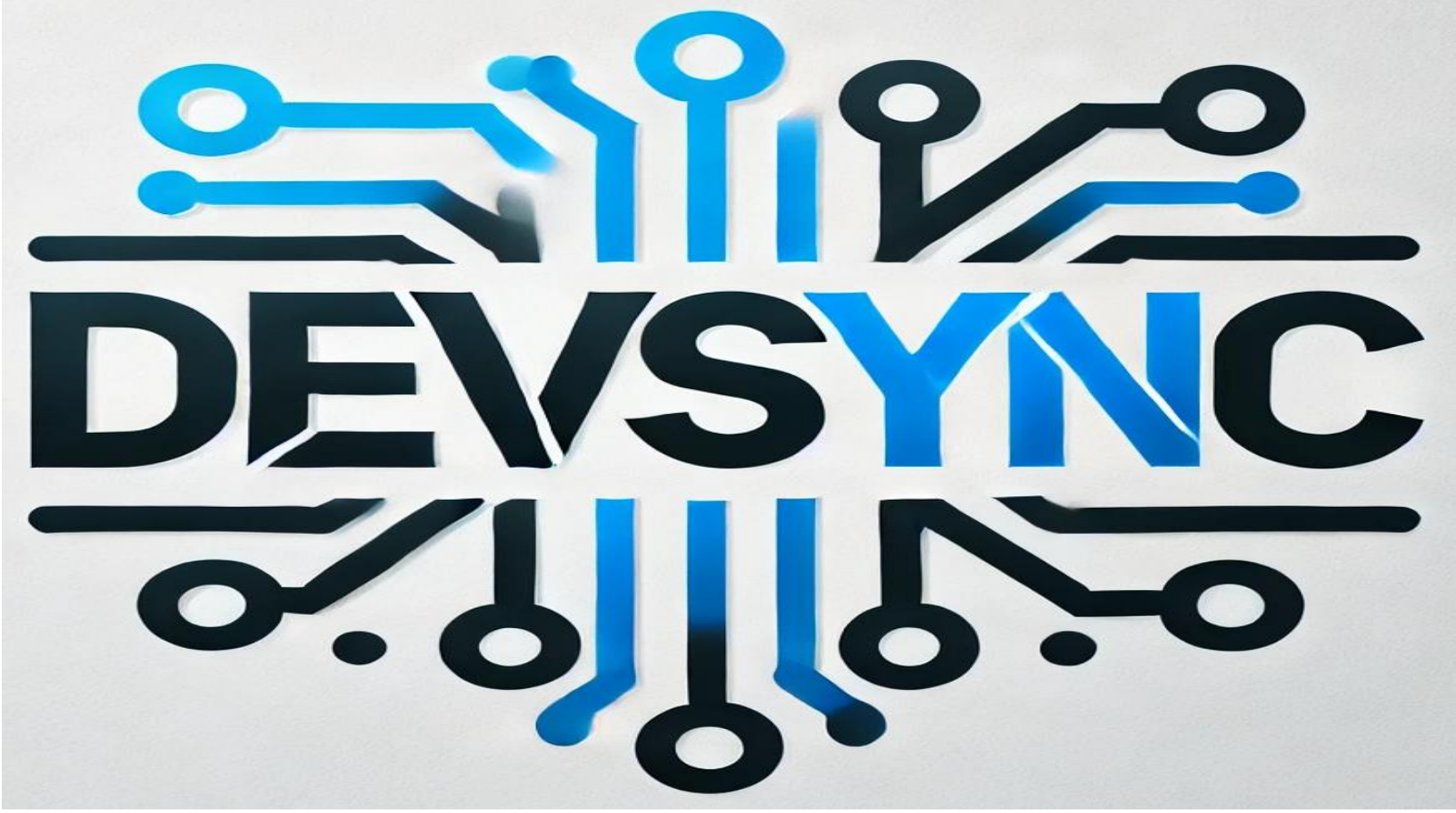

Documento de Arquitectura

DEVSYNC

Documento de Arquitectura

Versión 1

Contenido

Documento de Arquitectura

Versión 1

Introducción

El presente documento de arquitectura de software tiene como objetivo definir, estructurar y comunicar de manera clara y detallada la arquitectura del sistema diseñado para la gestión integral de un bar con tres sedes físicas, ubicadas en Chía, Restrepo y la Zona T de Bogotá.

