



Objetivos de la sesión:

- **Practicar** como realizar un nuevo **proyecto REST desde cero** con acceso a una base de datos.
- Aprender a realizar **paginaciones con filtrados y ordenaciones en Spring** para devolver solo un subconjunto del total de los datos.



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

Hasta ahora solo hemos utilizado tablas con pocos datos, pero si tenemos una tabla que tiene cientos de registros no es lógico mostrarlos todos cuando realizamos un listado.



¿ Como podemos mostrar solo un subconjunto de estos datos?



Para esta práctica vamos a crear otro proyecto REST CRUD sobre una BD. Sobre este nuevo proyecto practicaremos los nuevos conceptos:

Recordemos los pasos a seguir para crear un API RESTful:

- 1º Crear el proyecto Spring Boot**
- 2º Configurar la Base de Datos**
- 3º Crear las entidades JPA**
- 4º Inicializar con datos la BD**
- 5º Crear el repositorio para cada entidad JPA**
- 6º Crear la capa de servicios**
- 7º Crear la capa de control (controller)**
- 8º Pruebas**



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

1º Crear el proyecto Spring Boot:

Para empezar vamos a crear nuestro proyecto REST llamado **“dwes_futbol_rest”**. Le damos este nombre porque en una futura práctica ampliaremos la funcionalidad de este proyecto.

Lo hacemos igual como en “dwes_primer_rest”:

Java Projects\Añadir (+) \Spring Boot \Proyecto Maven \versión 2.6.3 \Java

e introducimos los siguientes datos:

Group Id: edu.alumno.NombreAlumno

Artifact Id: dwes_futbol_rest

Package: war

Versión de Java: 11

Dependencias:

Deberemos seleccionar las mismas dependencias del otro proyecto:

“Spring Web”, “H2 Database”, “Spring Data JPA”, “Lombok” y añadir “DevTools” y “Validation” y le damos al intro.



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

... continuación 1º Crear el proyecto Spring Boot:

Para continuar deberemos de copiar la dependencia de MapStruct y plugins como el compilador de maven. Lo más cómodo es que vayamos al proyecto “dwes_primer_rest” y copiemos de su fichero pom.xml de properties hacia abajo:

```
pom.xml 1 x
dwes_futbol_rest > pom.xml > ...
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4    <modelVersion>4.0.0</modelVersion>
5    <parent>
6      <groupId>org.springframework.boot</groupId>
7      <artifactId>spring-boot-starter-parent</artifactId>
8      <version>2.6.3</version>
9      <relativePath/> <!-- lookup parent from repository -->
10   </parent>
11   <groupId>edu.profesor.joseramon</groupId>
12   <artifactId>dwes_futbol_rest</artifactId>
13   <version>0.0.1-SNAPSHOT</version>
14   <packaging>war</packaging>
15   <name>dwes_futbol_rest</name>
16   <description>Demo project for Spring Boot</description>
17   <properties>
18     <java.version>1.8</java.version>
19     <org.projectlombok.version>1.18.16</org.projectlombok.version>
20     <org.mapstruct.version>1.4.1.Final</org.mapstruct.version>
21   </properties>
22   <dependencies>
23     <dependency>
24       <groupId>org.springframework.boot</groupId>
25       <artifactId>spring-boot-starter-data-jpa</artifactId>
26     </dependency>
27     <dependency>
28       <groupId>org.springframework.boot</groupId>
```




UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

2º Configurar la Base de Datos:

Podemos copiar y pegar de la práctica “dwes_primer_rest” la configuración de la base de datos en “application.properties”:

```
application.properties X
src > main > resources > application.properties
1 #Activar consola para acceder a la BD H2 via navegador
2 # localhost:puertoConfigurado/h2-console
3 spring.h2.console.enabled=true
4 #Configuración de la BD H2
5 spring.datasource.url=jdbc:h2:mem:testDwesDb
6 spring.datasource.driverClassName=org.h2.Driver
7 spring.datasource.username=dwes
8 spring.datasource.password=Simarro@1
9 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
10
11 #En la versión 2.4.2 no hace falta, pero en la
12 # 2.6.2 hace falta para poder hacer inserts en data.sql
13 spring.jpa.hibernate.ddl-auto=none
14
15 #CONFIGURACIÓN SOLO durante las pruebas:
16 # Habilitar estadísticas hibernate
17 spring.jpa.properties.hibernate.generate_statistics=true
18 # Habilitar LOGGER de las estadísticas de hibernate
19 logging.level.org.hibernate.stat=
20 # Mostrar que consultas esta realizando Hibernate
21 spring.jpa.show-sql=true
22 # Formatear las consultas
23 spring.jpa.properties.hibernate.format_sql=true
24 # Mostrar los parametros que estan enviandose a las consultas
25 logging.level.org.hibernate.type=debug
26 #FIN CONFIGURACIÓN SOLO durante las pruebas
```



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

3º Crear las entidades JPA:

Empezaremos de momento por crear la entidad “CiudadDb.java” :

```
CiudadDb.java x
dwes_futbol_rest > src > main > java > edu > profesor > joseramon > dwes_futbol_rest > model > db > CiudadDb.java > ...
1 package edu.profesor.joseramon.dwes_futbol_rest.model.db;
2
3 > import java.io.Serializable; ...
14
15 @NoArgsConstructor
16 @AllArgsConstructor
17 @Data
18 @Entity
19 @Table(name = "ciudades")
20 public class CiudadDb implements Serializable{
21     @Id
22     @GeneratedValue(strategy = GenerationType.IDENTITY)
23     private Long id;
24     @Size(min=4,message="El nombre debe de tener un tamaño mínimo de 4 caracteres")
25     private String nombre;
26     @Column(nullable = true)
27     private Long habitantes;
28 }
```



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

4º Inicializar con datos la BD:

