TODO para completar el proyecto en un día:

Preparativos Iniciales

1. **Configurar el proyecto:**
   - Crear un proyecto Spring Boot con Java 17 y Gradle.
   - Configurar la conexión con la base de datos SQL (usar MySQL Workbench y SQLyog Community).
2. **Estructura básica del proyecto:**
   - Crear paquetes:
     - `controller` para controladores.
     - `service` para la lógica de negocio.
     - `repository` para las operaciones con la base de datos.
     - `entity` para las entidades JPA.
     - `dto` para los objetos de transferencia de datos.
3. **Dependencias necesarias:**
   - Incluir dependencias en `build.gradle`:
     - Spring Web
     - Spring Data JPA
     - MySQL Connector
     - Lombok
     - Spring DevTools (opcional para desarrollo más rápido).

Implementación de Funcionalidades

1. **Entidades JPA:**
   - Crear las clases para las tablas principales:
     - `Customer`

    - `Menu`
    - `Dish`
    - `Order`
  - Asegurarse de incluir las relaciones necesarias (e.g., `OneToMany`, `ManyToOne`).
2. **Repositorios:**
  - Crear interfaces que extiendan `JpaRepository` para cada entidad.
3. **Controladores y Servicios:**
  - Implementar controladores y servicios para:
    - **Clientes:** Crear, listar, actualizar, eliminar.
    - **Menús:** Crear, listar, actualizar, eliminar.
    - **Platos:** Crear, listar, actualizar, eliminar.
    - **Pedidos:** Crear, listar, actualizar, eliminar.
4. **Reglas de Negocio:**
  - **Cliente frecuente:**
    - Implementar lógica para identificar clientes frecuentes (consulta en la base de datos para verificar pedidos).
    - Aplicar el descuento del 2.38% en pedidos de clientes frecuentes.
  - **Platos populares:**
    - Implementar lógica para identificar platos populares (consulta en la base de datos para verificar cantidad de compras).
    - Incrementar precio de platos populares en 5.73%.
5. **Patrones de Diseño:**
  - Usar **Observer** para actualizar automáticamente el estado de clientes y platos.
  - Implementar **Chain of Responsibility** para el manejo de reglas de negocio (descuentos, incrementos de precio).
6. **Validación de Operaciones HTTP:**
  - Implementar y probar métodos HTTP:

- `GET`, `POST`, `PUT`, `DELETE`.

Pruebas

1. **Pruebas con Postman/Insomnia:**
   - Crear una colección con todas las rutas de la API.
   - Probar cada operación CRUD para cada recurso.
   - Validar reglas de negocio (clientes frecuentes, platos populares).
2. **Pruebas unitarias:**
   - Agregar pruebas básicas para servicios usando JUnit.

Documentación

1. **README:**
   - Incluir pasos para ejecutar el proyecto.
   - Describir las reglas de negocio y cómo probarlas.
2. **Swagger:**
   - Incluir Swagger para documentar la API (opcional).

Planificación del Día

- **Primera hora:** Configurar proyecto, dependencias, y base de datos.
- **Horas 2-4:** Crear entidades, repositorios y controladores básicos.
- **Horas 5-6:** Implementar reglas de negocio y patrones de diseño.
- **Hora 7:** Realizar pruebas con Postman/Insomnia.
- **Hora 8:** Documentar el proyecto.

Si organizas bien las tareas y ya tienes experiencia básica en Spring Boot, este proyecto es realizable en un día. ¿Quieres comenzar con alguna de estas tareas?