**Tarea: API de Gestión de Reservas de Restaurantes**

Desarrollar una API REST que permita gestionar un sistema de reservas para un restaurante. Con la API se podrá:

1. Administrar clientes, mesas y reservas.
2. Consultar la disponibilidad de mesas para realizar reservas.
3. Validar los datos de entrada en todas las solicitudes para garantizar la integridad de la información.
4. Crear documentación de la API de forma automática mediante Swagger
5. Proteger los endpoints con autenticación y autorización mediante JWT.

**A. Modelo de Datos**

1. Diseñar las siguientes entidades con sus respectivas relaciones:
    - **Cliente**: Información básica como nombre, email y teléfono.
    - **Mesa**: Incluye un número único (a parte del ID) y descripción.
    - **Reserva**: Registra la fecha, hora, cliente, mesa y número de personas.
2. Implementar las relaciones entre las tablas:
    - Un cliente puede tener múltiples reservas pero cada reserva es de un único cliente.
    - Cada reserva debe estar asociada a una única mesa pero cada mesa tendrá múltiples reservas asociadas (en distintas fechas y/o horas).

**B. CRUD Completo**

Implementar las operaciones CRUD para:

1. **Clientes**: Crear, actualizar, eliminar y listar clientes.
2. **Mesas**: Crear, actualizar, eliminar y listar mesas.
3. **Reservas**:
    - Crear reservas verificando disponibilidad de mesas.
    - Borrar reserva a partir de su id
    - Listar las reservas de un día concreto utilizando un DTO donde se muestren los datos de la reserva, los datos de la mesa y los datos del cliente todo en un mismo objeto JSON del tipo: {nombre: ”Manolo”, email: ”m@m.com”, fecha_reserva: ”11/11/2025”, numero_mesa: 6, etc.)

**C. Validación de Datos**

1. Validar los datos de entrada en todas las solicitudes usando Spring Validation.
2. Ejemplos de validaciones:
    - El nombre de un cliente debe tener al menos 3 caracteres.

- El email debe tener un formato válido.
- No permitir reservas en fechas pasadas.
- Al insertar una reserva comprobar previamente que está libre para esa fecha

**D. Seguridad**

1. Implementar seguridad utilizando **Spring Security** con **JWT**.
2. Crear un endpoint para el login, que genere un token JWT al autenticarse con éxito.
3. Asegurar que los endpoints de inserción, modificación y borrado están protegidos.

**E. Documentación**

1. Documentar la API con Swagger para que sea fácil de usar y entender.

**F. Opcionales** (obligatorio hacer al menos uno si se hace en parejas)

1. Añadir una tabla en la BD con dos roles con las siguientes restricciones de acceso:
    1. Los usuarios con rol ROLE_ADMIN pueden acceder y gestionar clientes y mesas.
    2. Los usuarios con rol ROLE_USER solo pueden gestionar reservas.
2. Añadir paginación al listado de clientes (Ver cómo aquí)
3. "Dockerizar" la aplicación y MySQL y desplegar en un servidor como Heroku o similar (Ver cómo aquí)

---

**3. Entregables**

1. Subir código fuente completo del proyecto a GitHub y enviar enlace al repositorio.
2. Añadir enlace a la API desplegada si se despliega en Heroku o similar.
3. Enseñar funcionamiento al profesor en clase cuando se tenga terminada la tarea