

Documentación General del Proyecto – CaffeNet

Nombre del Proyecto

CaffeNet – Plataforma Integral para la Gestión de Cafeterías: Productos, Reservas y Pagos

Planteamiento del Problema

Hoy en día muchas cafeterías tienen problemas porque manejan todo a mano como los productos, el control del inventario, las reservas de las mesas y los pagos, así que al no tener un sistema que haga todo junto se cometen errores como que no se actualizan bien los productos, se duplican o mal asignan las reservas, es difícil controlar a las personas que usan el lugar y organizar los pagos. Además no hay roles claros para los usuarios y los administradores lo que hace que el sistema sea menos seguro y menos eficiente.

Objetivos del Sistema

Objetivo General

Crear un sistema modular y seguro que permita manejar al mismo tiempo y de forma independiente todas las operaciones de la cafetería, esto incluye administrar productos, reservas, usuarios y pagos, usando interfaces diferentes para administradores, usuarios y visitantes.

Objetivos Específicos

- Ayudar a los usuarios a ver los productos, hacer pedidos y reservar mesas con validaciones que eviten errores y con cálculo automático del costo.
- Dar a los administradores herramientas completas para manejar productos, mesas, usuarios y pagos, con control en tiempo real del stock y las reservas.
- Mantener reglas claras de seguridad y separación de roles para proteger la información y la privacidad.
- Crear una estructura del sistema que sea fácil de mejorar y mantener en el futuro.
- Ofrecer una experiencia fácil y accesible para usuarios registrados y visitantes.

Requerimientos del Sistema

Requerimientos Funcionales

Para el Administrador

- Inicio de sesión como administrador.
- Gestión total de productos: agregar, visualizar, editar, eliminar con actualización automática del stock basado en pedidos.
- Administración completa de mesas: creación, edición, eliminación, liberación y visualización con atributos detallados (número, capacidad, ubicación, precio por hora, estado).
- Visualización y manejo de reservas en curso, con la capacidad de validar y gestionar solicitudes.
- Gestión de usuarios registrados: listado y mediante un recuperar contraseña puede que el usuario realice edición de contraseña.

- Módulos de pagos diferenciados para productos y reservas, con filtros por correo o código, selección de métodos de pago y acceso al historial de facturas y comprobantes generados solamente si el Android es 11 o mayor.
- Edición y visualización de su propio perfil y soporte para recuperación de contraseña mediante correo.

Para el Usuario

- Registro e inicio de sesión en una interfaz dedicada.
- Visualización completa del catálogo de productos con detalles de nombre, descripción, precio y stock.
- Gestión de carrito de compras:
 - Añadir productos al carrito solo si hay stock disponible.
 - Visualizar los productos en el carrito con sus cantidades y precios.
 - Modificar la cantidad o eliminar productos dentro del carrito.
 - Realización de pedidos desde el carrito, que automáticamente actualizan el stock en el sistema.
- Reserva de mesas con validaciones en tiempo real (capacidad y disponibilidad), uso de código único y cálculo automático del costo total según duración y tarifa.
- Gestión y actualización de su perfil personal (nombre, teléfono, contraseña).
- Flujo seguro y sencillo para recuperación de contraseña.
- Restricción clara para que no acceda a funciones administrativas ni gestión de pagos.

Para Visitantes (No registrados)

- Continuar como Invitado.
- Visualización completa del catálogo de productos publicados con nombre, descripción, precio y stock.
- Acceso sin posibilidad de añadir productos al carrito ni realizar compras o reservas.
- Interfaz simplificada sin opciones de interacción comercial.

Requerimientos No Funcionales

- Seguridad: Implementación de mecanismos seguros para la autenticación y manejo de sesiones, incluyendo recuperación segura de contraseñas y control de acceso.
- Interfaces separadas: Diseño de interfaces específicas y adaptadas para cada tipo de usuario (administrador, usuario registrado y visitante), garantizando una experiencia de usuario adecuada y segura.
- Base de datos relacional MySQL: Uso de MySQL para asegurar integridad referencial, normalización adecuada y relaciones claras entre las entidades del sistema.
- Arquitectura modular y escalable: Estructura del sistema basada en módulos independientes que facilitan la escalabilidad, mantenimiento y evolución del proyecto.
- Compatibilidad móvil: La aplicación debe ser responsive y funcionar correctamente en dispositivos móviles Android, asegurando accesibilidad y usabilidad.
- Rendimiento óptimo: Optimización en las operaciones del sistema para garantizar tiempos de respuesta rápidos y fluidez en la experiencia de usuario, incluso con un gran volumen de datos.

Estructura Técnica – CaffeNet

Backend – Spring Boot

Arquitectura modular y reutilizable, basada en Controladores, Servicios y Repositorios, orientada a entidades con auditoría y borrado lógico.

Estructura de Paquetes

```
com.corhuila.Backend_CaffeNet
├── common.base
│   ├── ABaseEntity.java           # Entidad base con auditoría
│   ├── IBaseRepository.java       # Interfaz base para repositorios
│   ├── IBaseService.java          # Interfaz base para servicios
│   ├── ABaseService.java          # Lógica común de servicios
│   └── ABaseController.java       # Controlador REST base
├── common.Dto
│   └── ApiResponseDto.java        # Respuesta estándar para endpoints
├── modules
│   └── admin
│       ├── Controller/
│       │   └── AdminRestController.java
│       ├── Entity/
│       │   └── Admin.java
│       ├── Service/
│       │   └── AdminService.java
│       ├── IRepository/
│       │   └── IAdminRepository.java
│       └── request/
│           └── LoginAdminRequest.java
```

Funcionalidades Clave

ABaseEntity: Heredable. Incluye status, timestamps, y auditoría (createdBy, etc).

ABaseService: Centraliza lógica de:

save(), update(), delete() (lógico), findById(), findByStateTrue().

ABaseController: CRUD REST estándar, entrega ApiResponseDto.

Endpoints (Admin Module)

```
GET /api/admin  Listar administradores
POST /api/addAdmin  Crear nuevo admin (email fijo admin@gmail.com)
POST /api/loginAdmin  Login de administrador
GET /api/getAdmin/{email}  Buscar admin por email
GET /api/checkEmailAdmin  Validar si email ya existe
PUT /api/admin/{email}  Actualizar datos de un admin
```

Frontend – Ionic + Angular

Aplicación híbrida modular, orientada a roles (admin, guest, user), con separación lógica en core, common, modules, y shared.

Estructura del Proyecto

```
src/
├── app/
│   ├── common/
│   │   ├── guards/           # Protección de rutas (AuthGuard, etc.)
│   │   ├── interfaces/      # Interfaces TypeScript compartidas
│   │   └── services/         # Servicios globales (API, Token, etc.)
│   ├── core/
│   │   └── home/             # Página principal de bienvenida
│   ├── modules/
│   │   ├── admin/           # Funciones del administrador
│   │   ├── guest/           # Funciones para invitados
│   │   └── user/             # Funciones para usuarios registrados
│   ├── reservations/        # Módulo de reservas
│   ├── shared/components/    # Componentes reutilizables (cards, menús)
│   ├── api.config.ts         # Configuración base de URLs
│   ├── app.component.*       # Componente raíz
│   └── app.routes.ts         # Rutas principales
├── assets/
│   └── icon/
├── environments/
│   ├── environment.ts        # Dev
│   └── environment.prod.ts   # Producción
```

Rutas Principales

Rutas

- /login Login de usuario
- /register Registro de nuevo usuario
- /home Inicio de la app
- /mainmenu Menú del usuario autenticado
- /mainmenu-guest Menú para invitados
- /productos Listado de productos
- /cart Carrito de compras
- /profile Perfil del usuario
- /reserva/crear Crear nueva reserva
- /pedido/detalle/:id Detalle de pedido específico
- /admin Vista general para administrador

Arquitectura Técnica

- ApiService (api.service.ts): Comunicación centralizada HTTP con backend.
- TokenService (token.service.ts): Manejo de tokens en local storage.
- Guards: Protección de rutas (por sesión o rol).
- Modularización: Cada módulo encapsula sus vistas y lógica.