La arquitectura de software constituye la base sobre la cual se construye y organiza el sistema. Este documento incluye los elementos esenciales que sustentan dicha arquitectura, tales como el **diagrama de despliegue**, que describe la distribución del sistema en los distintos entornos de ejecución; el **diagrama de componentes**, que ilustra la organización lógica del sistema en módulos funcionales; y el **diagrama de arquitectura**, que expone la estructura global del sistema, sus capas y cómo interactúan entre sí.

Asimismo, se presenta el **modelo entidad-relación (E-R)**, el cual define la estructura de la base de datos que soportará el sistema, permitiendo una adecuada organización, integridad y acceso a la información relacionada con usuarios, productos, pedidos, roles y sedes. Este modelo facilita el mantenimiento del sistema y permite su escalabilidad a futuro.

El documento también recoge las **buenas prácticas de desarrollo** adoptadas durante el proyecto, las cuales garantizan la calidad del código, la seguridad del sistema, la facilidad de mantenimiento y la eficiencia en el desarrollo. Entre estas prácticas se destacan la separación de responsabilidades, el uso de patrones de diseño adecuados, el control de versiones, la documentación clara del código y la implementación de pruebas automatizadas.

El sistema será desarrollado bajo la **metodología ágil Scrum**, lo cual permite un enfoque iterativo e incremental, asegurando entregas funcionales en ciclos cortos (Sprints) y fomentando una colaboración constante con el cliente. Desde el Sprint 0, se priorizó el levantamiento de requerimientos, la elaboración de historias de usuario y la creación de prototipos iniciales, con el fin de alinear las expectativas del cliente con la realidad técnica del proyecto.

En resumen, este documento de arquitectura no solo guía al equipo de desarrollo en la construcción del sistema, sino que también sirve como referencia técnica para futuras expansiones, mantenimiento y auditorías. Su estructura refleja el compromiso con la calidad, la transparencia y la eficiencia, pilares fundamentales de una solución tecnológica orientada a transformar digitalmente los procesos de gestión en el sector de bebidas.