Al igual que el proyecto “dwes_primer_rest”, para inicializar la Base de Datos H2 hay que crear el fichero “data.sql” en la carpeta “src\main\resources” y copiar las sentencias siguientes que también están en el DRIVE (data.sql):

```
data.sql x
src > main > resources > data.sql
1  -- Copyright Jose Ramón Cebolla - 2022
2  -----
3  -- Table `ciudades`
4  -----
5  DROP TABLE IF EXISTS ciudades;
6  CREATE TABLE IF NOT EXISTS ciudades(
7  id IDENTITY,
8  nombre VARCHAR(50),
9  habitantes BIGINT,
10 CONSTRAINT pk_ciudades PRIMARY KEY(id));
11 -----
12 -- INSERCIÓN DE DATOS
13 -----
14 -----
15 -----
16 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (2,'Alacant',NULL);
17 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (3,'Albacete',NULL);
18 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (4,'Almeria',NULL);
19 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (5,'Astúries',NULL);
20 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (6,'Ávila',NULL);
21 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (7,'Badajoz',NULL);
22 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (8,'Barcelona',1615000);
23 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (9,'Bilbao',353000);
24 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (10,'Biscaia',NULL);
25 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (11,'Burgos',NULL);
26 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (12,'Càceres',NULL);
27 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (13,'Cadis',NULL);
28 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (14,'Cantàbria',NULL);
29 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (15,'Castelló',NULL);
30 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (16,'Ceuta',NULL);
31 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (17,'ciudad Real',NULL);
32 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (18,'Còrdova',NULL);
33 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (19,'Cornellà de Llobregat',85000);
34 INSERT INTO `ciudades` (`id`,`nombre`,`habitantes`) VALUES (20,'Girona',NULL);
```




UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

5º Crear el repositorio para cada entidad JPA:

Para hacer **CRUD (Create Read Update Delete)** sobre la tabla “**Ciudades**” volvemos a utilizar la interface **JpaRepository** de Spring y si queremos poder ordenar los resultados de un listado añadiremos el método con la clase **Sort**:

```
CiudadRepository.java X
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > repository > CiudadRepository.java >
1  package edu.profesor.joseramon.dwes_futbol_rest.repository;
2
3  import java.util.List;
4
5  import org.springframework.data.domain.Sort;
6  import org.springframework.data.jpa.repository.JpaRepository;
7  import org.springframework.stereotype.Repository;
8  import edu.profesor.joseramon.dwes_futbol_rest.model.db.CiudadDb;
9
10 @Repository
11 public interface CiudadRepository extends JpaRepository<CiudadDb,Long>{
12     //Métodos automáticos en Spring Data JPA
13     List<CiudadDb> findAll(Sort sort);
14 }
```



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

6º Crear la capa de servicios:

Para crear la capa de servicios vamos a crear primero una interface con los servicios que se ofrecerán y luego implementaremos una clase que rellene dichos métodos de la interface. De momento solo queremos poder consultar un dato o el listado ordenado o sin ordenar:

```
CiudadService.java X
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > srv > CiudadService.java > ...
1  package edu.profesor.joseramon.dwes_futbol_rest.srv;
2
3  import java.util.List;
4  import java.util.Optional;
5
6  import org.springframework.data.domain.Sort;
7
8  import edu.profesor.joseramon.dwes_futbol_rest.model.dto.CiudadInfo;
9  import edu.profesor.joseramon.dwes_futbol_rest.model.dto.CiudadList;
10
11 public interface CiudadService {
12     public Optional<CiudadInfo> getCiudadInfoById(Long id);
13     public List<CiudadList> findAllCiudadList();
14     public List<CiudadList> findAllCiudadList(Sort sort);
15 }
```



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

... continuación 6º Crear la capa de servicios:

Mappers:

Para poder realizar la implementación del servicio necesitaremos configurar CiudadMapper.java:

```
CiudadMapper.java X
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > srv > mapper > CiudadMapper.java > ...
1  package edu.profesor.joseramon.dwes_futbol_rest.srv.mapper;
2
3  import java.util.List;
4
5  import org.mapstruct.Mapper;
6  import org.mapstruct.factory.Mappers;
7
8  import edu.profesor.joseramon.dwes_futbol_rest.model.db.CiudadDb;
9  import edu.profesor.joseramon.dwes_futbol_rest.model.dto.CiudadInfo;
10 import edu.profesor.joseramon.dwes_futbol_rest.model.dto.CiudadList;
11
12 @Mapper
13 public interface CiudadMapper {
14     CiudadMapper INSTANCE= Mappers.getMapper(CiudadMapper.class);
15
16     CiudadInfo CiudadDbToCiudadInfo(CiudadDb ciudadDb);
17     CiudadList CiudadDbToCiudadList(CiudadDb ciudadDb);
18     List<CiudadList> ciudadesToCiudadList(List<CiudadDb> ciudadesDb);
19 }
```

NOTA: No se han implementado todos los métodos (guardar, modificar,...) porque en esta practica se pretende analizar como se puede realizar consultas devolviendo un subconjunto de los datos.



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

... continuación 6º Crear la capa de servicios:

Y a continuación implementaremos los métodos del servicio:

```
CiudadServiceImpl.java x
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > srv > impl > CiudadServiceImpl.java > ...
1 package edu.profesor.joseramon.dwes_futbol_rest.srv.impl;
2
3 import java.util.List;
4 import java.util.Optional;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.data.domain.Sort;
7 import org.springframework.stereotype.Service;
8 import edu.profesor.joseramon.dwes_futbol_rest.model.db.CiudadDb;
9 import edu.profesor.joseramon.dwes_futbol_rest.model.dto.CiudadInfo;
10 import edu.profesor.joseramon.dwes_futbol_rest.model.dto.CiudadList;
11 import edu.profesor.joseramon.dwes_futbol_rest.repository.CiudadRepository;
12 import edu.profesor.joseramon.dwes_futbol_rest.srv.CiudadService;
13 import edu.profesor.joseramon.dwes_futbol_rest.srv.mapper.CiudadMapper;
14
15 @Service
16 public class CiudadServiceImpl implements CiudadService{
17     private final CiudadRepository ciudadRepository;
18
19     @Autowired
20     public CiudadServiceImpl(CiudadRepository ciudadRepository){
21         this.ciudadRepository=ciudadRepository;
22     }
23
24     public List<CiudadList> findAllCiudadList(){
25         return CiudadMapper.INSTANCE.ciudadesToCiudadList(ciudadRepository.findAll());
26     }
27
28     public List<CiudadList> findAllCiudadList(Sort sort){
29         return CiudadMapper.INSTANCE.ciudadesToCiudadList(ciudadRepository.findAll(sort));
30     }
31
32     public Optional<CiudadInfo> getCiudadInfoById( Long id){
33         Optional<CiudadDb> ciudadDb=ciudadRepository.findById(id);
34         if (ciudadDb.isPresent()){
35             return Optional.of(CiudadMapper.INSTANCE.CiudadDbToCiudadInfo(ciudadDb.get()));
36         }else{
37             return Optional.empty();
38         }
39     }
40 }
41 }
```




UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

7º Crear la capa de control (controller):

Ahora implementamos los métodos de consulta del controlador:

```
CiudadRestController.java x
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > controller > CiudadRestController.java > ...
1  package edu.profesor.joseramon.dwes_futbol_rest.controller;
2
3  > import java.util.Collection; ...
19
20  @RestController
21  @RequestMapping("/api/v1/")
22  public class CiudadRestController {
23
24      private CiudadService ciudadService;
25
26      @Autowired
27      public CiudadRestController(CiudadService ciudadService) {
28          this.ciudadService=ciudadService;
29      }
30
31      @GetMapping("/ciudades")
32      public Collection<CiudadList> getCiudadesList() {
33          return ciudadService.findAllCiudadList();
34      }
35
36      @GetMapping("/ciudades/orden/{direccionOrden}")
37      public Collection<CiudadList> getCiudadesListOrderByNombre(
38          @PathVariable("direccionOrden") String direccionOrden) {
39          return ciudadService.findAllCiudadList(Sort.by(Direction.fromString(direccionOrden), "nombre"));
40      }
41
42      @GetMapping("/ciudades/{id}/info")
43      public ResponseEntity<CiudadInfo> getCiudadInfoById(
44          @PathVariable(value = "id") Long id) throws RuntimeException {
45          Optional<CiudadInfo> alumnoInfo = ciudadService.getCiudadInfoById(id);
46          if (alumnoInfo.isPresent())
47              return ResponseEntity.ok().body(alumnoInfo.get());
48          else throw new ResourceNotFoundException("Ciudad not found on :: "+ id);
49      }
50  }
```



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

8º Pruebas:

