapa con las claves originales y los primeros 5
valores
      var nuevoConteoProductos = Map.fromEntries(
        conteoProductos.entries.take(5),
      );

      conteoProductos = nuevoConteoProductos;
    }

    // Crear una lista de datos para el gráfico de pastel
```

```
    List<PieChartSectionData> generarSeccionesParaGrafico(Map<String, int>
conteo, Color colorLetra) {
      return conteo.entries.map((entry) {
        return PieChartSectionData(
          color: _regresaColor(), // Asegúrate de definir correctamente la
función _regresaColor()
          value: entry.value.toDouble(), // Convierte la cantidad a double
          title: entry.key, // Usa la clave como título
          radius: 150,
          titleStyle: TextStyle(fontSize: 12, fontWeight: FontWeight.bold,
color: colorLetra),
        );
      }).toList();
    }

    // Llamada a la función con tu mapa conteoProductos
    sectionsChart = generarSeccionesParaGrafico(conteoProductos,
colorLetra);
  }

  @override
  Widget build(BuildContext context) {
    _EncontrarTop();
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.background,
        title: const Text('Productos mas vendidos'),
      ),
      body: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(20.0),
            child: SizedBox(
              width: MediaQuery.of(context).size.width,
              height: 300,
              child: PieChart(
                PieChartData(
                  borderData: FlBorderData(
                    show: false,
                  ),
                  sectionsSpace: 0,
                  centerSpaceRadius: 0,
                  sections: sectionsChart,
                ),
              ),
```

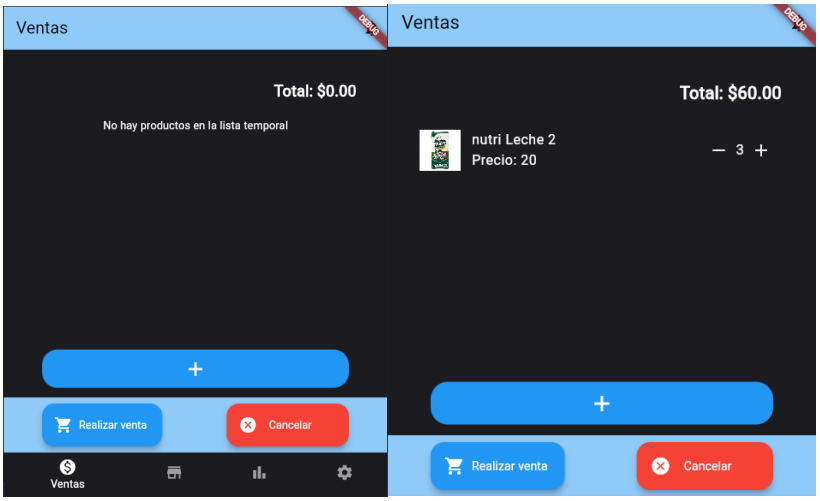
```
            ),
          ),
          Expanded(
            child: SingleChildScrollView(
              child: Column(
                children: [
                  for (int i = 0; i < sectionsChart.length; i++)
                    ListTile(
                      leading: CircleAvatar(backgroundColor:
sectionsChart[i].color),
                      title: Text(
                        sectionsChart[i].title.toString(),
                        style: TextStyle(color: colorLetra), // Cambia el
color del texto
                      ),
                      subtitle: Text(
                        sectionsChart[i].value.toStringAsFixed(0),
                        style: TextStyle(color: colorLetra), // Cambia el
color del texto
                      ),
                    ),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

7. manual de usuario

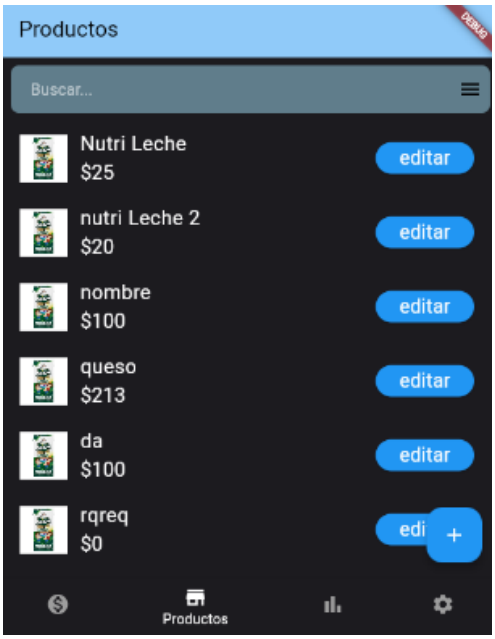
Al entrar a la aplicación deberás iniciar sesión con el usuario y contraseña que se te proporcionaron