Diagrama de Arquitectura

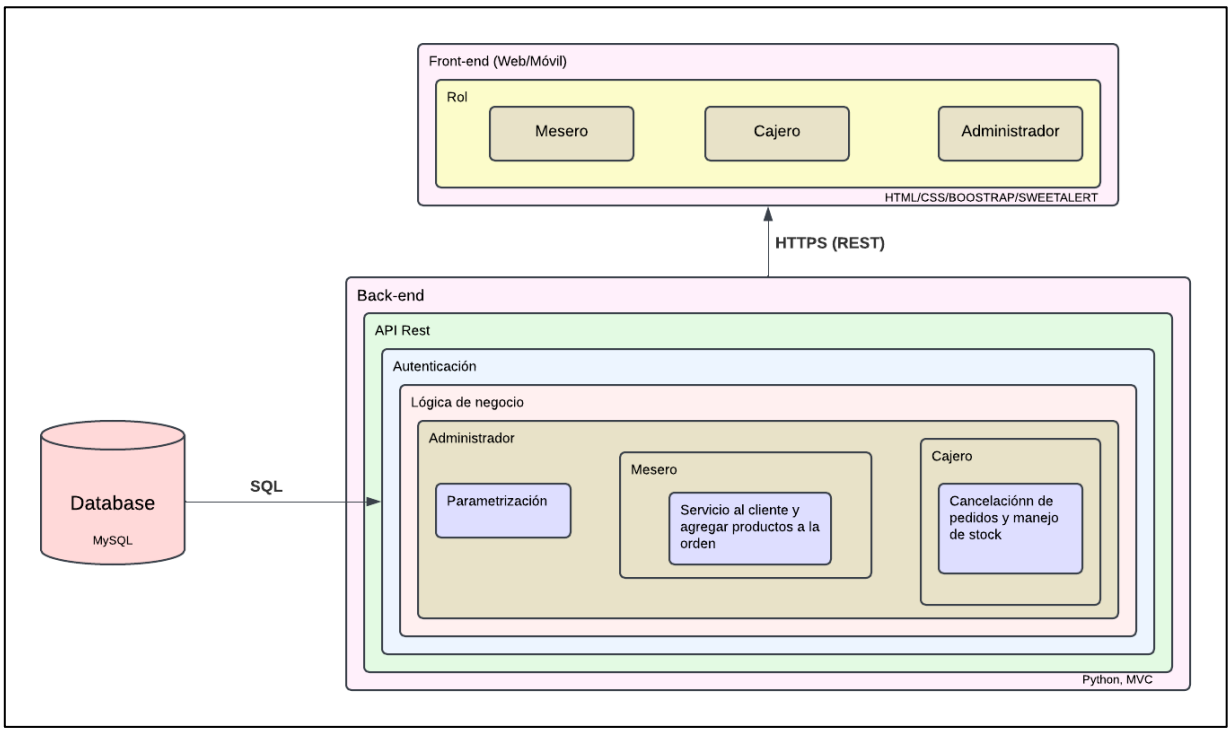

Figura 1. Diagrama de arquitectura de aplicación

Explicación

El diagrama anterior (Fig. 1) representa la estructura de los componentes internos de la aplicación destinada a la gestión del bar; la interacción y la comunicación que hay entre ellos.

Se represento el aplicativo en 3 módulos/capas principales, denominados como **Front-end (Interfaz de usuario), Back-end (lógica de negocio y procesamiento)** y **Database (Almacenamiento de información / datos)**, estos módulos se comunican mediante el protocolo HTTP / HTTPS(REST) para las capas Back y Front, por otro lado, la capa Back y la base de datos se comunican mediante SQL (Esta conexión esta directa a la autenticación de los usuarios en el back).

Explicación de las capas

Front-end:

Es la interfaz del usuario, acoge la capa que se denominó “Rol” que contiene los tres diferentes usuarios que permite el sistema, siendo estos:

- Administrador
- Mesero
- Cajero

Es decir, se representa que cada rol tiene una interfaz acoplada a los permisos, necesidades y funciones que se le asignan como lo dicte el rol.

Esta capa de la aplicación, se construirá mediante HTML, CSS para estructuración y estilo, el framework Bootstrap que ayudará en la cualidad de ser resposive (Adaptable a computadores y celulares) y SweetAlert2 que permitirá mostrar notificaciones / alertas personalizadas a los usuarios.

Se comunica con el Back mediante el protocolo HTTPs (REST), el front-end se encargará de mandar solicitudes al Back como autenticación, gestión de órdenes, visualización de reportes y visualización / adición de stock.

Back-end:

La capa Back se encarga de procesar las peticiones realizadas por la capa front-end, autenticación de usuarios y acceso a la base de datos.

A diferencia de la capa Front-end, esta contiene más módulos contenidos en el debido al volumen de solicitudes y procesos que se requieren ejecutar; estos módulos están explicados y definidos como:

- **API Rest:** Provee endpoints que el front puede consumir, además de manejar solicitudes como GET (consultas), POST (creación), PUT (modificación) y DELETE (eliminación).
- **Autenticación:** Sistema de control de acceso para los usuarios validando las credenciales que permitan el acceso.
- **Lógica de negocio:** Representa la funcionalidad principal del sistema, esta organización se definió por rol, por ende, contiene el rol de administrador con las tareas propias a realizar además de poder ejecutar las acciones realizadas por los otros dos roles; el rol de mesero y cajero contienen dentro de ellas las funciones y permisos que permitan ese rol.