PDF:

- Generación: @cesarbr/pdf-generator
- Apertura: @capawesome-team/capacitor-file-opener

Configuración:

```
npm install @cesarbr/pdf-generator
npm install @capawesome-team/capacitor-file-opener
npx cap sync
```

Comunicación Frontend–Backend

La comunicación entre el frontend (construido en Angular/Ionic) y el backend (Spring Boot) se establece exclusivamente mediante solicitudes HTTP a través de una API RESTful. Todo el flujo está diseñado para ser seguro, estructurado y reutilizable.

Gestión Centralizada de Endpoints

- Todas las solicitudes al backend se manejan de forma centralizada mediante un servicio llamado ApiService en Angular.
- Este servicio encapsula la lógica de peticiones HTTP (GET, POST, PUT, DELETE) y actúa como punto único para:

Agregar cabeceras comunes (como tokens de autorización).

Manejar errores globales (una respuesta con error 401, 403, 500, etc.).

Loggear solicitudes o respuestas para debugging en desarrollo.

- Formato de Respuesta Unificado con ApiResponseDto Cada endpoint en el backend responde utilizando una estructura estándar definida por un DTO (Data Transfer Object) llamado ApiResponseDto.

Esto permite que el frontend procese las respuestas de manera predecible, ya que todos los resultados contienen campos comunes como:

- mensaje (string): mensaje informativo o de error.
- data (any): contenido de la respuesta (ej. lista de productos, reserva, usuario).

- estado (boolean o código): indica éxito o fallo de la operación.

Ejemplo de una respuesta típica:

```
{
  "mensaje": "Reserva creada correctamente",
  "estado": true,
  "data": {
    "codigoReserva": "ABC123",
    "fecha": "2025-06-01"
  }
}
```

Autenticación y Manejo del Token

Una vez implementado el módulo de seguridad (Spring Security con JWT), el backend generará un token JWT al momento de iniciar sesión exitosamente.

Este token se almacenará de forma segura en el SessionStorage del navegador para su uso temporal (se borra al cerrar la sesión o el navegador).

El token se incluirá automáticamente en los headers de cada solicitud HTTP como:

```
Authorization: Bearer <token>
Esto permite que el backend valide el acceso en cada endpoint según el rol y permisos del usuario autenticado.
```

Ventajas de este enfoque Mantenibilidad: Al tener un punto único (ApiService), es fácil modificar cómo se hacen las solicitudes sin tocar múltiples componentes.

Reusabilidad: Cualquier componente o módulo puede consumir la API sin duplicar código.

Seguridad: El uso de tokens JWT y headers Authorization permite implementar protección basada en roles y sesiones temporales.

Escalabilidad: El formato unificado de respuesta y la separación entre cliente y servidor permiten extender el sistema fácilmente (por ejemplo, una app móvil nativa puede consumir la misma API).

Base de datos

Entidades y Atributos

1. usuario

Representa a los usuarios del sistema (clientes, empleados u otros).

- **email:** PK, correo electrónico del usuario (identificador único).
- **password:** contraseña para acceder al sistema.

- **full_name**: nombre completo del usuario.
 - **telefono**: número de contacto del usuario.
-

2. admin

Representa a los administradores del sistema, con permisos especiales.

- **email**: **PK**, correo electrónico del administrador (único).
- **password**: contraseña del administrador.
- **full_name**: nombre completo del administrador.

3. reserva

Representa una reserva hecha por un usuario para ocupar una mesa.

- **id**: **PK**, identificador único de la reserva.
 - **fechaInicio**: fecha en que comienza la reserva.
 - **fechaFin**: fecha en que finaliza la reserva.
 - **numero_personas**: número de personas en la reserva.
 - **estado**: estado de la reserva (activa, cancelada, completada).
 - **codigo**: código único asociado a la reserva.
 - **precio**: monto total estimado de la reserva.
 - **mesa_id**: **FK**, referencia a la mesa reservada.
 - **usuario_email**: **FK**, referencia al usuario que realiza la reserva.
-

4. mesa

Representa las mesas disponibles en el establecimiento.

- **numero**: **PK**, número identificador de la mesa.
 - **capacidad**: número de personas que puede acomodar.
 - **ubicacion**: ubicación de la mesa dentro del local.
 - **estado**: disponibilidad de la mesa (libre, ocupada, reservada).
 - **precio**: costo de la reserva de la mesa.
-

5. PagoReserva

Representa el pago realizado para una reserva.

- **id**: **PK**, identificador del pago.
 - **metodo_pago**: método usado (efectivo, tarjeta, etc.).
 - **monto**: valor pagado.
 - **usuario_email**: **FK**, referencia al usuario que paga.
 - **reserva_id**: **FK**, referencia a la reserva asociada.
-

6. comprobante_reserva

Representa el comprobante generado por el pago de una reserva.

- **id:** PK, identificador del comprobante.
 - **fecha_emision:** fecha en la que se generó el comprobante.
 - **reserva_id:** FK, reserva relacionada.
 - **usuario_email:** FK, usuario asociado.
 - **pago_reserva_id:** FK, pago asociado.
-

7. producto

Representa los productos ofrecidos (alimentos, bebidas, etc.).

- **id:** PK, identificador del producto.
 - **nombre:** nombre del producto.
 - **descripcion:** descripción del producto.
 - **precio:** precio unitario del producto.
 - **estado:** disponibilidad del producto (activo/inactivo).
 - **stock:** cantidad disponible en inventario.
 - **url:** imagen o enlace relacionado con el producto.
-

8. detalle_pedido

Contiene los productos solicitados en un pedido.

- **id:** PK, identificador del detalle de pedido.
 - **fecha_emision:** fecha del pedido.
 - **carbuy_id:** FK, carrito asociado.
 - **producto_id:** FK, producto solicitado.
 - **usuario_email:** FK, usuario que realizó el pedido.
-

9. pago

Representa un pago por un pedido.

- **id:** PK, identificador del pago.
 - **detalle_pedido_id:** FK, detalle del pedido pagado.
 - **usuario_email:** FK, usuario que realizó el pago.
 - **monto:** cantidad pagada.
 - **metodo_pago:** forma en que se realizó el pago.
-

10. comprobante

Representa el comprobante generado por un pago de pedido.

- **id:** PK, identificador del comprobante.
- **detalle_pedido_id:** FK, pedido al que pertenece.
- **pago_id:** FK, pago relacionado.

- **fecha_emision:** fecha en que se generó el comprobante.
 - **usuario_email:** FK, usuario asociado.
-

11. carbuy

Representa un carrito de compras temporal antes de generar el pedido.

- **id:** PK, identificador del carrito.
- **estado:** estado del carrito (activo, cerrado).
- **total:** total del carrito.
- **cantidad:** número de productos.
- **producto_id:** FK, producto en el carrito.

Arquitectura del Proyecto

- Arquitectura **Modular** basada en MVC (Modelo-Vista-Controlador).
- Separación clara de módulos funcionales: productos, usuarios, reservas, pagos, entre otras.
- Diferenciación de rutas y componentes en frontend según roles.
- Controladores backend que filtran permisos y validan acciones.
- Servicios y repositorios para encapsular la lógica y acceso a datos.
- Manejo de errores y excepciones para asegurar estabilidad y trazabilidad.
- Diagramas de flujo para procesos clave como reserva, pago y gestión de stock.

