

DOCUMENTO DE DISEÑO DEL SISTEMA

1. Datos generales

- **Proyecto:** Sistema Web de Gestión y Pedido Interactivo para Cafetería (Latacunga)
- **Versión:** 1.0
- **Fecha:** 15/01/2026
- **Responsable:** Equipo de proyecto

2. Objetivo del documento

Definir el diseño lógico y técnico del sistema, especificando la arquitectura, modelos de datos, componentes principales y diagramas UML necesarios para guiar el desarrollo e implementación.

3. Alcance del diseño

El diseño cubre:

- Modelo de datos (ER)
- Modelo de clases (dominio y principales entidades)
- Flujos clave (diagramas de secuencia)
- Arquitectura (C4 / Componentes)
- Diseño de despliegue (infraestructura)

4. Arquitectura del sistema

4.1 Arquitectura C4 – Nivel 1 (Contexto)

Sistema: Plataforma web para pedidos y administración de cafetería.

Actores externos:

- **Cliente:** accede al menú por QR y realiza pedidos.
- **Cajero/a:** gestiona pedidos, pagos y entrega.
- **Cocina:** prepara pedidos y actualiza estados.
- **Gerente:** administra menú, usuarios y reportes.
- **Servicios externos (opcionales):** pasarela de pago, servicio de correo o notificaciones push.

Relaciones principales:

- Cliente ↔ Sistema (consultar menú, crear pedido, ver estado)
- Caja ↔ Sistema (confirmar pago, revisar pedidos)
- Cocina ↔ Sistema (cambiar estado, ver cola de preparación)
- Gerente ↔ Sistema (CRUD productos/usuarios, reportes)

4.2 Arquitectura C4 – Nivel 2 (Contenedores)

Contenedores del sistema:

1. **Frontend Web (Cliente y Paneles)**
 - Interfaz responsive (menú + panel caja + panel cocina + panel gerente)
2. **API Backend (Servicios REST)**
 - Autenticación, pedidos, productos, reportes, notificaciones
3. **Base de Datos Relacional**
 - Persistencia de usuarios, productos, pedidos, estados
4. **Servicio de Notificaciones (opcional)**
 - WebSockets / SSE / notificaciones por correo (si se implementa)

4.3 Arquitectura por Componentes (Nivel 3)

Backend – módulos/componentes:

- **Auth Module:** login, roles, permisos, tokens
- **Menu Module:** categorías, productos, disponibilidad
- **Orders Module:** carrito, creación de pedido, estados
- **Kitchen Module:** cola de pedidos, cambios de estado
- **Cashier Module:** confirmación de pago / control de entrega
- **Reports Module:** consultas agregadas por fecha y productos
- **Notifications Module:** emisión de eventos y mensajes a roles

Patrón recomendado:

- Separación por capas: Presentación (UI), Aplicación (servicios), Dominio (entidades/reglas), Infraestructura (BD).
-

5. Modelo de Datos (ER)

5.1 Entidades principales

- **Usuario (User)**
- **Rol (Role)**
- **Producto (Product)**
- **Categoría (Category)**
- **Pedido (Order)**
- **DetallePedido (OrderItem)**
- **EstadoPedido (OrderStatusHistory)**

5.2 Relación entre entidades (descripción)

- Un **Rol** tiene muchos **Usuarios**.
- Una **Categoría** contiene muchos **Productos**.
- Un **Pedido** pertenece a un **Cliente** (Usuario) o registra datos mínimos (si se permite pedido sin cuenta).
- Un **Pedido** tiene muchos **DetallePedido**.
- Un **Pedido** tiene historial de estados (**EstadoPedido**).

5.3 Diccionario de datos (ejemplo resumido)

Tabla: Role

- id (PK)
- name (Gerente, Cajero, Cocina, Cliente)

Tabla: User

- id (PK)
- full_name
- email (unique)
- password_hash
- role_id (FK → Role.id)
- is_active
- created_at

Tabla: Category

- id (PK)
- name

- is_active

Tabla: Product

- id (PK)
- name
- description
- price
- image_url (opcional)
- is_available
- category_id (FK → Category.id)

Tabla: Order

- id (PK)
- order_code (unique)
- customer_id (FK → User.id, nullable si no hay cuenta)
- total
- current_status
- created_at