Si realizamos pruebas vemos que funciona: (Fijate que 'Àvila' tiene acento)

```
test_Rest.http x
test_Rest.http > ...
1  ## getCiudadesList
   Send Request
2  GET http://localhost:8080/api/v1/ciudades HTTP/1.1
3  Content-Type: application/json
4  ###
5  ## getCiudadesListOrderByNombre
   Send Request
6  GET http://localhost:8080/api/v1/ciudades/orden/desc HTTP/1.1
7  Content-Type: application/json
8  ###
9  ## getCiudadInfoById
   Send Request
10 GET http://localhost:8080/api/v1/ciudades/56/info HTTP/1.1
11 Content-Type: application/json
12 ###
```

```
Response(27ms) x
1 HTTP/1.1 200
2 Content-Type: application/json
3 Transfer-Encoding: chunked
4 Date: Thu, 03 Feb 2022 10:04:03 GMT
5 Connection: close
6
7 √ [
8 √ {
9   "id": 6,
10  "nombre": "Àvila",
11  "habitantes": null
12 },
13 √ {
14   "id": 59,
15   "nombre": "Zamora",
16   "habitantes": null
17 },
18 √ {
19   "id": 58,
20   "nombre": "Vigo",
21   "habitantes": 297000
22 },
23 √ {
24   "id": 56,
25   "nombre": "València",
26   "habitantes": 798000
27 },
28 √ {
29   "id": 57,
30   "nombre": "Valladolid",
31   "habitantes": 313000
32 },
33 √ {
34   "id": 55,
35   "nombre": "Toledo",
36   "habitantes": null
37 }
```

```
Response(268ms) x
1 HTTP/1.1 200
2 Content-Type: application/json
3 Transfer-Encoding: chunked
4 Date: Thu, 03 Feb 2022 09:32:02 GMT
5 Connection: close
6
7 √ [
8 √ {
9   "id": 2,
10  "nombre": "Alacant",
11  "habitantes": null
12 },
13 √ {
14   "id": 3,
15   "nombre": "Albacete",
16   "habitantes": null
17 },
18 √ {
19   "id": 4,
20   "nombre": "Almeria",
21   "habitantes": null
22 },
23 √ {
24   "id": 5,
25   "nombre": "Astúries",
26   "habitantes": null
27 },
28 √ {
29   "id": 6,
30   "nombre": "Àvila",
31   "habitantes": null
32 },
33 √ {
34   "id": 7,
35   "nombre": "Badajoz",
36   "habitantes": null
37 },
38 √ {
39   "id": 8,
40   "nombre": "Baleares",
41   "habitantes": null
42 },
43 √ {
44   "id": 9,
45   "nombre": "Barcelona",
46   "habitantes": null
47 },
48 √ {
49   "id": 10,
50   "nombre": "Burgos",
51   "habitantes": null
52 },
53 √ {
54   "id": 11,
55   "nombre": "Caceres",
56   "habitantes": null
57 },
58 √ {
59   "id": 12,
60   "nombre": "Cádiz",
61   "habitantes": null
62 },
63 √ {
64   "id": 13,
65   "nombre": "Castellón",
66   "habitantes": null
67 },
68 √ {
69   "id": 14,
70   "nombre": "Cataluña",
71   "habitantes": null
72 },
73 √ {
74   "id": 15,
75   "nombre": "Ceuta",
76   "habitantes": null
77 },
78 √ {
79   "id": 16,
80   "nombre": "Córdoba",
81   "habitantes": null
82 },
83 √ {
84   "id": 17,
85   "nombre": "Cuenca",
86   "habitantes": null
87 },
88 √ {
89   "id": 18,
90   "nombre": "García",
91   "habitantes": null
92 },
93 √ {
94   "id": 19,
95   "nombre": "Girona",
96   "habitantes": null
97 },
98 √ {
99   "id": 20,
100  "nombre": "Granada",
101  "habitantes": null
102 },
103 √ {
104   "id": 21,
105   "nombre": "Guadalajara",
106   "habitantes": null
107 },
108 √ {
109   "id": 22,
110   "nombre": "Huelva",
111   "habitantes": null
112 },
113 √ {
114   "id": 23,
115   "nombre": "Huesca",
116   "habitantes": null
117 },
118 √ {
119   "id": 24,
120   "nombre": "Ibiza",
121   "habitantes": null
122 },
123 √ {
124   "id": 25,
125   "nombre": "Jaén",
126   "habitantes": null
127 },
128 √ {
129   "id": 26,
130   "nombre": "Lérida",
131   "habitantes": null
132 },
133 √ {
134   "id": 27,
135   "nombre": "Lugo",
136   "habitantes": null
137 },
138 √ {
139   "id": 28,
140   "nombre": "Madrid",
141   "habitantes": null
142 },
143 √ {
144   "id": 29,
145   "nombre": "Málaga",
146   "habitantes": null
147 },
148 √ {
149   "id": 30,
150   "nombre": "Marbella",
151   "habitantes": null
152 },
153 √ {
154   "id": 31,
155   "nombre": "Melilla",
156   "habitantes": null
157 },
158 √ {
159   "id": 32,
160   "nombre": "Murcia",
161   "habitantes": null
162 },
163 √ {
164   "id": 33,
165   "nombre": "Navarra",
166   "habitantes": null
167 },
168 √ {
169   "id": 34,
170   "nombre": "Ourense",
171   "habitantes": null
172 },
173 √ {
174   "id": 35,
175   "nombre": "Palencia",
176   "habitantes": null
177 },
178 √ {
179   "id": 36,
180   "nombre": "Pamplona",
181   "habitantes": null
182 },
183 √ {
184   "id": 37,
185   "nombre": "Pontevedra",
186   "habitantes": null
187 },
188 √ {
189   "id": 38,
190   "nombre": "Salamanca",
191   "habitantes": null
192 },
193 √ {
194   "id": 39,
195   "nombre": "San Sebastián",
196   "habitantes": null
197 },
198 √ {
199   "id": 40,
200   "nombre": "Segovia",
201   "habitantes": null
202 },
203 √ {
204   "id": 41,
205   "nombre": "Sevilla",
206   "habitantes": null
207 },
208 √ {
209   "id": 42,
210   "nombre": "Soria",
211   "habitantes": null
212 },
213 √ {
214   "id": 43,
215   "nombre": "Tarragona",
216   "habitantes": null
217 },
218 √ {
219   "id": 44,
220   "nombre": "Tenerife",
221   "habitantes": null
222 },
223 √ {
224   "id": 45,
225   "nombre": "Teruel",
226   "habitantes": null
227 },
228 √ {
229   "id": 46,
230   "nombre": "Tlaxcala",
231   "habitantes": null
232 },
233 √ {
234   "id": 47,
235   "nombre": "Torreón",
236   "habitantes": null
237 },
238 √ {
239   "id": 48,
240   "nombre": "Tortosa",
241   "habitantes": null
242 },
243 √ {
244   "id": 49,
245   "nombre": "Trujillo",
246   "habitantes": null
247 },
248 √ {
249   "id": 50,
250   "nombre": "Valencia",
251   "habitantes": null
252 },
253 √ {
254   "id": 51,