Manual de Usuario

Para Usuarios Registrados

- **Acceso:** Registro e inicio de sesión mediante interfaz dedicada, con opción "¿Olvidaste tu contraseña?" que permite al usuario solicitar una contraseña temporal vía correo electrónico y luego cambiarla por una nueva segura.

The image displays two mobile application screens side-by-side. The left screen, titled 'Registro' (Registration), has a subtitle 'Datos personales' (Personal data) and contains four input fields: 'Nombre completo' (Full name), 'Correo Electrónico' (Email), 'Teléfono' (Phone), and 'Contraseña' (Password). Below these fields is a red 'Registrarse' (Register) button and a link '¿Ya tienes cuenta? [Inicia Sesión](#)' (Do you already have an account? [Log in](#)). The right screen, titled 'Ingreso' (Login), contains two input fields: 'Correo:' (Email) and 'Contraseña:' (Password). Below these is a red 'Iniciar sesión' (Log in) button. Further down are two links: '¿Aún no tienes cuenta? [Regístrate](#)' (Don't you have an account yet? [Register](#)) and '¿Olvidaste tu contraseña?' (Forgot your password?). At the bottom, there is a horizontal separator line followed by two links: 'Continuar como [Invitado](#)' (Continue as [Guest](#)) and 'Continuar como [Administrador](#)' (Continue as [Administrator](#)).

- **Productos:** Visualización del catálogo con información completa, con la opción de añadir productos al carrito solo si hay stock disponible en el menú principal.



- **Carrito de Compras:**

- Visualizar los productos agregados al carrito con cantidad, precio individual y subtotal.
- Modificar cantidades o eliminar productos dentro del carrito en cualquier momento.
- El sistema impide añadir productos sin stock suficiente.

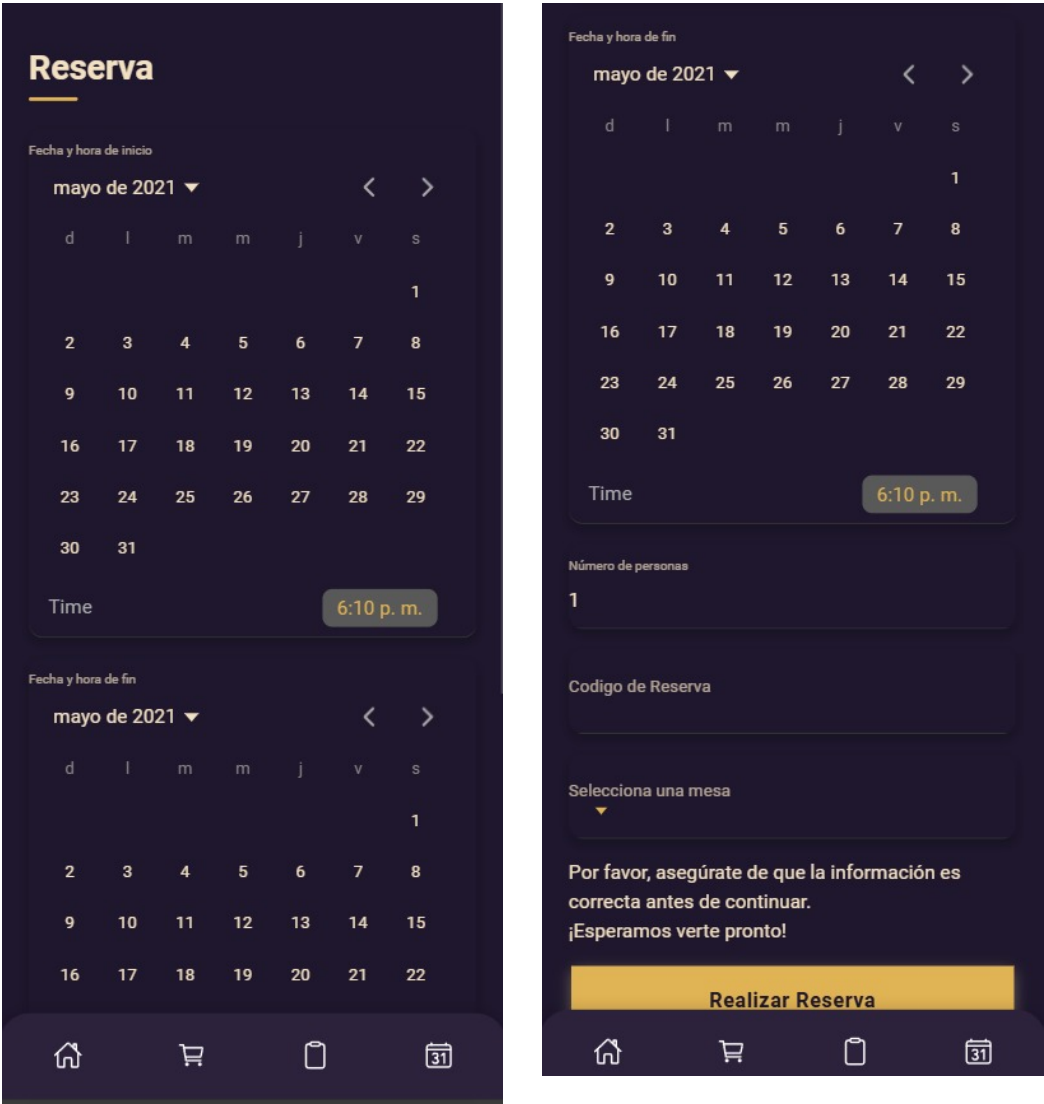


- **Pedidos:**

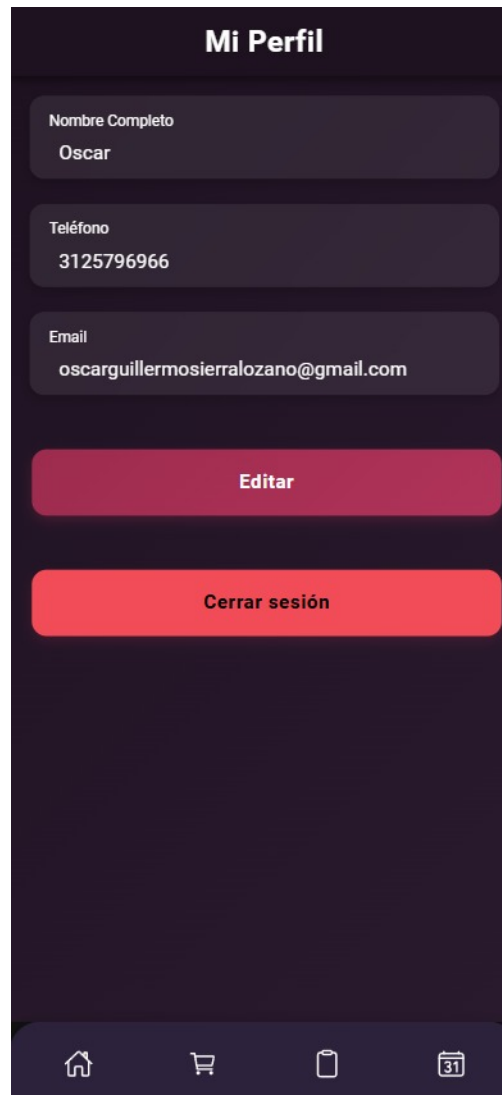
- Genera el pedido desde el carrito.
- El stock se actualiza automáticamente tras cada pedido.
- Recepción de confirmación del pedido.