Tabla: OrderItem

- id (PK)
- order_id (FK → Order.id)
- product_id (FK → Product.id)
- quantity
- unit_price

- subtotal

Tabla: OrderStatusHistory

- id (PK)
 - order_id (FK → Order.id)
 - status (Pendiente, Confirmado/Pagado, En preparación, Listo, Entregado)
 - changed_by (FK → User.id)
 - changed_at
-

6. Diseño de Clases (UML)

6.1 Clases principales (conceptual)

- **User**
 - id, fullName, email, role, isActive
- **Role**
 - id, name
- **Category**
 - id, name, isActive
- **Product**
 - id, name, description, price, isAvailable, category
- **Order**
 - id, orderCode, customer, total, currentStatus, createdAt
 - methods: calculateTotal(), updateStatus()

- **OrderItem**
 - id, product, quantity, unitPrice, subtotal
 - methods: calculateSubtotal()
- **OrderStatusHistory**
 - id, order, status, changedBy, changedAt

6.2 Relaciones (resumen UML)

- Role 1..* User
 - Category 1..* Product
 - Order 1..* OrderItem
 - Order 1..* OrderStatusHistory
 - Product 1..* OrderItem
-

7. Diagramas de Secuencia (flujos clave)

7.1 Secuencia: Cliente realiza pedido por QR

Participantes: Cliente → Frontend → Orders API → DB → Notifications

Flujo:

1. Cliente escanea QR y abre menú.
2. Frontend solicita productos (GET /products).
3. Cliente agrega al carrito y confirma.
4. Frontend envía pedido (POST /orders).
5. Orders API valida disponibilidad y guarda Order + OrderItems.
6. DB responde con order_code y estado “Pendiente”.

7. Notifications emite evento a caja y cocina.
8. Frontend muestra estado del pedido al cliente.

7.2 Secuencia: Cocina actualiza estado

Participantes: Cocina → Frontend → Kitchen API → DB → Notifications
Flujo:

1. Cocina visualiza cola de pedidos (GET /orders?status=pendiente).
2. Selecciona pedido y cambia estado (PATCH /orders/{id}/status).
3. API guarda historial de estados y actualiza estado actual.
4. Notifications notifica a cliente y caja (si aplica).

7.3 Secuencia: Gerente administra producto

Participantes: Gerente → Admin UI → Menu API → DB
Flujo:

1. Gerente crea/edita producto (POST/PUT /products).
2. API valida campos y guarda en DB.
3. Menú se actualiza en tiempo real (si hay cache, se invalida).

8. Diseño de Interfaces (UI) – resumen

- **Vista Menú (Cliente):** categorías, productos, carrito, confirmar pedido, estado.
 - **Panel Caja:** lista de pedidos, confirmación de pago, filtro por estado.
 - **Panel Cocina:** cola de preparación, cambiar estado, orden de llegada.
 - **Panel Gerente:** CRUD productos/categorías/usuarios, reportes.
-

9. Diseño de Despliegue (Infraestructura)

9.1 Componentes de despliegue

- **Cliente:** Navegador web móvil/PC
- **Servidor de Aplicación:** Backend (API REST)
- **Servidor Web/Frontend:** hosting estático o contenedor
- **Base de Datos:** PostgreSQL/MySQL
- **Balanceador/Proxy (opcional):** Nginx
- **Certificado SSL:** HTTPS

9.2 Escenario de despliegue recomendado (simple)

- Frontend: hosting estático (o mismo servidor)
- Backend: VPS con Docker (opcional) o servicio PaaS
- BD: instancia gestionada o local en VPS

9.3 Requisitos de seguridad en despliegue

- HTTPS obligatorio
 - Variables de entorno para credenciales
 - Backups diarios/semanales de BD
 - Control de acceso por roles y logs de auditoría
-

10. Consideraciones de diseño y estándares

- Nomenclatura consistente (RF-xx, HU-xx, CU-xx)
 - Uso de patrones (MVC / Clean Architecture recomendado)
 - Validación de datos y manejo de errores (códigos HTTP)
 - Versionado de API (ej. /api/v1)
-

11. Conclusión

El diseño propuesto define una arquitectura modular, con separación clara de responsabilidades, modelo de datos relacional y flujos principales mediante secuencias, permitiendo implementar el sistema de forma mantenable y escalable.