Para la construcción de esta capa se utilizará la tecnología MVC (Modelo – Vista – Controlador) y el lenguaje de programación Python. Esta capa se conecta a la base de datos mediante SQL.

Database:

Se encargará de contener la información y data importante (como la autenticación de usuarios), esta capa se conecta directamente con la capa Back-end, se empleará el gestor de base de datos MySQL.

Diagrama de Despliegue

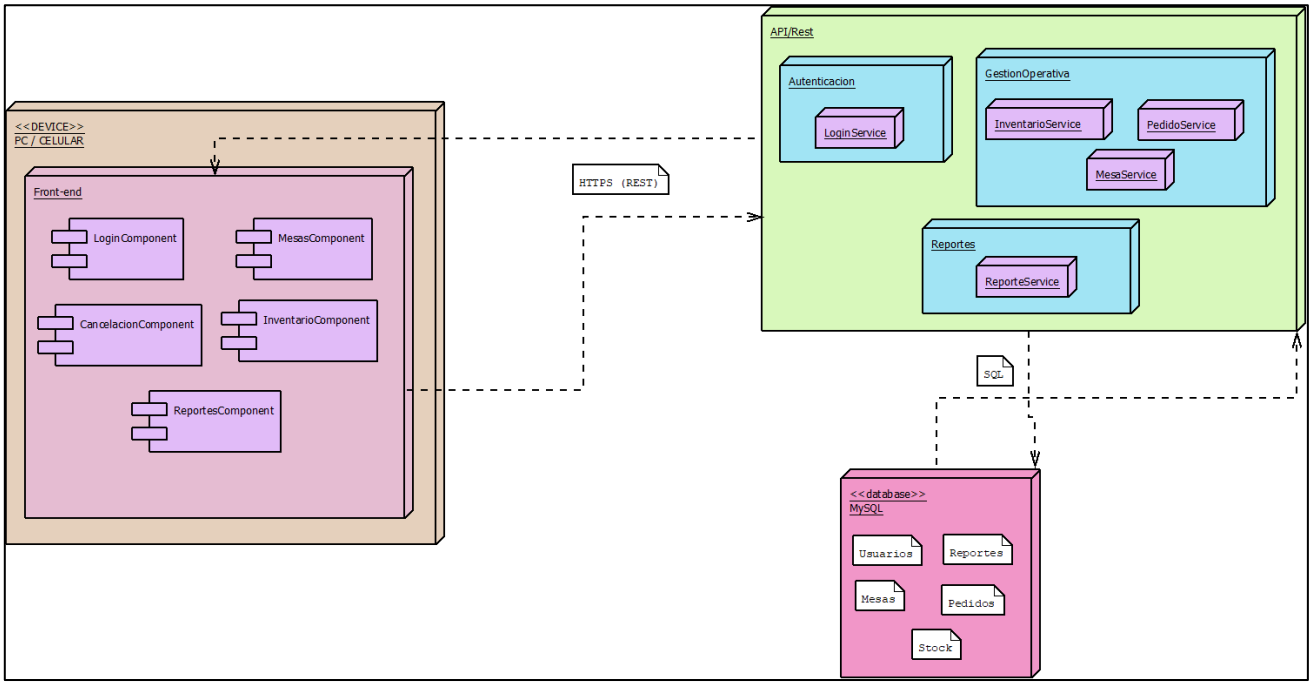

Figura 2. Diagrama de despliegue

Explicación

En la Figura 2, se puede observar el diagrama de despliegue el cual representa la distribución de los componentes del aplicativo en diferentes nodos, siendo así:

El primer nodo <<DEVICE>> representa el nodo físico por el que los usuarios tendrán acceso al sistema, este contiene el nodo front-end (Interfaz) con los diferentes componentes que contribuyen al buen funcionamiento del sistema.