255   "nombre": "Valladolid",
256   "habitantes": null
257 },
258 √ {
259   "id": 52,
260   "nombre": "Vizcaya",
261   "habitantes": null
262 },
263 √ {
264   "id": 53,
265   "nombre": "Vitoria",
266   "habitantes": null
267 },
268 √ {
269   "id": 54,
270   "nombre": "Vitoria",
271   "habitantes": null
272 },
273 √ {
274   "id": 55,
275   "nombre": "Vitoria",
276   "habitantes": null
277 },
278 √ {
279   "id": 56,
280   "nombre": "Vitoria",
281   "habitantes": null
282 },
283 √ {
284   "id": 57,
285   "nombre": "Vitoria",
286   "habitantes": null
287 },
288 √ {
289   "id": 58,
290   "nombre": "Vitoria",
291   "habitantes": null
292 },
293 √ {
294   "id": 59,
295   "nombre": "Vitoria",
296   "habitantes": null
297 },
298 √ {
299   "id": 60,
300   "nombre": "Vitoria",
301   "habitantes": null
302 },
303 √ {
304   "id": 61,
305   "nombre": "Vitoria",
306   "habitantes": null
307 },
308 √ {
309   "id": 62,
310   "nombre": "Vitoria",
311   "habitantes": null
312 },
313 √ {
314   "id": 63,
315   "nombre": "Vitoria",
316   "habitantes": null
317 },
318 √ {
319   "id": 64,
320   "nombre": "Vitoria",
321   "habitantes": null
322 },
323 √ {
324   "id": 65,
325   "nombre": "Vitoria",
326   "habitantes": null
327 },
328 √ {
329   "id": 66,
330   "nombre": "Vitoria",
331   "habitantes": null
332 },
333 √ {
334   "id": 67,
335   "nombre": "Vitoria",
336   "habitantes": null
337 },
338 √ {
339   "id": 68,
340   "nombre": "Vitoria",
341   "habitantes": null
342 },
343 √ {
344   "id": 69,
345   "nombre": "Vitoria",
346   "habitantes": null
347 },
348 √ {
349   "id": 70,
350   "nombre": "Vitoria",
351   "habitantes": null
352 },
353 √ {
354   "id": 71,
355   "nombre": "Vitoria",
356   "habitantes": null
357 },
358 √ {
359   "id": 72,
360   "nombre": "Vitoria",
361   "habitantes": null
362 },
363 √ {
364   "id": 73,
365   "nombre": "Vitoria",
366   "habitantes": null
367 },
368 √ {
369   "id": 74,
370   "nombre": "Vitoria",
371   "habitantes": null
372 },
373 √ {
374   "id": 75,
375   "nombre": "Vitoria",
376   "habitantes": null
377 },
378 √ {
379   "id": 76,
380   "nombre": "Vitoria",
381   "habitantes": null
382 },
383 √ {
384   "id": 77,
385   "nombre": "Vitoria",
386   "habitantes": null
387 },
388 √ {
389   "id": 78,
390   "nombre": "Vitoria",
391   "habitantes": null
392 },
393 √ {
394   "id": 79,
395   "nombre": "Vitoria",
396   "habitantes": null
397 },
398 √ {
399   "id": 80,
400   "nombre": "Vitoria",
401   "habitantes": null
402 },
403 √ {
404   "id": 81,
405   "nombre": "Vitoria",
406   "habitantes": null
407 },
408 √ {
409   "id": 82,
410   "nombre": "Vitoria",
411   "habitantes": null
412 },
413 √ {
414   "id": 83,
415   "nombre": "Vitoria",
416   "habitantes": null
417 },
418 √ {
419   "id": 84,
420   "nombre": "Vitoria",
421   "habitantes": null
422 },
423 √ {
424   "id": 85,
425   "nombre": "Vitoria",
426   "habitantes": null
427 },
428 √ {
429   "id": 86,
430   "nombre": "Vitoria",
431   "habitantes": null
432 },
433 √ {
434   "id": 87,
435   "nombre": "Vitoria",
436   "habitantes": null
437 },
438 √ {
439   "id": 88,
440   "nombre": "Vitoria",
441   "habitantes": null
442 },
443 √ {
444   "id": 89,
445   "nombre": "Vitoria",
446   "habitantes": null
447 },
448 √ {
449   "id": 90,
450   "nombre": "Vitoria",
451   "habitantes": null
452 },
453 √ {
454   "id": 91,
455   "nombre": "Vitoria",
456   "habitantes": null
457 },
458 √ {
459   "id": 92,
460   "nombre": "Vitoria",
461   "habitantes": null
462 },
463 √ {
464   "id": 93,
465   "nombre": "Vitoria",
466   "habitantes": null
467 },
468 √ {
469   "id": 94,
470   "nombre": "Vitoria",
471   "habitantes": null
472 },
473 √ {
474   "id": 95,
475   "nombre": "Vitoria",
476   "habitantes": null
477 },
478 √ {
479   "id": 96,
480   "nombre": "Vitoria",
481   "habitantes": null
482 },
483 √ {
484   "id": 97,
485   "nombre": "Vitoria",
486   "habitantes": null
487 },
488 √ {
489   "id": 98,
490   "nombre": "Vitoria",
491   "habitantes": null
492 },
493 √ {
494   "id": 99,
495   "nombre": "Vitoria",
496   "habitantes": null
497 },
498 √ {
499   "id": 100,
500   "nombre": "Vitoria",
501   "habitantes": null
502 },
503 √ {
504   "id": 101,
505   "nombre": "Vitoria",
506   "habitantes": null
507 },
508 √ {
509   "id": 102,
510   "nombre": "Vitoria",
511   "habitantes": null
512 },
513 √ {
514   "id": 103,
515   "nombre": "Vitoria",
516   "habitantes": null
517 },
518 √ {
519   "id": 104,
520   "nombre": "Vitoria",
521   "habitantes": null
522 },
523 √ {
524   "id": 105,
525   "nombre": "Vitoria",
526   "habitantes": null
527 },
528 √ {
529   "id": 106,
530   "nombre": "Vitoria",
531   "habitantes": null
532 },
533 √ {
534   "id": 107,
535   "nombre": "Vitoria",
536   "habitantes": null
537 },
538 √ {
539   "id": 108,
540   "nombre": "Vitoria",
541   "habitantes": null
542 },
543 √ {
544   "id": 109,
545   "nombre": "Vitoria",
546   "habitantes": null
547 },
548 √ {
549   "id": 110,
550   "nombre": "Vitoria",
551   "habitantes": null
552 },
553 √ {
554   "id": 111,
555   "nombre": "Vitoria",
556   "habitantes": null
557 },
558 √ {
559   "id": 112,
560   "nombre": "Vitoria",
561   "habitantes": null
562 },
563 √ {
564   "id": 113,
565   "nombre": "Vitoria",
566   "habitantes": null
567 },
568 √ {
569   "id": 114,
570   "nombre": "Vitoria",
571   "habitantes": null
572 },
573 √ {
574   "id": 115,