- **Reservas:** Selección de fecha, código único ingresado por el usuario, número de personas y mesa disponible, con validaciones en tiempo real.

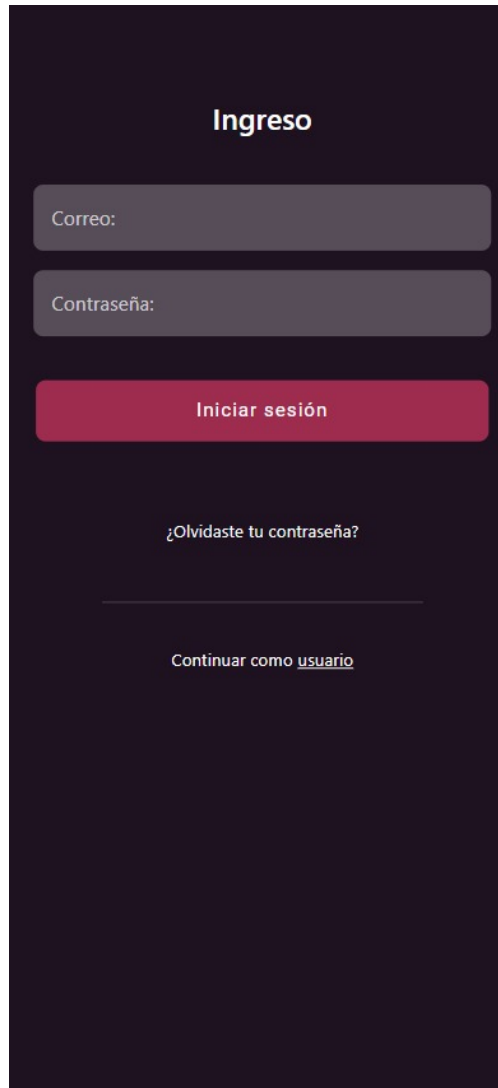


- **Perfil:** Edición de datos personales y contraseña.



Para Administradores

- **Acceso:** Inicio de sesión exclusivo y recuperación de contraseña por correo.

A login form with a dark purple background. At the top, the word 'Ingreso' is centered in white. Below it are two light gray input fields: the first is labeled 'Correo:' and the second is labeled 'Contraseña:'. A red button with the text 'Iniciar sesión' is positioned below the password field. Further down, the text '¿Olvidaste tu contraseña?' is centered, followed by a horizontal line. At the bottom, the text 'Continuar como [usuario](#)' is centered, with 'usuario' being a blue link.

Ingreso

Correo:

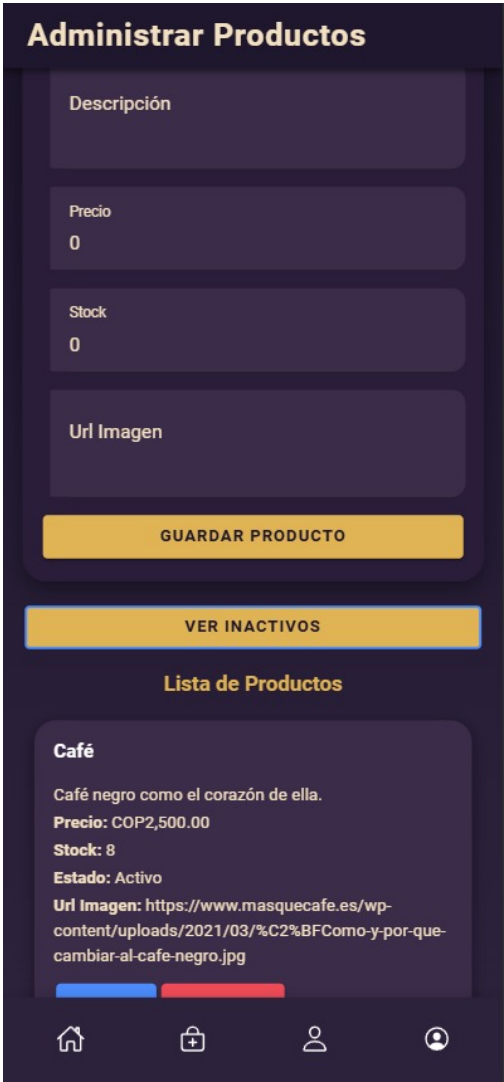
Contraseña:

Iniciar sesión

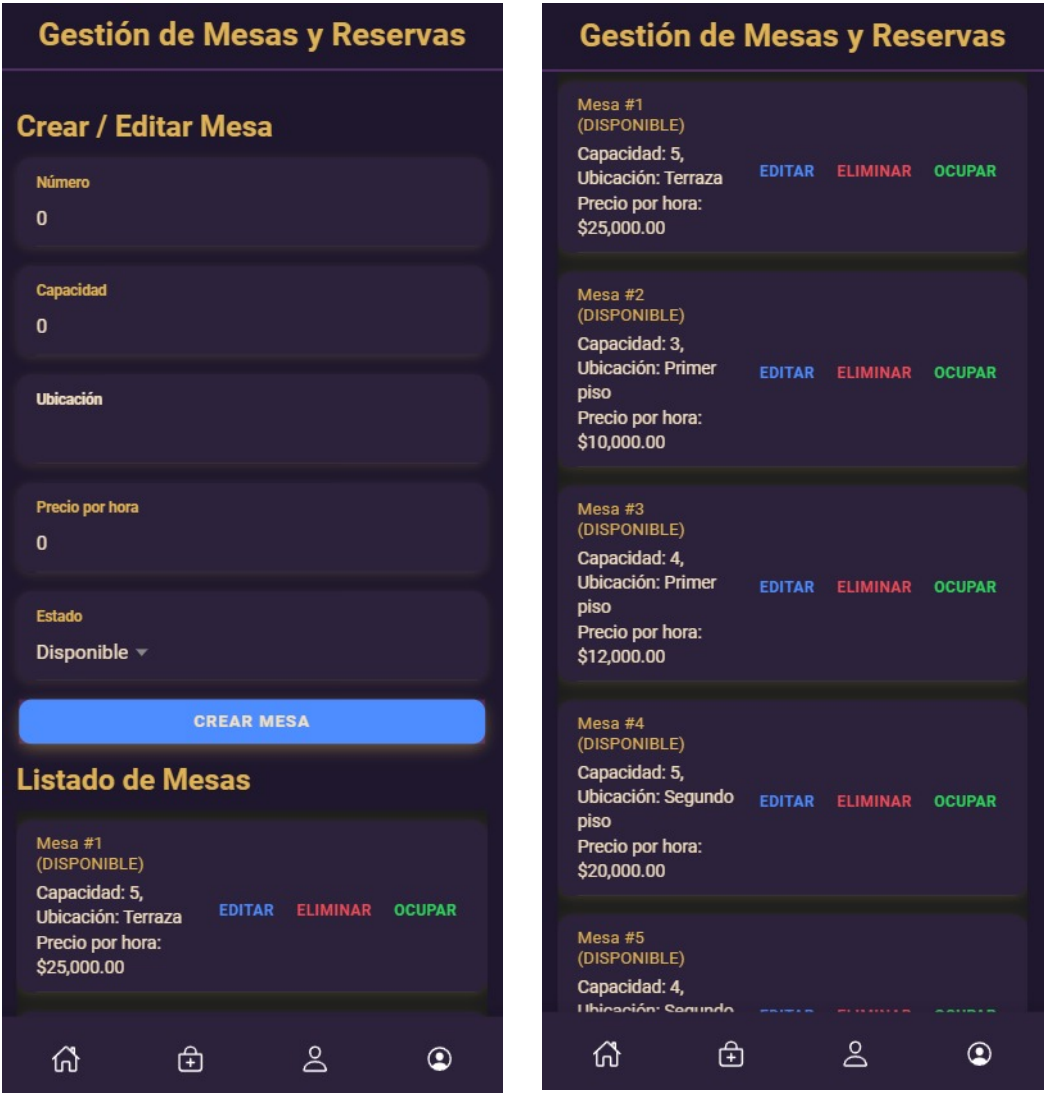
¿Olvidaste tu contraseña?