- **LoginComponent:** Acceso
- **MesasComponent:** Estado de las mesas del establecimiento
- **CancelacionComponent:** Se encarga de cancelar los pedidos (pedidos a pagar)
- **InventarioComponent:** Stock del bar
- **ReportesComponent:** Reportes de operación

El nodo front-end se comunica con la API mediante el protocolo HTTPS, este nodo se dividido en tres funcionalidades principales que contienen los servicios que serán consumidos por los usuarios, se nombraron:

- **Autenticacion:** Contiene el login de acceso
- **GestionOperativa:**

Inventario: Permite crear, actualizar y consultar información sobre productos y stock disponible.
Pedido: Administra la creación y actualización de pedidos realizados por los clientes.
Mesa: Controla el estado de ocupación y disponibilidad de las mesas.

- **Reportes:** Contiene los reportes generados

Finalmente, el nodo lógico de base de datos se comunicará mediante SQL facilitando la consulta y realización de transacciones y contiene las entidades principales del aplicativo, indispensables para el buen funcionamiento.

Diagrama de Componentes

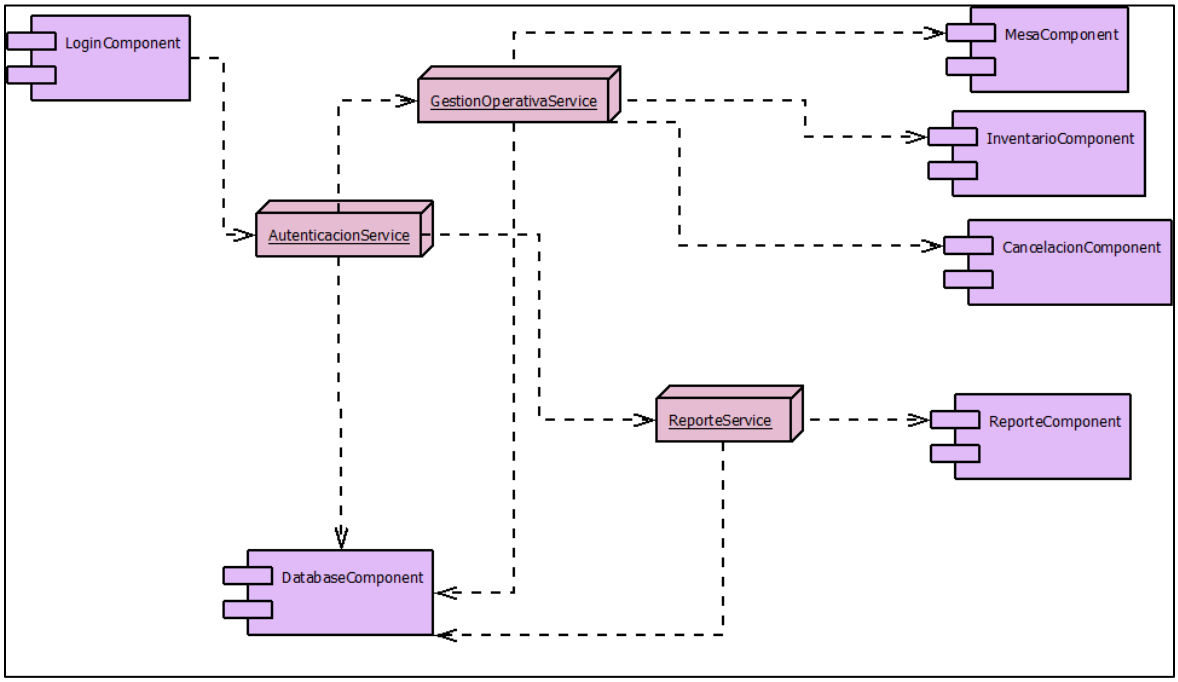

Figura 3. Diagrama de Componentes

Explicación

El diagrama representado en la figura 3, se divide en capas, las cuales son:

- **Capa de Presentación:**

  LoginComponent, MesaComponent, InventarioComponent, CancelacionComponent, ReporteComponent interactúan directamente con el usuario desde la interfaz.