575   "nombre": "Vitoria",
576   "habitantes": null
577 },
578 √ {
579   "id": 116,
580   "nombre": "Vitoria",
581   "habitantes": null
582 },
583 √ {
584   "id": 117,
585   "nombre": "Vitoria",
586   "habitantes": null
587 },
588 √ {
589   "id": 118,
590   "nombre": "Vitoria",
591   "habitantes": null
592 },
593 √ {
594   "id": 119,
595   "nombre": "Vitoria",
596   "habitantes": null
597 },
598 √ {
599   "id": 120,
600   "nombre": "Vitoria",
601   "habitantes": null
602 },
603 √ {
604   "id": 121,
605   "nombre": "Vitoria",
606   "habitantes": null
607 },
608 √ {
609   "id": 122,
610   "nombre": "Vitoria",
611   "habitantes": null
612 },
613 √ {
614   "id": 123,
615   "nombre": "Vitoria",
616   "habitantes": null
617 },
618 √ {
619   "id": 124,
620   "nombre": "Vitoria",
621   "habitantes": null
622 },
623 √ {
624   "id": 125,
625   "nombre": "Vitoria",
626   "habitantes": null
627 },
628 √ {
629   "id": 126,
630   "nombre": "Vitoria",
631   "habitantes": null
632 },
633 √ {
634   "id": 127,
635   "nombre": "Vitoria",
636   "habitantes": null
637 },
638 √ {
639   "id": 128,
640   "nombre": "Vitoria",
641   "habitantes": null
642 },
643 √ {
644   "id": 129,
645   "nombre": "Vitoria",
646   "habitantes": null
647 },
648 √ {
649   "id": 130,
650   "nombre": "Vitoria",
651   "habitantes": null
652 },
653 √ {
654   "id": 131,
655   "nombre": "Vitoria",
656   "habitantes": null
657 },
658 √ {
659   "id": 132,
660   "nombre": "Vitoria",
661   "habitantes": null
662 },
663 √ {
664   "id": 133,
665   "nombre": "Vitoria",
666   "habitantes": null
667 },
668 √ {
669   "id": 134,
670   "nombre": "Vitoria",
671   "habitantes": null
672 },
673 √ {
674   "id": 135,
675   "nombre": "Vitoria",
676   "habitantes": null
677 },
678 √ {
679   "id": 136,
680   "nombre": "Vitoria",
681   "habitantes": null
682 },
683 √ {
684   "id": 137,
685   "nombre": "Vitoria",
686   "habitantes": null
687 },
688 √ {
689   "id": 138,
690   "nombre": "Vitoria",
691   "habitantes": null
692 },
693 √ {
694   "id": 139,
695   "nombre": "Vitoria",
696   "habitantes": null
697 },
698 √ {
699   "id": 140,
700   "nombre": "Vitoria",
701   "habitantes": null
702 },
703 √ {
704   "id": 141,
705   "nombre": "Vitoria",
706   "habitantes": null
707 },
708 √ {
709   "id": 142,
710   "nombre": "Vitoria",
711   "habitantes": null
712 },
713 √ {
714   "id": 143,
715   "nombre": "Vitoria",
716   "habitantes": null
717 },
718 √ {
719   "id": 144,
720   "nombre": "Vitoria",
721   "habitantes": null
722 },
723 √ {
724   "id": 145,
725   "nombre": "Vitoria",
726   "habitantes": null
727 },
728 √ {
729   "id": 146,
730   "nombre": "Vitoria",
731   "habitantes": null
732 },
733 √ {
734   "id": 147,
735   "nombre": "Vitoria",
736   "habitantes": null
737 },
738 √ {
739   "id": 148,
740   "nombre": "Vitoria",
741   "habitantes": null
742 },
743 √ {
744   "id": 149,
745   "nombre": "Vitoria",
746   "habitantes": null
747 },
748 √ {
749   "id": 150,
750   "nombre": "Vitoria",
751   "habitantes": null
752 },
753 √ {
754   "id": 151,
755   "nombre": "Vitoria",
756   "habitantes": null
757 },
758 √ {
759   "id": 152,
760   "nombre": "Vitoria",
761   "habitantes": null
762 },
763 √ {
764   "id": 153,
765   "nombre": "Vitoria",
766   "habitantes": null
767 },
768 √ {
769   "id": 154,
770   "nombre": "Vitoria",
771   "habitantes": null
772 },
773 √ {
774   "id": 155,
775   "nombre": "Vitoria",
776   "habitantes": null
777 },
778 √ {
779   "id": 156,
780   "nombre": "Vitoria",
781   "habitantes": null
782 },
783 √ {
784   "id": 157,
785   "nombre": "Vitoria",
786   "habitantes": null
787 },
788 √ {
789   "id": 158,
790   "nombre": "Vitoria",
791   "habitantes": null
792 },
793 √ {
794   "id": 159,
795   "nombre": "Vitoria",
796   "habitantes": null
797 },
798 √ {
799   "id": 160,
800   "nombre": "Vitoria",
801   "habitantes": null
802 },
803 √ {
804   "id": 161,
805   "nombre": "Vitoria",
806   "habitantes": null
807 },
808 √ {
809   "id": 162,
810   "nombre": "Vitoria",
811   "habitantes": null
812 },
813 √ {
814   "id": 163,
815   "nombre": "Vitoria",
816   "habitantes": null
817 },
818 √ {
819   "id": 164,
820   "nombre": "Vitoria",
821   "habitantes": null
822 },
823 √ {
824   "id": 165,
825   "nombre": "Vitoria",
826   "habitantes": null
827 },
828 √ {
829   "id": 166,
830   "nombre": "Vitoria",
831   "habitantes": null
832 },
833 √ {
834   "id": 167,
835   "nombre": "Vitoria",
836   "habitantes": null
837 },
838 √ {
839   "id": 168,
840   "nombre": "Vitoria",
841   "habitantes": null
842 },
843 √ {
844   "id": 169,
845   "nombre": "Vitoria",
846   "habitantes": null
847 },
848 √ {
849   "id": 170,
850   "nombre": "Vitoria",
851   "habitantes": null
852 },
853 √ {
854   "id": 171,
855   "nombre": "Vitoria",
856   "habitantes": null
857 },
858 √ {
859   "id": 172,
860   "nombre": "Vitoria",
861   "habitantes": null
862 },
863 √ {
864   "id": 173,
865   "nombre": "Vitoria",
866   "habitantes": null
867 },
868 √ {
869   "id": 174,
870   "nombre": "Vitoria",
871   "habitantes": null
872 },
873 √ {
874   "id": 175,
875   "nombre": "Vitoria",
876   "habitantes": null
877 },
878 √ {
879   "id": 176,
880   "nombre": "Vitoria",
881   "habitantes": null
882 },
883 √ {
884   "id": 177,
885   "nombre": "Vitoria",
886   "habitantes": null
887 },
888 √ {
889   "id": 178,
890   "nombre": "Vitoria",
891   "habitantes": null
892 },