Continuar como [usuario](#)

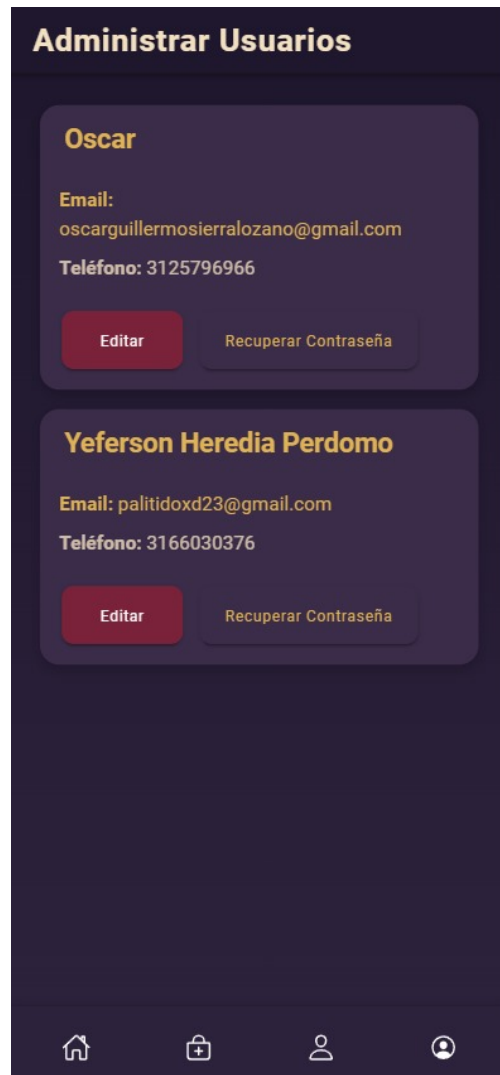
- **Productos:** CRUD completo con control automático de stock.



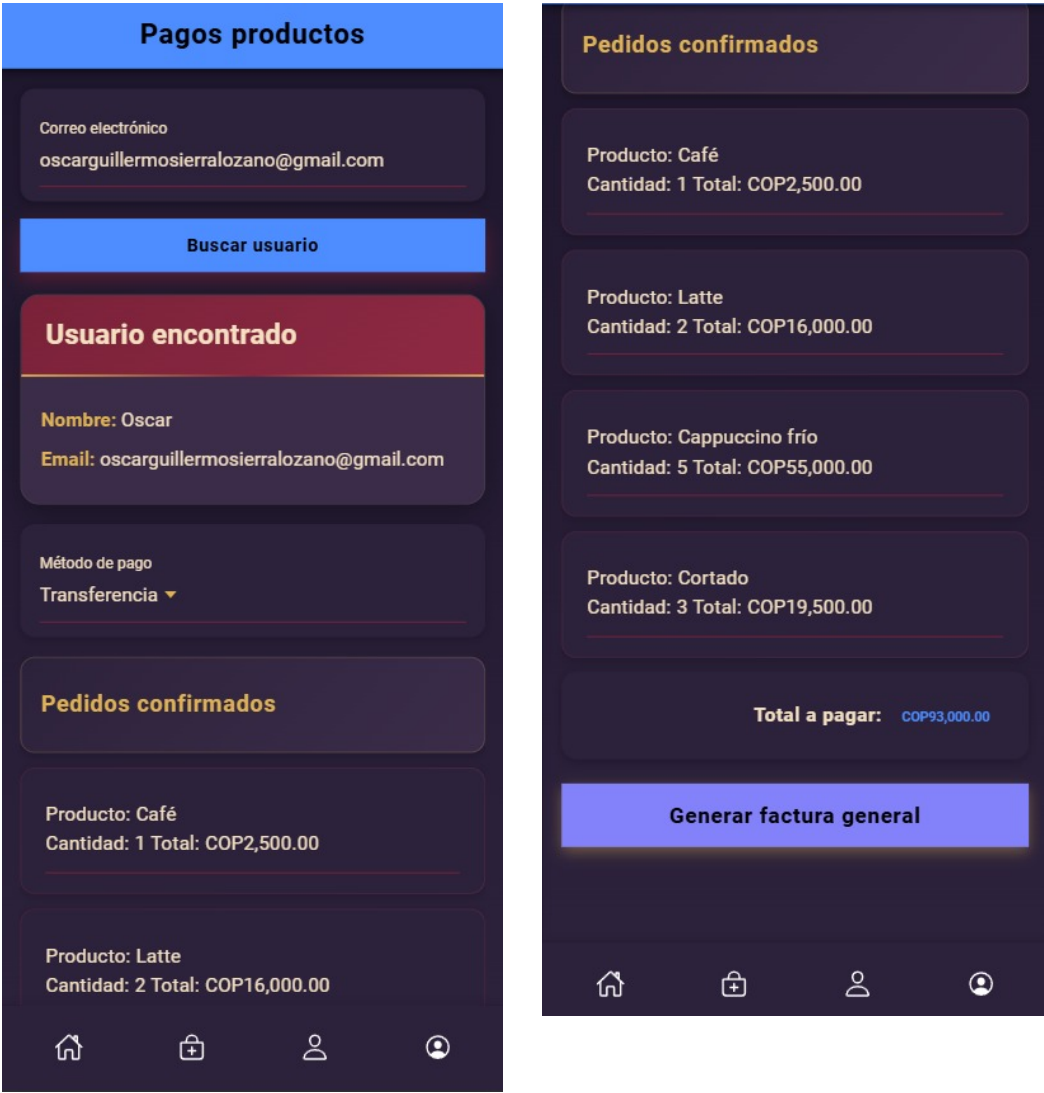
- **Mesas y Reservas:** Gestión integral.



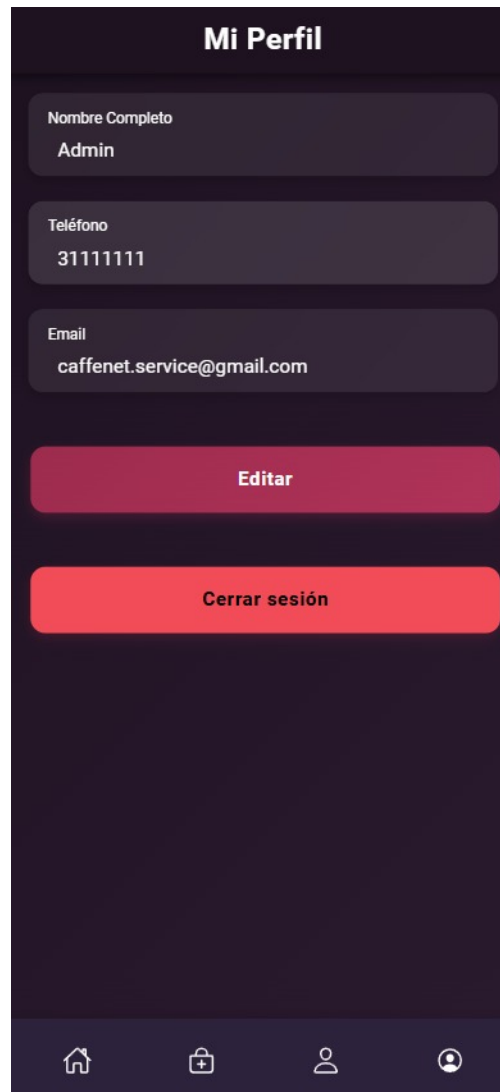
- **Usuarios:** Gestión y recuperación de credenciales.



