Instrucciones

El objetivo de esta actividad es conocer el entendimiento que el participante tiene hasta el momento sobre Spring Boot.

- *¿Cuál es el objetivo principal de Spring Boot? Menciona una buena práctica de diseño (ejem: patrón de diseño) incluida dentro de Spring Boot.*

El objetivo principal de spring boot es facilitar el desarrollo con spring permitiendo al programador no tener que configurar nada. Una buena práctica incluida en spring seria la inyección de dependencias, para ello Utilizamos la anotación @Autowired, crea las instancias de las clases que otro objeto necesita y lo suministra para que una clase los pueda utilizar. De esa forma no tenemos que estar instanciando.

```java
@RestController
@RequestMapping(value = "/kardex")
@Log4j2
public class KardexController {
    @Autowired
    private KardexService kardexService;

    @DeleteMapping("/{id}")
    public void deleteKardex(@PathVariable (value = "id") Long id){
         kardexService.deleteKardex(id);
    }
```

La anotación @Autowired se utiliza para inyectar una instancia de la clase KardexService en el controlador. Esto significa que el controlador depende del servicio y puede utilizar sus métodos y funcionalidades, por ejemplo deleteKardex().

- *Explica de forma breve qué es una anotación en Spring y cuál es su función.*
  Opcional: Explica el mecanismo detrás de la anotación @Data

Son etiquetas especiales que se aplican mediante el uso del símbolo "@" a los elementos del código para proporcionar información adicional. Su función principal es la configuración del comportamiento de Spring y sus componentes, ya que, al aplicar una anotación, le estamos proporcionando instrucciones específicas a Spring sobre cómo debe comportarse ese elemento.

```java
import jakarta.persistence.*;

@Entity
@Table(name ="kardex")
@Data
@NoArgsConstructor
public class Kardex {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long id;
    @Column(name ="folio")
    private String folioKardex;
    @Column(name = "materia_id", nullable = false, precision = 30)
    private Long materia;
    @Column(name = "calificacion", nullable = false, precision = 6)
    private Double calificacion;
    @ManyToOne
    @JoinColumn( name = "alumno_id")
    private Alumno alumno;


}
```

La anotación "@Data" es parte de Lombok. Nos ayuda a reducir la cantidad de código relacionado con los métodos getters, setters, equals y toString de nuestras entidades. Evitando así la necesidad de escribirlos manualmente. Esto simplifica el código y mejora la legibilidad.

Con @Data

```java
package edu.uady.academia.entity;

import jakarta.persistence.*;

@Entity
@Table (name = "alumnos")
@Data
@NoArgsConstructor
public class Alumno {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String nombre;
    private String apellidos;
    private int edad;
    private String sexo;
    private boolean documentacionCompleta;
}
```

Sin @Data

```java
@Entity
@Table (name = "alumnos")
@NoArgsConstructor
public class Alumno {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String nombre;
    private String apellidos;
    private int edad;
    private String sexo;
    private boolean documentacionCompleta;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getApellidos() {
        return apellidos;
    }
    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }
    public int getEdad() {
        return edad;
    }
```

- *Explica de forma clara cuál es la intención de cada paquete incluido en el proyecto: Entity, Controller, Repository, Service, DTO*

Entity: Son clases que representa entidades o modelos de datos de la aplicación.

Controller: son clases donde tenemos nuestros endpoints, se encargan de manejar las solicitudes HTTP entrantes, aquí tenemos los métodos get, post, put y delete. Estas clases se encargan de recibir los datos del cliente, interactuar con los servicios correspondientes y devolver las respuestas adecuadas.

Repository: Contiene las clases que se encarga de hacer todas las conexiones a la base de datos.

Service: Contiene las clases que implementan la lógica de negocio de la aplicación

DTO: Tiene las clases DTO de nuestras entidades donde nosotros vamos a poner solamente la información que vamos a necesitar o que queremos exponer, ya que no debemos exponer las entidades de la base de datos.

- *¿Explica el mecanismo básico para construir un endpoint con Spring boot? ¿Cuál es el mínimo de clases/interfaces/anotaciones involucradas y cómo están relacionadas entre ellas?*

Para hacer un endpoint básico como mínimo se necesita una clase controladora (con un @RestController de anotación) y un método dentro de la clase controladora. Como otra anotación indispensable podemos usar el @RequestMapping que nos define la ruta que el endpoint manejará y sirve tanto para la clase como para el método o al igual podemos utilizar anotaciones de métodos de HTTP para los métodos.

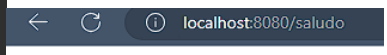
```java
package edu.uady.academia.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;


@RestController
public class RestTest {

    @GetMapping(value= "/saludo")
    public String saludo() {
        return "Hola mundo";
    }

}
```



Hola mundo

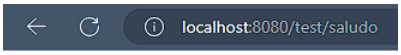
```java
package edu.uady.academia.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


@RestController
@RequestMapping(value="/test")
public class RestTest {

    @GetMapping(value= "/saludo")
    public String saludo() {
        return "Hola mundo";
    }

}
```



Hola mundo

Creo que este sería el endpoint más básico, ya que después fuimos usando en una clase controladora servicios que tienen la lógica del sistema, los cuales se conectan a una interfaz repositorio que tiene la conexión a la base de datos y claro igual tenemos una clase entidad.