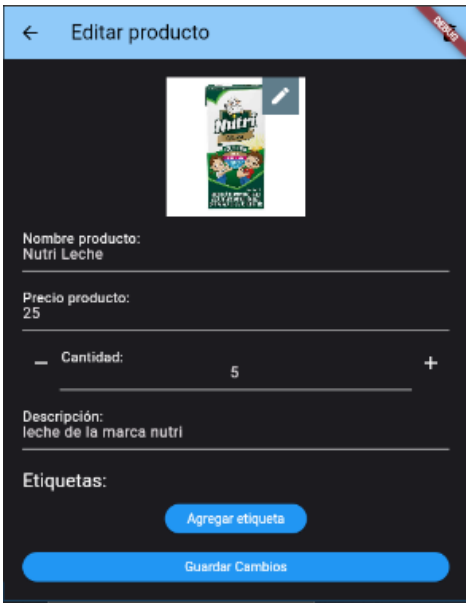
Al entrar en la pantalla ventas tendrás que agregar los productos dan al mas y seleccionado la cantidad y al finalizar a realizar ventas por otro lado si te arrepientes puedes cancelar la venta

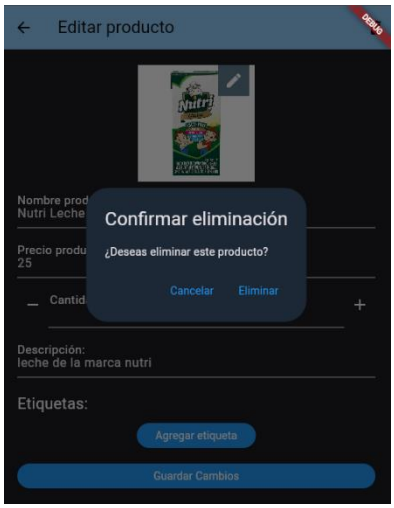


- al entrar en la pantalla productos podremos ver que tenemos una barra de búsqueda que por defecto esta en modo por nombre, pero se puede cambiar por etiquetas dando click alas 3 rayas de la derecha



Al dar click en editar en cualquier producto abre la pantalla editar producto y puedes cambiar cualquier campo de dicho producto o eliminar dando click en el icono de bote de basura de arriba a la derecha

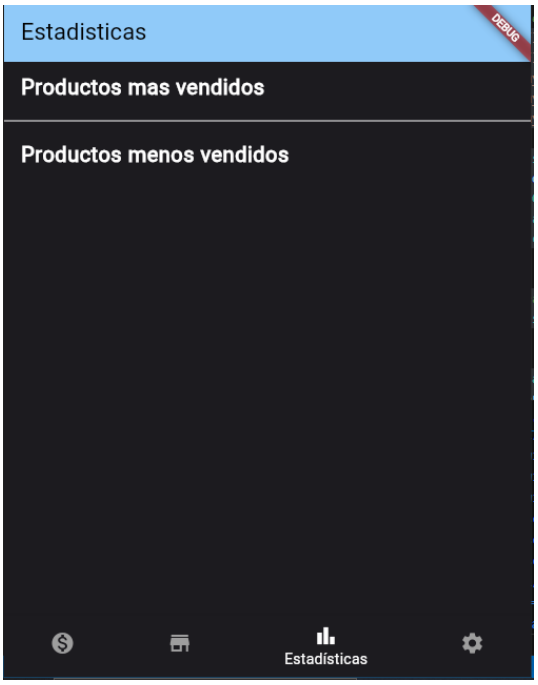




Al dar click en cualquier producto se te mostrara los detalles de este

Al dar click en el icono de “+” abrirá la pantalla agregar producto este te pedirá que llenes los campos y una vez hecho eso podrás agregarlo a la base de datos y productos

Por otro lado en estadísticas como su nombre indica muestra 2 listas de estadísticas al dar click a cualquiera de estas



Por ultimo en configuraciones veras opciones extras como cambiar de modo de vista, ver las ventas realizadas, agregar o editar un nuevo usuario(solo para administrador)