- **Pagos:** Control de productos, reservas y comprobantes.

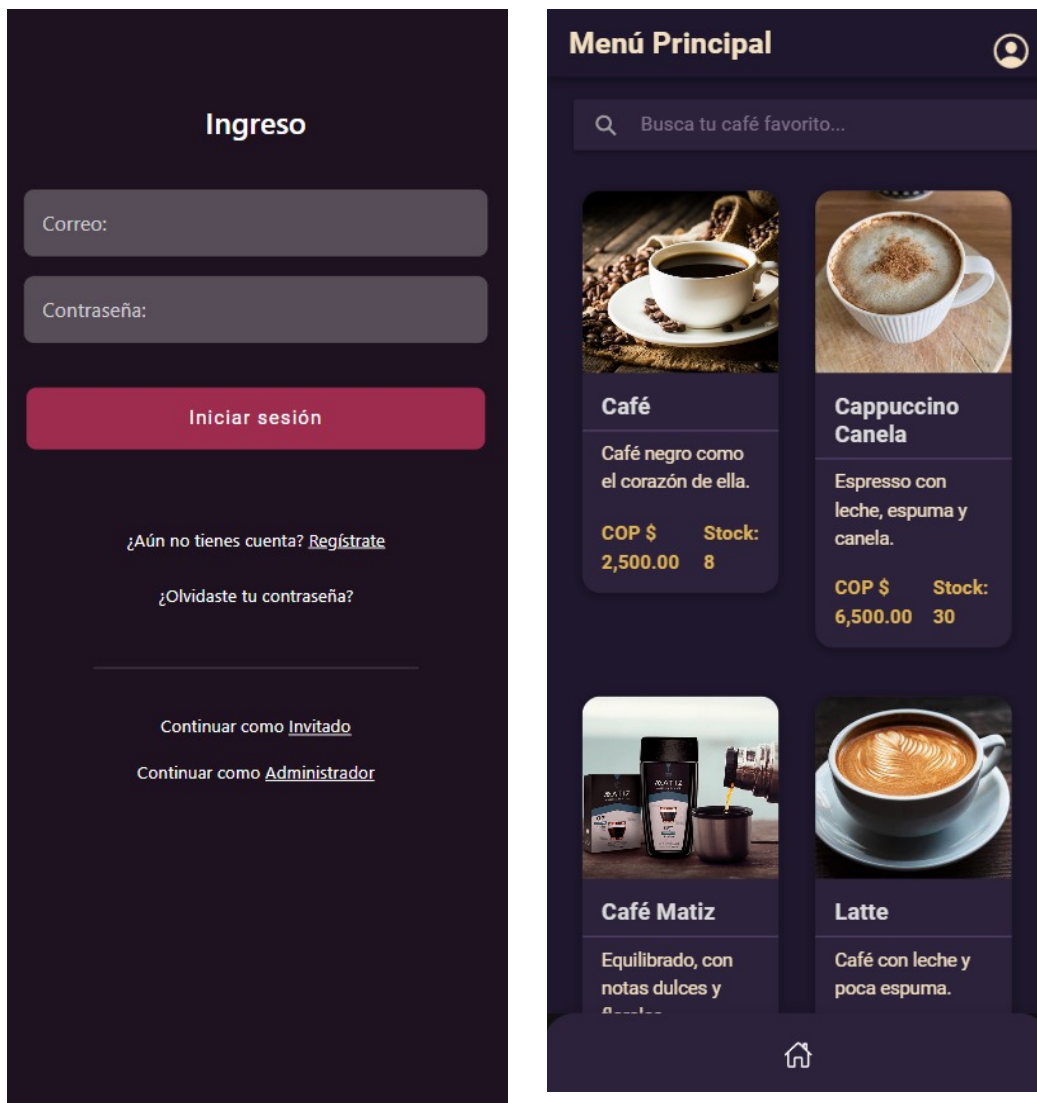


- **Perfil Admin:** Edición de perfil y contraseña.



Para Visitantes

- **Acceso como Invitado** sin necesidad de registrarse.
- **Visualización restringida** del catálogo (nombre, descripción, precio y stock).

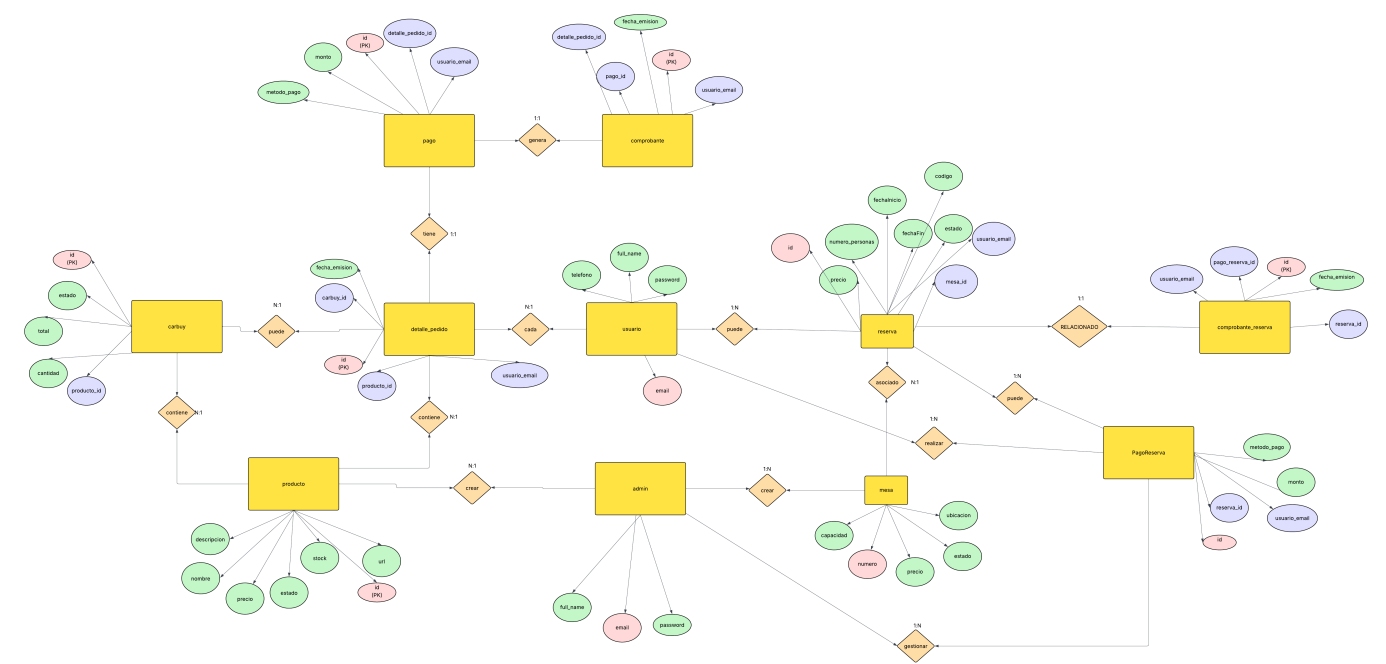


- **Restricciones:** Sin acceso a compras, reservas ni funciones personales.

