INSTITUTO PROFESIONAL
SAN SEBASTIAN

EVALUACIÓN DE UNIDAD 2

DESARROLLO DE SOFTWARE WEB II

Profesor
Víctor Cofre Farias
Integrantes
Osvaldo Maulén
Sebastian Lucero

Documentación del código

Este documento describe la arquitectura, funcionamiento y componentes principales del sistema de gestión de clientes, mesas y reservaciones desarrollado con Spring Boot, Thymeleaf y PostgreSQL.

Arquitectura del proyecto SaborGourmet

La arquitectura del sistema sigue el patrón MVC (Model–View–Controller) junto con capas adicionales de Servicios y Repositorios, lo que permite mantener el proyecto modular, escalable y fácil de mantener.

- Capa de Presentación (Views – HTML + Thymeleaf) : Responsable de mostrar la información al usuario y recibir sus datos desde formularios.

/templates/clientes/

- list.html
- form.html

/templates/mesas/

- list.html
- form.html

/templates/reservaciones/

- list.html
- form.html

index.html

Rol: Renderiza las pantallas, muestras listas, formularios y mensajes de error o éxito.

- Capa de Controladores (Controllers): Aquí llegan las peticiones HTTP.
- Los controladores llaman a los servicios, procesan la respuesta y finalmente muestran una vista.

- ClienteController
- ClienteViewController
- MesaController
- MesaViewController
- ReservacionController
- ReservacionViewController

Rol: Gestionan rutas como GET /clientes, POST /mesas, GET /reservaciones/crear. Conectan la vista con la lógica del negocio.

- Capa de Servicios (Services): Contiene la lógica del negocio.

Interfaces:

- ClienteService
- MesaService
- ReservacionService

Implementaciones (impl):

- ClienteServiceImpl
- MesaServiceImpl
- ReservacionServiceImpl

Rol:
Aquí vive la lógica como:

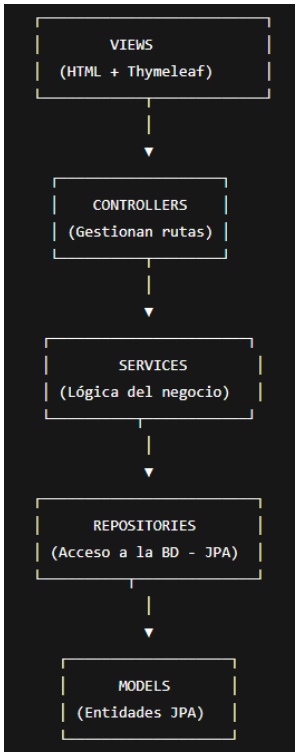
- Validar cantidad de personas en una mesa
- Crear, editar, eliminar clientes
- Reglas de reservas
- Manejo de errores
- Validaciones personalizadas

Los controladores **nunca hablan directamente con la base de datos**, siempre pasan por aquí.

- Capa de Acceso a Datos (Repositories): Utiliza Spring Data JPA para comunicarse con la base de datos.
- ClienteRepository
- MesaRepository
- ReservacionRepository

Rol:
Proveen métodos como:

- save(), findAll(), findById(), delete()
- Consultas personalizadas si son necesarias
- **Capa de Modelo (Entities)**
- Clases que representan las tablas de la base de datos.
- Cliente
- Mesa
- Reservacion
- EstadoReservacion
- **Rol:**
  Definen atributos, relaciones, validaciones básicas y se mapean a la BD mediante JPA.



Estructura de Paquetes

• controllers: Manejan solicitudes HTTP.

• models: Entidades JPA.

• repositories: Repositorios JPA para CRUD.

• services: Interfaces con lógica de negocio.

• services.impl: Implementaciones de servicios.

• templates: Formularios y listados HTML.

Modelos (Entidades)

• Cliente: id, nombre, apellido, teléfono, email.

• Mesa: id, número, capacidad.

• Reservacion: id, cliente, mesa, fecha inicio, fecha término, personas, estado.

• EstadoReservacion: Enum con estados (PENDIENTE, CONFIRMADA, CANCELADA).

Repositorios

Cada entidad tiene un repositorio que extiende JpaRepository, permitiendo operaciones CRUD y consultas personalizadas.

Servicios

Los servicios contienen la lógica de negocio, incluyendo:

- Validación de capacidad de la mesa.

- Detección de choques de reservas.

- Manejo de integridad referencial.

Controladores

• Controllers REST: Proveen endpoints JSON.

• ViewControllers: Renderizan páginas Thymeleaf.

Se encargan de conectar las vistas con la lógica de negocio.

Validaciones Importantes

• No permitir reservar más personas que la capacidad de la mesa.

• Prevenir reservas que se traslapen en horario.

• Evitar eliminar clientes con reservas activas (error de integridad referencial).

Flujo de Reservas

I. El usuario crea una reserva seleccionando mesa y cliente.
II. El sistema valida:
    a. Capacidad disponible.
    b. No existencia de choque horario.
III. Si pasa las validaciones, la reserva se guarda.
IV. Si falla, muestra un mensaje de error en pantalla.

Conclusión

El sistema permite gestionar de forma eficiente las operaciones principales del restaurante, con una arquitectura modular, validaciones completas y una interfaz simple basada en Thymeleaf.