893 √ {
894   "id": 179,
895   "nombre": "Vitoria",
896   "habitantes": null
897 },
898 √ {
899   "id": 180,
900   "nombre": "Vitoria",
901   "habitantes": null
902 },
903 √ {
904   "id": 181,
905   "nombre": "Vitoria",
906   "habitantes": null
907 },
908 √ {
909   "id": 182,
910   "nombre": "Vitoria",
911   "habitantes": null
912 },
913 √ {
914   "id": 183,
915   "nombre": "Vitoria",
916   "habitantes": null
917 },
918 √ {
919   "id": 184,
920   "nombre": "Vitoria",
921   "habitantes": null
922 },
923 √ {
924   "id": 185,
925   "nombre": "Vitoria",
926   "habitantes": null
927 },
928 √ {
929   "id": 186,
930   "nombre": "Vitoria",
931   "habitantes": null
932 },
933 √ {
934   "id": 187,
935   "nombre": "Vitoria",
936   "habitantes": null
937 },
938 √ {
939   "id": 188,
940   "nombre": "Vitoria",
941   "habitantes": null
942 },
943 √ {
944   "id": 189,
945   "nombre": "Vitoria",
946   "habitantes": null
947 },
948 √ {
949   "id": 190,
950   "nombre": "Vitoria",
951   "habitantes": null
952 },
953 √ {
954   "id": 191,
955   "nombre": "Vitoria",
956   "habitantes": null
957 },
958 √ {
959   "id": 192,
960   "nombre": "Vitoria",
961   "habitantes": null
962 },
963 √ {
964   "id": 193,
965   "nombre": "Vitoria",
966   "habitantes": null
967 },
968 √ {
969   "id": 194,
970   "nombre": "Vitoria",
971   "habitantes": null
972 },
973 √ {
974   "id": 195,
975   "nombre": "Vitoria",
976   "habitantes": null
977 },
978 √ {
979   "id": 196,
980   "nombre": "Vitoria",
981   "habitantes": null
982 },
983 √ {
984   "id": 197,
985   "nombre": "Vitoria",
986   "habitantes": null
987 },
988 √ {
989   "id": 198,
990   "nombre": "Vitoria",
991   "habitantes": null
992 },
993 √ {
994   "id": 199,
995   "nombre": "Vitoria",
996   "habitantes": null
997 },
998 √ {
999   "id": 200,
1000  "nombre": "Vitoria",
1001  "habitantes": null
1002 },
1003 √ {
1004   "id": 201,
1005   "nombre": "Vitoria",
1006   "habitantes": null
1007 },
1008 √ {
1009   "id": 202,
1010   "nombre": "Vitoria",
1011   "habitantes": null
1012 },
1013 √ {
1014   "id": 203,
1015   "nombre": "Vitoria",
1016   "habitantes": null
1017 },
1018 √ {
1019   "id": 204,
1020   "nombre": "Vitoria",
1021   "habitantes": null
1022 },
1023 √ {
1024   "id": 205,
1025   "nombre": "Vitoria",
1026   "habitantes": null
1027 },
1028 √ {
1029   "id": 206,
1030   "nombre": "Vitoria",
1031   "habitantes": null
1032 },
1033 √ {
1034   "id": 207,
1035   "nombre": "Vitoria",
1036   "habitantes": null
1037 },
1038 √ {
1039   "id": 208,
1040   "nombre": "Vitoria",
1041   "habitantes": null
1042 },
1043 √ {
1044   "id": 209,
1045   "nombre": "Vitoria",
1046   "habitantes": null
1047 },
1048 √ {
1049   "id": 210,
1050   "nombre": "Vitoria",
1051   "habitantes": null
1052 },
1053 √ {
1054   "id": 211,
1055   "nombre": "Vitoria",
1056   "habitantes": null
1057 },
1058 √ {
1059   "id": 212,
1060   "nombre": "Vitoria",
1061   "habitantes": null
1062 },
1063 √ {
1064   "id": 213,
1065   "nombre": "Vitoria",
1066   "habitantes": null
1067 },
1068 √ {
1069   "id": 214,
1070   "nombre": "Vitoria",
1071   "habitantes": null
1072 },
1073 √ {
1074   "id": 215,
1075   "nombre": "Vitoria",
1076   "habitantes": null
1077 },
1078 √ {
1079   "id": 216,
1080   "nombre": "Vitoria",
1081   "habitantes": null
1082 },
1083 √ {
1084   "id": 217,
1085   "nombre": "Vitoria",
1086   "habitantes": null
1087 },
1088 √ {
1089   "id": 218,
1090   "nombre": "Vitoria",
1091   "habitantes": null
1092 },
1093 √ {
1094   "id": 219,
1095   "nombre": "Vitoria",
1096   "habitantes": null
1097 },
1098 √ {
1099   "id": 220,
1100   "nombre": "Vitoria",
1101   "habitantes": null
1102 },
1103 √ {
1104   "id": 221,
1105   "nombre": "Vitoria",
1106   "habitantes": null
1107 },
1108 √ {
1109   "id": 222,
1110   "nombre": "Vitoria",
1111   "habitantes": null
1112 },
1113 √ {
1114   "id": 223,
1115   "nombre": "Vitoria",
1116   "habitantes": null
1117 },
1118 √ {
1119   "id": 224,
1120   "nombre": "Vitoria",
1121   "habitantes": null
1122 },
1123 √ {
1124   "id": 225,
1125   "nombre": "Vitoria",
1126   "habitantes": null
1127 },
1128 √ {
1129   "id": 226,
1130   "nombre": "Vitoria",
1131   "habitantes": null
1132 },
1133 √ {
1134   "id": 227,
1135   "nombre": "Vitoria",
1136   "habitantes": null
1137 },
1138 √ {
1139   "id": 228,
1140   "nombre": "Vitoria",
1141   "habitantes": null
1142 },
1143 √ {
1144   "id": 229,
1145   "nombre": "Vitoria",
1146   "habitantes": null
1147 },
1148 √ {
1149   "id": 230,
1150   "nombre": "Vitoria",
1151   "habitantes": null
1152 },
1153 √ {
1154   "id": 231,
1155   "nombre": "Vitoria",
1156   "habitantes": null
1157 },
1158 √ {
1159   "id": 232,
1160   "nombre": "Vitoria",
1161   "habitantes": null
1162 },
1163 √ {
1164   "id": 233,
1165   "nombre": "Vitoria",
1166   "habitantes": null
1167 },
1168 √ {
1169   "id": 234,
1170   "nombre": "Vitoria",
1171   "habitantes": null
1172 },
1173 √ {
1174   "id": 235,
1175   "nombre": "Vitoria",
1176   "habitantes": null
1177 },
1178 √ {
1179   "id": 236,
1180   "nombre": "Vitoria",
1181   "habitantes": null
1182 },
1183 √ {
1184   "id": 237,
1185   "nombre": "Vitoria",
1186   "habitantes": null
1187 },
1188 √ {
1189   "id": 238,
1190   "nombre": "Vitoria",
1191   "habitantes": null
1192 },
1193 √ {
1194   "id": 239,
1195   "nombre": "Vitoria",
1196   "habitantes": null
1197 },
1198 √ {
119
```