Modelo Entidad-Relación (MER)

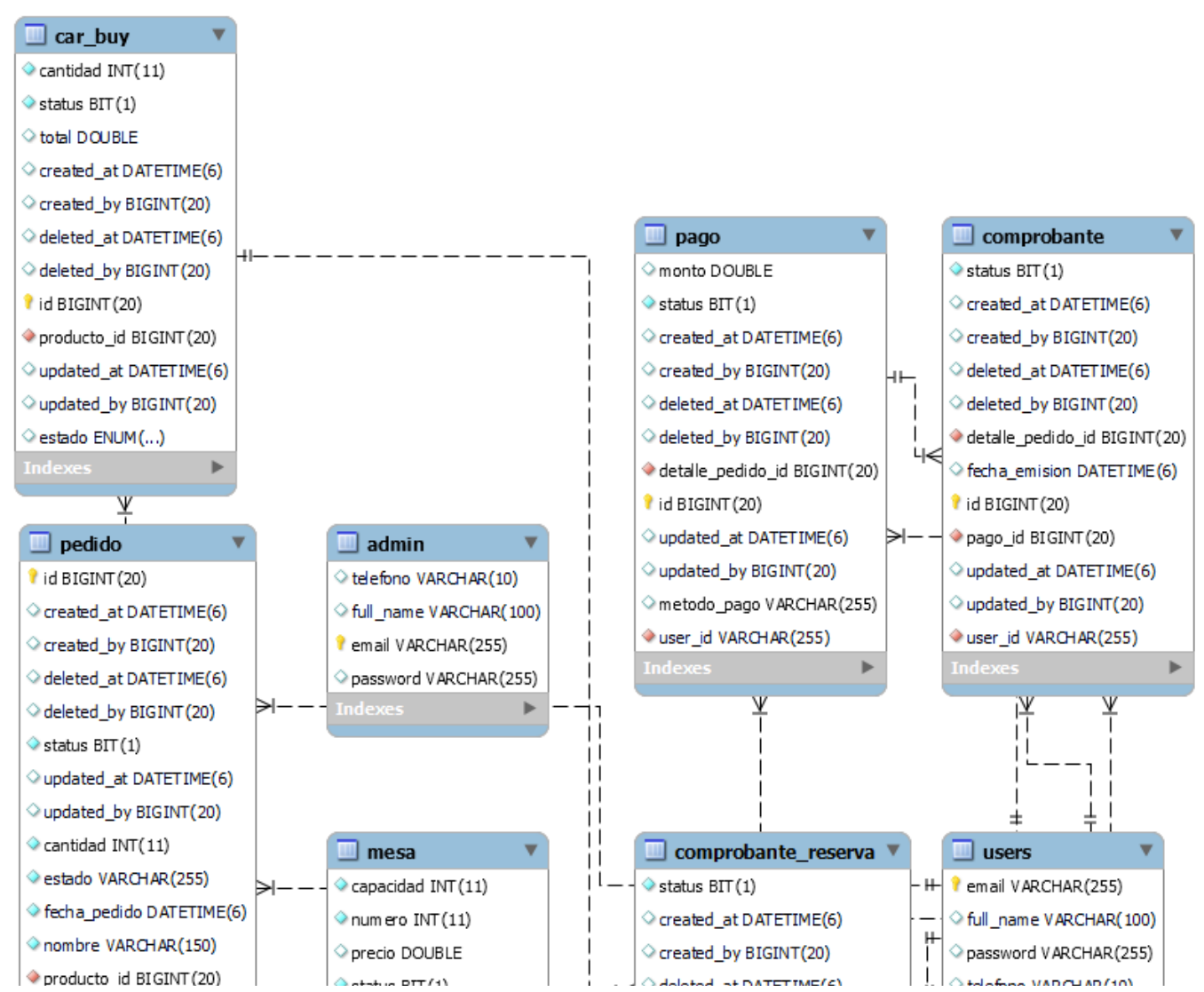
El modelo entidad-relación representa la estructura de los datos del sistema, incluyendo entidades clave como usuarios, administradores, productos, mesas, reservas, pagos y carrito de compras, ya que estas entidades se relacionan para organizar efectivamente las operaciones de la cafetería.

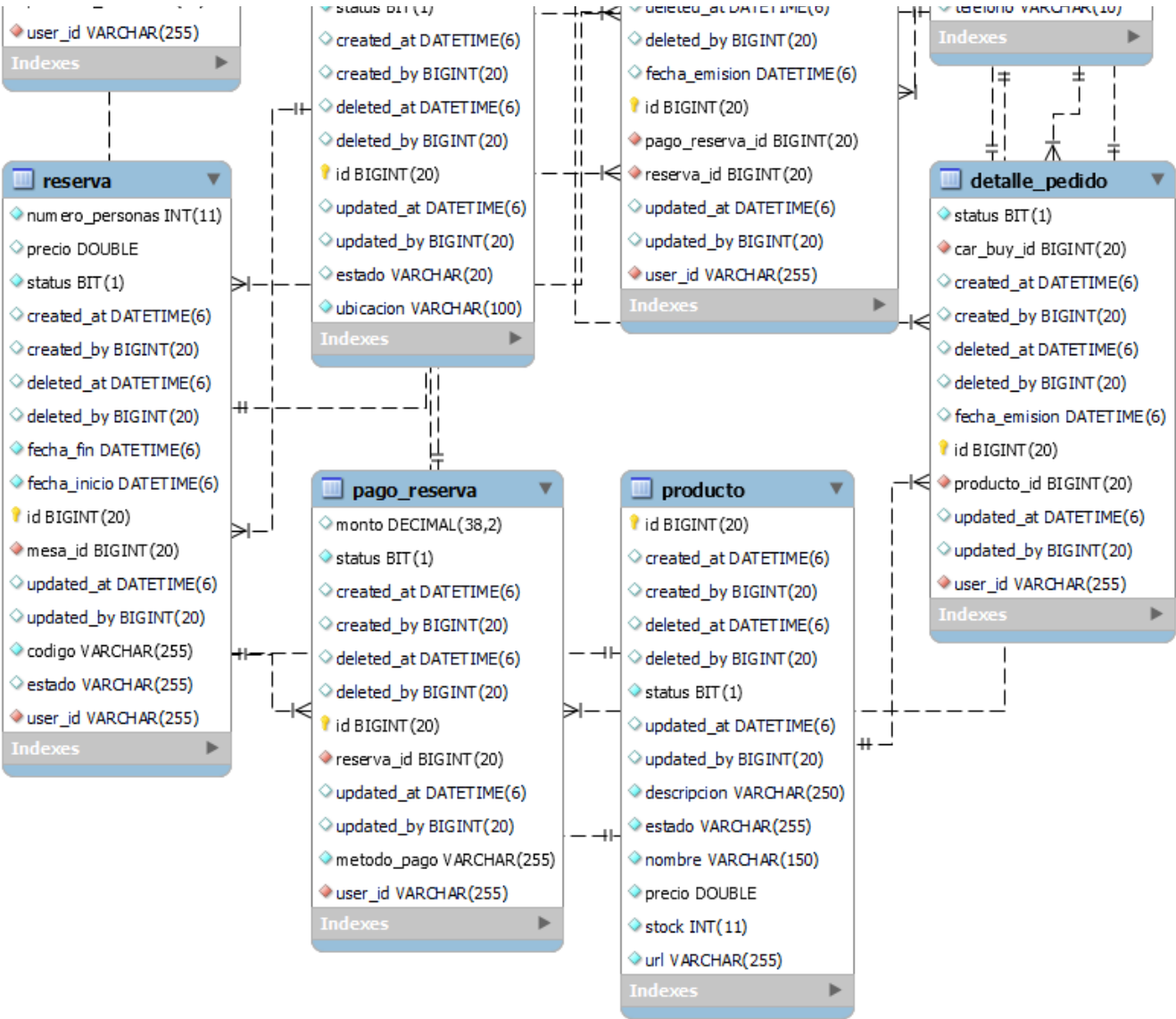
Este modelo facilita el registro de reservas, control de inventario, gestión de pagos y generación de comprobantes, diferenciando claramente las funciones de administrador y usuario, lo que mejora la seguridad y eficiencia del sistema.



