ACCESO A DATOS

UD06: PRÁCTICA 3 - SPRING BOOT (DTO)

Realiza un proyecto Java denominado **adt6_practica3** que contenga todo lo que se pide a continuación. Para ello crea un base de datos con el mismo nombre en MySQL.

Gestión de usuarios

El objetivo de esta práctica es profundizar en el uso de DTOs (Data Transfer Objects) en una aplicación Spring Boot, utilizando la entidad Usuario y los DTOs UsuarioRequestDto y UsuarioResponseDto.

Para ello vamos a crear una API Rest que ofrezca servicios web para la gestión de usuarios. La aplicación tendrá una base de datos de usuarios donde almacenará y tendrá las siguiente validaciones a nivel de DTOs:

- nombre (no puede ser blanco)
- apellidos (debe tener entre 6 y 50 caracteres)
- email (debe ser un correo válido)
- password (debe ser letras mayúsculas y/o minúsculas y/o números entre 4 y 8 caracteres)
- fechaCreación

Deberá ofrecer las siguientes operaciones básicas:

1. Búsqueda de un usuario concreto.
2. Búsqueda de todos los usuarios.
3. Registro de un nuevo usuario.
4. Modificar un usuario.
5. Dar de baja un usuario.

Crear los DTOs asociados

Para la creación y actualización de usuarios utilizaremos el DTO UsuarioRequestDto con los siguientes datos, el resto de información se creará automáticamente:

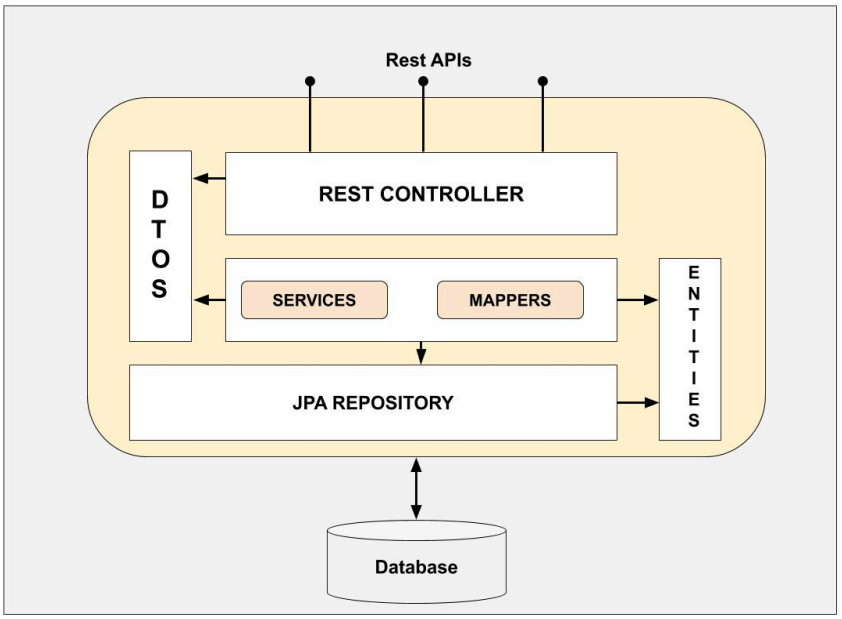
- nombre, apellidos, email, password.

Para las respuestas utilizaremos el DTO UsuarioResponseDto con los siguientes datos, el resto es oculto para los clientes de nuestra API:

- nombre, email, fechaCreación.

Recordar hacer la configuración del ModelMapper, es decir, añadir la dependencia al pom.xml y configurar un Bean de ModelMapper como lo tenéis en los apuntes.

