UD4. Servicio RESTful

Resultado de Aprendizaje (RA)	RA4, RA5
Horas Previstas	20 h

1. Creación de la BD y configuración de ésta en Spring

- Gestor de BD

  create database prueba;
  create user 'user' identified by '1234';
  grant all on prueba. to 'user';*

- Proyecto Spring (AplicationProporties)

  spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/db_example
  spring.datasource.username=user
  spring.datasource.password=1234
  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

  spring.jpa.hibernate.ddl-auto=update
  #spring.jpa.show-sql: true

2. Gestión de dependencias

- Todas las dependencias que utiliza el proyecto se encuentran en el archivo pom.xml.
- En el caso de Maven se pueden ver las dependencias a instalar en la página de Maven Repository *https://mvnrepository.com/*
- Si se importa Spring Security y no se realiza la configuración de este, se puede excluir la clase encargada de su gestión en el arranque de la aplicación mediante la siguiente línea de código *@SpringBootApplication(exclude={SecurityAutoConfiguration.class})* en la cabecera de la clase con el main.
- En este proyecto se implementará Swagger que es una herramienta utilizada para documentar API's, que permitirá ver esta en la url *http://localhost:8080/swagger-ui/index.html*. Para añadir *Swagger al proyecto se debe poner en pom.xml:*

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.0.4</version>
</dependency>
```

3. Camino seguido por la información

Servidor (Repositorio + Entidades > Servicio > Controlador) > Cliente

4. Elementos de un proyecto Spring Boot

- Repositorio (Interface)

- Usa la etiqueta *@Repository*
- Extiende de *CrudRepository<Objeto, Integer>*
- Cuenta con las acciones de Create, Read, Update, Delete (por defecto) más todas las query que se quieran definir con *@Query.*
- JPA ofrece una seria de consultas por defecto que se pueden consultar en *https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html*

- Entidades (Clase)
  - Usa la etiqueta *@Entity* que indica que se creará una tabla en la BD.
  - Van a extender de DomainEntity (clase abstracta y de creación propia) para implementar los atributos id y versión en todas las entidades del proyecto.
    - Id usa la *etiqueta @id* y versión la e*tiqueta @version*
    - Usa la etiqueta *@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)* lo que hará que la clase DomainEntity no se cree en la BD, pero si sus atributos en cada una de las clases que extiendan de DomainEntity.

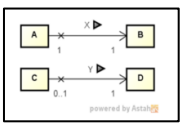
- Service (Clase)
  - Usa la etiqueta *@Service* que indica que se se trata de un servicio, en el que se implementa toda la lógica de negocio de la aplicación.
  - Se suele utilizar en el la etiqueta @Autowired para importar otros servicios o repositorios.

- Controlador (Clase)
  - Usa la etiqueta @RestController para indicar que se trata de un controlador.
  - Con las siguientes etiquetas se indica la ruta del controlador
    - @RequestMapping("/usuarios")
    - @GetMapping("/view")
    - @DeteleMapping("/delete") /
    - @PostMapping("/create")
    - @PutMapping("/delete")

5. Hibernate + JPA

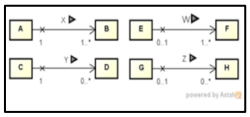
Hibernate es un framework de mapeo objeto-relacional (ORM) para Java, mientras que JPA (Java Persistence API) es una especificación estándar de este lenguaje de programación que permite el mapeo objeto-relacional (Hibernate implementa esta especificación).

- Relaciones
  - OneToMany



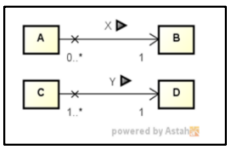

```
@OneToOne(optional=false)
public B getB() {
  return b;
}
```

  - OneToMany



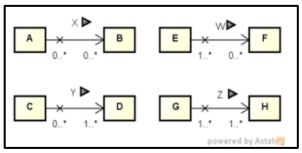

```
@OneToMany
public Collection<B> getBs() {
  return bs;
}
```

- ManyToOne



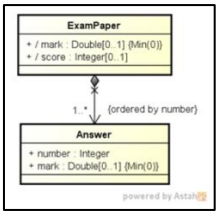

```
@ManyToOne(optional=false)
public B getB() {
  return b;
}
```

- ManyToMany



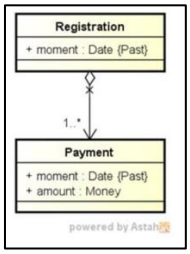

```
@ManyToMany
public Collection<B> getBs() {
  return bs;
}
```

- Composición



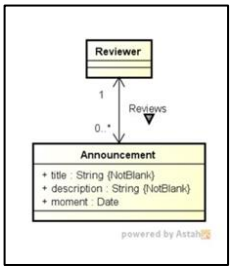

```
@OneToMany(cascade =
           CascadeType.ALL)
public Collection<Answer>
       getAnswers() {
  return answers;
}
```

- Agregación




```
@ManyToMany(cascade =
  {CascadeType.PERSIST,
   CascadeType.MERGE,
   CascadeType.DETACH,
   CascadeType.REFRESH})
public Collection<Payment>
       getPayments() {
  return payments;
}
```

- Bidireccionales




```
public class Announcement {
  …
  @ManyToOne(optional=false)
  public Reviewer
    getReviewer() {
      return reviewer;
  }
  …
}
```

```
public class Reviewer {
  …
  @OneToMany(mappedBy="reviewer")
  public Collection<Announcement>
  getAnnouncements() {
    return announcements;
  }
  …
}
```

- Otras etiquetas:
  - *@Column(unique = true)*: Se utiliza para indicar que el valor del atributo es único.
  - *@Table(uniqueConstraints=@UniqueConstraint(columnNames={"year"})):* Otra forma de realizar la misma restricción que la anterior.
  - *@NotBlank:* Atributo no puede estar en blanco.
  - *@NotNull:* Atributo no puede valer null.
  - *@URL:* El valor del atributo debe ser una url.
  - *@Email:* El valor del atributo debe ser email.
  - *@Max(n): El valor deberá ser menor o igual a n.*

- *@Min(n): El valor deberá ser mayor o igual a n.*
- *@Pattern(regexp = "r")*: El valor del atributo debe cumplir con el patrón establecido.
- *@Digits(fraction = n, integer = m):* Especifica el nº máximo de cifras enteras y decimales.
- *@Size(min = n, max = m):* Longitud max y min del atributo.