Este documento pretende explicar una aplicación sencilla de Spring Boot centrándose solo en los archivos programados con Java (también se hizo uso de JavaScript, CSS y demás, pero para este ejercicio solo nos enfocaremos en Java).

restDemo / rest /src / main / java / com / academy / rest / config / CorsConfig.java

El primer archivo de la carpeta config es el CorsConfig.java. Este archivo configura el Cross-Origin Resource Sharing (CORS) permitiendo que el front-end se comunique con el back-end. CORS es un mecanismo de seguridad que impide o permite que un dominio pueda hacer peticiones a otro dominio.

Al iniciar la aplicación, Spring detecta la clase CorsConfig por la anotación @Configuration. La anotación @Bean crea un objeto que permite reutilizarlo si es necesario, inyectar sus dependencias y controlar su ciclo de vida; más práctico que usar "new" muchas veces.

El corsFilter() crea un filtro que interceptará las peticiones HTTP y aplicará la configuración CORS.

setAllowCredentials en true permite que se envíen cookies, tokens o credenciales en las peticiones entre dominios.

Al inicio se hace un import de Arrays; esto es porque en el bloque de corsFilter() se hace una lista para especificar qué dominios están permitidos para hacer peticiones al back-end (localhost:8080), permite cualquier encabezado HTTP (*), se definen los métodos HTTP permitidos (GET, POST, PUT, DELETE, OPTIONS), y se crea un objeto para asociar la configuración CORS a todas las rutas del back-end (urlBasedCorsConfigurationSource y "/**").

Finalmente se hace un return con del filtro CORS con esta nueva configuración.

restDemo / rest /src / main / java / com / academy / rest / config / WebConfig.java

Este archivo es solo para que Spring Boot encuentre todo lo relacionado al front-end en la carpeta src/main/resources/static.

restDemo / rest /src / main / java / com / academy / rest / controller / ClienteController.java

Este archivo es un controlador; eso significa que permite controlar las peticiones HTTP para que estas devuelvan información adecuadamente.

Los imports com.academy no son parte de Spring Boot, más bien son archivos creados por nosotros mismos y al importarlos podemos (prácticamente) maniobrar con lo programado en ellos desde otros archivos.

La anotación @RestController le dice a Spring que esta clase expone una API REST.

La anotación @RequestMapping(“[URL]”) establece que todas las rutas dentro de este controlador comienzan con la URL dentro de sus paréntesis; para este ejercicio “/api/clientes”.

La anotación @Autowired hace inyección de dependencias (traer un objeto desde otra clase). Para este ejercicio se inyecta una implementación de la interfaz IClienteService.

Las anotaciones @GetMapping, @PostMapping, @PutMapping y @DeleteMapping contienen lo que se va a ejecutar ante una petición GET, POST, PUT y DELETE respectivamente.

La anotación @PathVariable extrae valores desde la URL; para este ejercicio se extrae el valor del ID.

La anotación @RequestBody toma el cuerpo del request JSON y lo convierte en algún objeto; para este ejercicio lo convierte en un objeto Cliente.

Finalmente, este archivo contiene métodos para obtener todos los clientes, obtener cliente por ID, crear cliente, actualizar cliente, eliminar cliente, buscar cliente por email, buscar clientes por nombre, buscar cliente por teléfono, mostrar clientes ordenados por fecha de registro.

restDemo / rest /src / main / java / com / academy / rest / controller / HelloController.java

Este archivo parece creado por defecto por Spring Boot; no hace gran cosa, solo devuelve el texto "Hello World Spring Boot" al visitar la URL /hello (solicitud GET).

restDemo / rest /src / main / java / com / academy / rest / entity / Cliente.java

Este archivo define la entidad Cliente. Utiliza JPA (Hibernate) para interactuar con la base de datos (me recuerda a LINQ con C#).

La anotación @Entity marca esta clase como una entidad JPA, es decir, una tabla en la base de datos; y la anotación @Table(name = ”Cliente”) especifica el nombre exacto de la tabla para no usar el nombre de la clase por defecto (recordar que en programación es muy diferente **C**liente de **c**liente).

La anotación @Id es para el campo de la clave primaria y @GeneratedValue define el tipo de dato que será el valor generado.

La anotación @Column mapea las columnas de la base de datos a las variables que se manejarán en el código.

Se usan constructores; uno vacío para instanciar la clase, y otro para crear clientes nuevos desde el código.

Se crean getters y setters para acceder y modificar los atributos del objeto.

Entonces esta clase representa una tabla llamada Cliente con los campos ID, nombre, email, teléfono y fecha de registro.

restDemo / rest /src / main / java / com / academy / rest / repository / ClienteRepository.java

Este archivo es para un repositorio en Spring Data JPA llamado ClienteRepository.

La anotación @Repository marca el archivo como un repositorio, @Param mapea parámetros de los métodos con los de las consultas, y @Query sirve para hacer consultas SQL, solo que aquí se llaman JPQL.

restDemo / rest /src / main / java / com / academy / rest / service / ClienteServiceImpl.java

Este archivo contiene la lógica de negocio y es el servicio principal. La anotación @Service es la que especifica que es un servicio.

@Autowired como se comentó antes, hace inyección de dependencias; para este ejercicio una instancia de ClientRepository.

Tiene métodos para obtener todos los clientes, obtener un cliente por ID, crear un nuevo cliente, actualizar un cliente existente, eliminar cliente, buscar cliente por email, buscar clientes por nombre (coincidencia parcial), buscar por teléfono, obtener todos los clientes ordenados de manera descendente por fecha de registro.

restDemo / rest /src / main / java / com / academy / rest / service / IClienteService.java

Este archivo es de una interfaz. Devuelve todos los clientes como una lista, busca un cliente por su ID, crea un nuevo cliente, actualiza un cliente existente con nuevos datos, elimina un cliente por su ID, busca un cliente por su correo electrónico, busca clientes cuyo nombre contenga el texto dado, busca un cliente por su teléfono, devuelve todos los clientes ordenados por la fecha en que se registraron.

restDemo / rest /src / main / java / com / academy / rest /RestApplication.java

Este archivo es el punto de entrada. El public class RestApplication ejecuta el método main, y la anotación @SpringBootApplication combina las tres anotaciones @Configuration (marca la clase como fuente de configuración para Spring), @EnableAutoConfiguration (activa la configuración automática de Spring Boot), y @ComponentScan (le dice a Spring que busque componentes como @Controller, @Service, @Repository, dentro del mismo paquete y sub-paquetes).

restDemo / rest /src / test / java / com / academy / rest /RestApplicationTests.java

Este archivo es una clase de prueba creado automáticamente al iniciar un proyecto de Spring Boot. Sirve para verificar que el contexto de la aplicación carga correctamente.