Programación funcional API de Stream



Para realizar las transformaciones entre los DTOs y las entidades podemos hacer uso de la programación funcional con la **API de Stream**. Esta API nos ofrece una herramienta que permite el procesamiento de datos de forma eficiente. Los streams son colecciones de datos que se pueden procesar, facilitando operaciones de agregación y transformación sobre este tipo. Podemos hacer dos tipos de operaciones:

- **Operaciones intermedias:** son aquellas que transforman un stream en otro stream. Estas operaciones son "perezosas", lo que significa que no se ejecutan hasta que se invoca una operación terminal sobre el stream.
  - **filter(Predicate<T> predicate):** filtra elementos en función de una condición.
  - **map(Function<T, R> mapper):** transforma los elementos aplicando una función.
  - **distinct():** elimina elementos duplicados del stream.
  - **sorted():** ordena los elementos del stream según su orden natural.
  - **sorted(Comparator<T> comparator):** ordena según el comparador proporcionado.
- **Operaciones terminales:** son aquellas que producen un resultado. Estas operaciones provocan la ejecución de las operaciones intermedias previamente aplicadas.
  - **forEach(Consumer<T> action):** realiza una acción dada por cada elemento.
  - **toArray():** convierte el stream en un array.
  - **collect(Collectors.toList()) o toList():** convierte el stream a un listado.
  - **reduce(BinaryOperator<T> accumulator):** combina los elementos en un único valor.
  - **count():** cuenta el número de elementos en el stream.
  - **findFirst():** devuelve un elemento de tipo Optional.

Más información en el canal de Aula en la nube:

- https://www.youtube.com/watch?v=pip5g8KfNUo

📖 Programación funcional

Es un paradigma de programación que se centra en el uso de funciones como unidades fundamentales. En lugar de cambiar variables y estados (programación imperativa), la programación funcional se centra en realizar transformaciones a los datos a través de funciones inmutables. Tenemos 5 conceptos que nos ayudan a realizar este tipo de programación: *funciones Lambdas, funciones de orden superior, funciones puras, inmutabilidad y streams*.

Más información:

- https://www.youtube.com/watch?v=80a0TRSZDEQ

Datos de ejemplo:

```
INSERT INTO usuario(nombre, apellidos, email, password, fecha_creacion)
VALUES
   ('Carlos', 'García López', 'carlosgl@example.com', 'pass1', '2024-01-01'),
   ('Ana', 'Martínez Ruiz', 'anamr@example.com', 'pass2', '2024-01-02'),
   ('José', 'Sánchez Fernández', 'josesf@example.com', 'pass3', '2024-01-03'),
   ('Lucía', 'Rodríguez Pérez', 'luciarp@example.com', 'pass4', '2024-01-04'),
   ('Manuel', 'Gómez Alonso', 'manuelga@example.com', 'pass5', '2024-01-05'),
   ('Sara', 'López Morales', 'saralm@example.com', 'pass6', '2024-01-06'),
   ('Pablo', 'Muñoz Martín', 'pablomm@example.com', 'pass7', '2024-01-07'),
   ('Laura', 'Díaz Sánchez', 'laurads@example.com', 'pass8', '2024-01-08'),
   ('David', 'Moreno Jiménez', 'davidmj@example.com', 'pass9', '2024-01-09'),
   ('Marta', 'Romero Ortiz', 'martaro@example.com', 'pass10', '2024-01-10'),
   ('Javier', 'Torres Gutiérrez', 'javiertg@example.com', 'pass11', '2024-01-11'),
   ('Elena', 'Gil Navarro', 'elenagn@example.com', 'pass12', '2024-01-12'),
   ('Álvaro', 'Vázquez Rubio', 'alvarovr@example.com', 'pass13', '2024-01-13'),
   ('Beatriz', 'Herrera Molina', 'beatrizhm@example.com', 'pass14', '2024-01-14'),
   ('Diego', 'Cano Ramírez', 'diegocr@example.com', 'pass15', '2024-01-15'),
   ('Irene', 'Prieto Castillo', 'irenepc@example.com', 'pass16', '2024-01-16'),
   ('Oscar', 'Santos Cabrera', 'oscarsc@example.com', 'pass17', '2024-01-17'),
   ('Patricia', 'Garrido Rey', 'patriciagr@example.com', 'pass18', '2024-01-18'),
   ('Rubén', 'Cortés Domínguez', 'rubencd@example.com', 'pass19', '2024-01-19'),
   ('Nerea', 'Campos Soto', 'nereacs@example.com', 'pass20', '2024-01-20');
```