- **Capa de Servicios:**

  AutenticacionService: gestiona el inicio de sesión y la verificación de credenciales.

  GestionOperativaService: centraliza la lógica de negocio relacionada con la operación del restaurante (mesas, inventario, cancelaciones).

  ReporteService: se encarga de generar y entregar reportes administrativos por sede o considerando todas las sedes.

- **Capa de Persistencia:**

  DatabaseComponent: único responsable de acceder y modificar la base de datos, evitando acceso directo desde los componentes.

- **Flujo de dependencias:**

  Los componentes dependen de los servicios, y estos dependen del componente de base de datos.

Modelo Entidad – Relación (E-R)

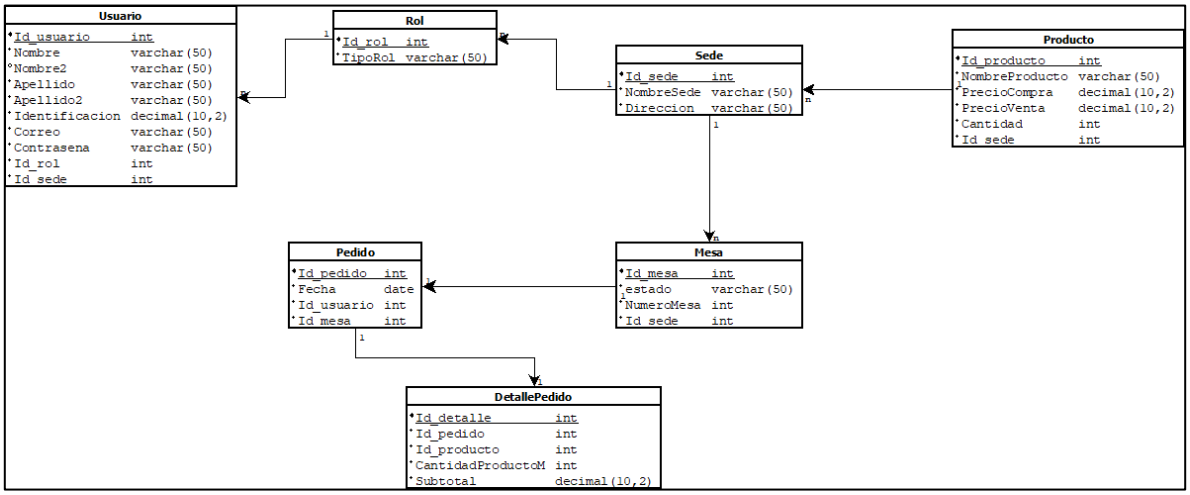

Figura 5. Modelo Entidad – Relación

El modelo entidad relación (Figura 5), representa la estructura de la base de datos que se implementara en el sistema de la gestión del bar, se dividieron entre 7 tablas cada una con sus atributos y relaciones para el correcto funcionamiento de la base, se representaron así:

Explicación

Tabla1→ Usuario: Contiene toda la información de las personas que interactúan con el usuario (meseros, cajeros, administradores).

Para ello, se le pusieron los diferentes atributos, que identifiquen al usuario, como su id único, nombre y apellidos debidamente atomizados (1FN), identificación, correo, contraseña y las relaciones a las tablas sede y rol.

Tabla2 → Rol: Contiene los 3 diferentes roles que se manejaran dentro del sistema, por ende, sus atributos es el id único de identificación y el tipo de rol ya sea mesero, cajero o administrador.

Está relacionado a la tabla usuario para asignar el rol al usuario (Un rol puede ser asignado a muchos usuarios).

Tabla3 → Sede: Representa las sucursales que tiene el bar, siendo Zona T en Bogotá, Chía y Restrepo; sus atributos son el id único, el nombre y la dirección física de la sede.

Se relaciona con las tablas de la siguiente manera, una sede puede tener muchos usuarios (que tiene muchos roles), mesas y productos.

