Proyecto API Rest Cliente (restDemo)

Se realiza proyecto de una API Rest básica de gestión de clientes (Cliente), creado con lenguaje Java con Spring Boot, en el cual para las pruebas de los endpoints de la API se ejecutan con las operaciones CRUD completas (Create, Read, Update, Delete), con consultas personalizadas, conectándose a base de datos MySQL y el despliegue en Docker.

Tecnologías utilizadas:

- Java 17
- Spring Boot
- Spring Data JPA.
- MySQL
- Maven
- Docket
- Postman (opcional para las pruebas).

Se construye la API basándose en la estructura por capas (Entity, Repository, Service y Controller).

1- Para la capa **Entity** (o Model, aquí se representa el modelo de datos):

Se crea la clase “Cliente” (com.academy.rest.entity).

En la clase se generan los campos siguientes → id, nombre, email, telefono, fechaRegistro. Utiliza también LocalDateTime para registrar automáticamente la fecha de creación del cliente.

Y dentro de la clase Cliente se crean los constructores de cada campo, y sus getters y setters y se adjuntan las siguientes anotaciones:

- @Entity, @Table para hacer el mapeo a la tabla Cliente.
- @Id, @GeneratedValue, como identificador automático.
- @Column

2- Para la capa **Repository:** (Encapsula la lógica de acceso a la base de datos).

Se crea la clase “ClienteRepository”, la cual extiende o hereda de “JpaRepository”.

En está clase se extiende JpaRepository para el acceso a la base de datos, y se incluyen también:

- Un Optional, con método findByEmail, para filtrar la búsqueda de los clientes por email.
- Un List, para la búsqueda por nombre, usandoló en método “findByNombreContainingIgnoreCase”.
- Consultas personalizadas con @Query
    - findByTelefono
    - findAllOrderByFechaRegistroDesc

3- Capa **Service:**

a. Service Interface → **IClienteService.java**

Se define en la clase IClienteService las operaciones del negocio disponibles para los clientes. Abstracción útil para desacoplar.

- Obtener todos los clientes.
- Buscar cliente por ID, email, nombre o telefono.
- Crear, actualizar o eliminar un cliente.
- Listar clientes ordenados por fecha de registro.

b. Service Implementation (Implementación del servicio) → **ClienteServiceImpl.java**

En está clase se implementa la lógica de negocio real. Válida que no se registren correos duplicados antes de crear o actualizar. Llama al repositorio para ejecutar las operaciones necesarias, manejando excepciones cuando un cliente no se encuentra.

4- Capa **Controller:** ClienteController.java

Controlador Rest (@RestController) que expone los endpoints HTTP.

Maneja respuestas con ResponseEntity y códigos HTTP adecuados (

200 OK, 201 Created, 400 Bad Request, 404 Not Found).

Los principales endpoints de la API son:

Método	URL	Función
GET	/api/clientes	Obtener todos los clientes.
GET	/api/clientes/{id}	Obtener clientes por ID.
PUT	/api/clientes/{id}	Actualizar cliente
POST	/api/clientes	Crear Cliente
DELETE	/api/clientes/{id}	Eliminar un cliente.
GET	/api/clientes/email/{email}	Buscar por email
GET	/api/clientes/nombre/{nombre}	Buscar por nombre parcial
GET	/api/clientes/telefono/{telefono}	Buscar por telefono
GET	/api/clientes/ordenados	Ordenar lista por fecha

Se utiliza Docker como contenedor de MySQL, el cual fue MySQL en donde se realizó la conexión con la API en **application.properties**.

Y como adicional, se realizó, para completar la API, su parte de front-end, utilizando lenguaje JavaScript, creando el archivo HTML, CSS y la lógica de los botones con JS también.