Modelo Relacional (MR)

El modelo relacional define cómo se almacenan y gestionan los datos en tablas relacionadas para procesos clave como usuarios, productos, pedidos, pagos, reservas y mesas, además, permite registrar compras, administrar reservas y mantener la integridad de la información mediante auditoría y eliminación lógica.

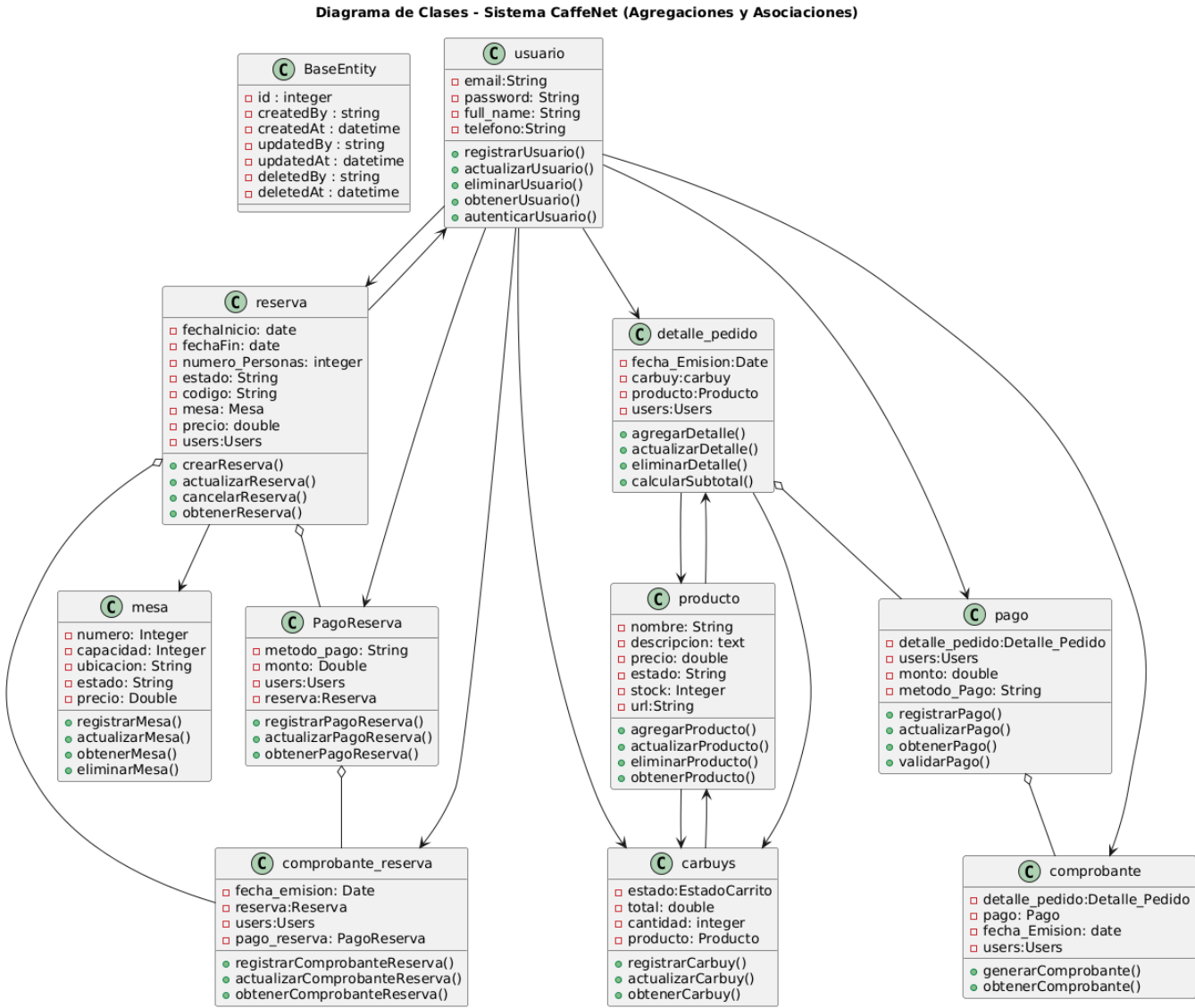




Diagramas UML:

1. Clases

El diagrama de clases presenta la estructura orientada a objetos del sistema, mostrando las clases principales y sus relaciones, soportando una gestión eficiente de usuarios, productos, reservas, pagos y pedidos, facilitando escalabilidad, reutilización y mantenimiento.

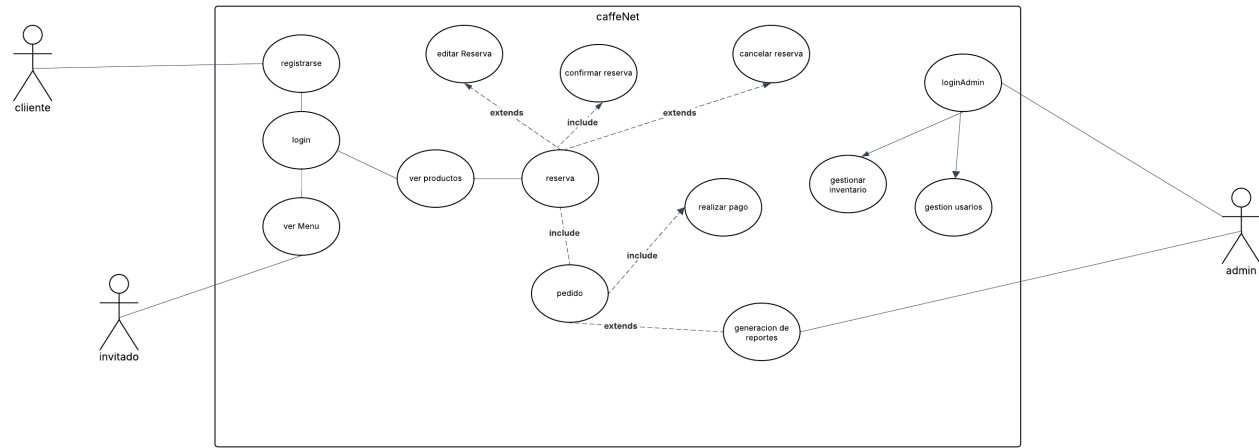


2. Caso de Uso

Este diagrama explica las tareas principales del sistema y la conexión con los actores: Cliente, Invitado y Administrador.

- Cliente: puede registrarse, realizar reservas, pedidos y pagos.
- Invitado: solo visualiza el menú y productos.
- Administrador: gestiona inventario, usuarios y reportes.

Se incluyen relaciones de tipo «include» (funcionalidades obligatorias) y «extend» (funcionalidades opcionales) para mostrar el flujo de acciones.



2. Secuencia

Este diagrama muestra el recorrido del usuario dentro de la aplicación, desde el registro o ingreso como invitado, pasando por la navegación del menú, la gestión de reservas y pedidos, hasta la generación de reportes y calificación del servicio. Representa decisiones clave que guían el flujo del proceso.