Tabla4 → Producto: Se refiere al stock y la información básica de un producto que el bar tiene para la venta y el valor en que se adquirió.

Sus atributos son, Id_producto, NombreProducto, PrecioCompra, PrecioVenta, Cantidad y Id_sede, el id_sede es relevante debido a que relaciona las tablas por los productos que hay disponibles en cada sede, delimitando la información por sede.

Tabla5 → Mesa: Contiene la información de la disponibilidad de las mesas y el identificador por mesa, sus atributos son el id, el estado, se relaciona con la sede teniendo en cuenta que varias mesas pueden estar en una sede.

Tabla6 → Pedido: Registra la orden o solicitud realizada mediante el mesere a la mesa, sus atributos son identificador único, la fecha del pedido, se relaciona el mesero que tomo la orden y la mesa en la que se encuentran ubicados. Su relación esta dada por un pedido asociado a una mesa.

Tabla7 → DetallePedido: Tabla intermedia entre Pedido y Producto que permite registrar múltiples productos por pedido, con sus cantidades y subtotales.

Sus atributos son, el identificador del detalle, identificador del pedido, del producto y la cantidad de unidades solicitadas en esa orden, así mismo el subtotal de la compra hasta el momento.

Sus relaciones están dadas por:

- Un pedido puede tener múltiples productos.
- Un producto puede estar en múltiples pedidos

Buenas prácticas de Desarrollo

Con base en OWASP (Open Worldwide Application Security Project), se relacionan las siguientes prácticas para el desarrollo seguro y sostenible del sistema.

ID	PRACTICA	DESCRIPCIÓN	APLICACIÓN
01	Gestión segura de contraseñas (Authentication)	Mediante algoritmos hash seguros (bcrypt, Argon2) para las contraseñas.	Evitar guardar la contraseña en texto plano, para la gestión de usuarios; se guarda el hash de la contraseña.
02	Validación de entrada (Input Validation)	Todas las entradas del usuario deben validarse antes de ser procesadas, para evitar inyecciones y errores.	Validar campos como nombres, correos, cantidades, etc., tanto en el frontend como en el backend.
03	Prevención de inyecciones SQL (SQL Injection)	Utiliza sentencias preparadas y consultas parametrizadas para evitar ataques por inyección SQL.	odas las consultas que interactúan con la base de datos deben usar prepared statements
04	Control de acceso (Authorization)	Los usuarios sólo deben acceder a recursos permitidos según su rol (administrador, mesero, etc.).	El campo Id_rol en Usuario debe usarse para restringir acciones dependiendo del tipo de usuario.
05	Gestión de sesiones (Session Management)	Manejar correctamente las sesiones: expiración, cierre de sesión, regeneración de ID tras login.	Al autenticarse, se debe iniciar una sesión segura que expire después de cierto tiempo de inactividad.
06	Gestión de errores (Error Handling)	No mostrar mensajes de error detallados al usuario. Los errores deben registrarse de forma segura para revisión.	Si hay un error al registrar un pedido, debe registrarse internamente, pero no exponerse al cliente.
07	Seguridad en APIs REST (REST Security)	Las APIs deben requerir autenticación, validación de entrada, y usar HTTPS.	Si el sistema expone endpoints REST, deben protegerse con tokens y HTTPS.
08	Manejo seguro de archivos (File Upload)	Al subir archivos al sistema, limitar el tamaño, tipo y nombre.	En caso de agregar módulo de imágenes de productos, restringir extensiones y validar contenido.
09	Protección contra ataques de fuerza bruta (Authentication)	Implementar límites de intentos de login, captchas, o tiempo de espera entre intentos.	Evitar que usuarios prueben contraseñas múltiples veces sin restricción.
10	Uso de HTTPS (Transport Layer Protection)	Toda comunicación debe hacerse usando HTTPS para proteger la confidencialidad de los datos.	Asegurar que el sistema esté configurado con HTTPS en todas sus rutas, especialmente para login y formularios.