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

Paginación, filtrado y ordenación:

Pretendemos tener peticiones que nos permitan filtrar, ordenar y paginar de manera independiente con urls similares a:

api/v1/ciudades

api/v1/ciudades?size=5

api/v1/ciudades?sort=nombre,desc&page=1&size=5

api/v1/ciudades?nombre=Val&page=1&size=3

api/v1/ciudades?sort=habitantes,asc&sort=id,asc&page=4&size=10



¿ Como podemos hacerlo?



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



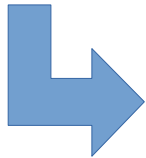
Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

... continuación **Paginación, filtrado y ordenación:**

Empezamos preguntandonos si cuando filtramos por un String existe alguna funcionalidad que nos permite hacer un 'LIKE' para comprobar si el String contiene una palabra sin tener que hacer una Query. La respuesta es si gracias a "Containing":

Implementa el service y el controller para que te devuelva el siguiente resultado:

```
CiudadRepository.java X
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > repository > CiudadRepository.java > ...
1 package edu.profesor.joseramon.dwes_futbol_rest.repository;
2
3 > import java.util.List; ...
9
10 @Repository
11 public interface CiudadRepository extends JpaRepository<CiudadDb,Long>{
12     //Métodos automáticos en Spring Data JPA
13     List<CiudadDb> findAll(Sort sort);
14     //Filtrado por nombre usando equivalente a 'LIKE' usando "Containing"
15     List<CiudadDb> findByNombreContaining(String nombre,Sort sort);
16 }
```



```
test_Rest.http X
test_Rest.http > ...
7 Content-Type: application/json
8 ###
9 ## getCiudadInfoById
10 Send Request
11 GET http://localhost:8080/api/v1/ciudades/56/info HTTP/1.1
12 Content-Type: application/json
13 ###
14 ## getCiudadesByNombreContainingListOrderByName
15 Send Request
16 GET http://localhost:8080/api/v1/ciudades/nombre/Val/orden/desc HTTP/1.1
17 Content-Type: application/json
18 ###
```



```
Response(43ms) X
1 HTTP/1.1 200
2 Content-Type: application/json
3 Transfer-Encoding: chunked
4 Date: Thu, 03 Feb 2022 10:28:02 GMT
5 Connection: close
6
7 [
8   {
9     "id": 56,
10    "nombre": "València",
11    "habitantes": 798000
12  },
13  {
14    "id": 57,
15    "nombre": "Valladolid",
16    "habitantes": 313000
17  }
18 ]
```




UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

... continuación **Paginación, filtrado y ordenación:**

Para realizar la paginación vamos a utilizar la interfaz de **Spring Data “Page”** que a su vez hereda de **“Slice”**:

```
public interface Page<T> extends Slice<T> {  
    static <T> Page<T> empty();  
    static <T> Page<T> empty(Pageable pageable);  
    long getTotalElements();  
    int getTotalPages();  
    <U> Page<U> map(Function<? super T,? extends U> converter);  
}
```

```
public interface Slice<T> extends Streamable<T> {  
    int getNumber();  
    int getSize();  
    int getNumberOfElements();  
    List<T> getContent();  
    boolean hasContent();  
    Sort getSort();  
    boolean isFirst();  
    boolean isLast();  
    boolean hasNext();  
    boolean hasPrevious();  
    ...  
}
```

No vamos a utilizar Slice, simplemente comentar que si alguna vez el rendimiento necesita mejorarse seria hora de utilizar Slice, porque tiene menos información que Page.



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

... continuación Paginación, filtrado y ordenación:

Spring Data Pageable:

Pageable es una interfaz que se utiliza como parámetro en el Repository y que permite aplicar paginación y ordenación a la base de datos.

Pageable contiene información sobre la página solicitada, como podría ser el número de página o el tamaño de la página.

```
public interface Pageable {  
    int getPageNumber();  
    int getPageSize();  
    long getOffset();  
    Sort getSort();  
    Pageable next();  
    Pageable previousOrFirst();  
    Pageable first();  
    boolean hasPrevious();  
    ...  
}
```

Cuando queremos utilizar la paginación (con o sin filtro/ordenación) podemos añadir el método como parámetro:

```
CiudadRepository.java X  
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > repository > CiudadRepository.java > Langu  
1 package edu.profesor.joseramon.dwes_futbol_rest.repository;  
2  
3 > import java.util.List; ...  
11  
12 @Repository  
13 public interface CiudadRepository extends JpaRepository<CiudadDb, Long> {  
14     //Métodos automáticos en Spring Data JPA  
15     List<CiudadDb> findAll(Sort sort);  
16     //Filtrado por nombre usando equivalente a 'LIKE' usando "Containing"  
17     List<CiudadDb> findByNombreContaining(String nombre, Sort sort);  
18     //Paginación  
19     Page<CiudadDb> findAll(Pageable pageable);  
20     Page<CiudadDb> findByNombreContaining(String nombre, Pageable pageable);  
21 }
```



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA

Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com



... continuación **Paginación, filtrado y ordenación:**

Ahora la pregunta seria:



¿Que más hace falta tocar y como? ¿Hace falta tocar el mapper? ¿Y los servicios?



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

... continuación **Paginación, filtrado y ordenación:**

El mapper no hace falta tocarlo porque no es capaz de convertir un `Page<CiudadDb>` a un `Page<CiudadList>`.



Entonces ¿Como podemos devolver una `Page<CiudadList>`?



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

... continuación **Paginación, filtrado y ordenación:**

El sistema más cómodo es NO hacerlo. Realmente solo necesitamos una estructura de datos genérica (que permita pasar cualquier tipo de lista) donde se indique:

- El número de la página solicitada
- El tamaño de pagina que queremos
- El total de elementos que coinciden con la consulta
- El total de páginas si tenemos el tamaño especificado de página
- El contenido (lista de datos)
- El criterio de ordenación

En la siguiente página vamos a generar una nueva clase donde almacenar toda esta información de manera genérica. Esto significa que igual permite almacenar un `List<CiudadList>` que un `List<CiudadInfo>`.



¿Como?



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

... continuación **Paginación, filtrado y ordenación:**

Cuando especificamos `<T>` estamos permitiendo cualquier tipo de dato y conseguimos una estructura capaz de almacenar cualquier tipo de lista. **Crea PaginaDto.java:**

```
PaginaDto.java x
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > model > dto > PaginaDto.java > ...
1  package edu.profesor.joseramon.dwes_futbol_rest.model.dto;
2
3  import java.util.List;
4
5  import org.springframework.data.domain.Sort;
6
7  import lombok.AllArgsConstructor;
8  import lombok.Getter;
9  import lombok.NoArgsConstructor;
10 import lombok.Setter;
11 @Getter
12 @Setter
13 @NoArgsConstructor
14 @AllArgsConstructor
15 public class PaginaDto<T> {
16     int number; //número de página solicitada
17     int size; //tamaño de la página
18     long totalElements; //total de elementos devueltos por la consulta sin paginación
19     int totalPages; //total páginas teniendo en cuenta el tamaño de cada página
20
21     List<T> content; //lista de elementos
22     Sort sort; //ordenación de la consulta
23 }
```



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

... continuación Paginación, filtrado y ordenación:

Ahora desde el servicio y su implementación podemos crear un nuevo método que devuelva una `PaginaDto<CiudadList>`:

```
CiudadServiceImpl.java X
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > srv > impl > CiudadServiceImpl.java > CiudadServiceImpl > findAllPageCiudadList(Pageable)
1 package edu.profesor.joseramon.dwes_futbol_rest.srv.impl;
2
3 import java.util.List;
4 import java.util.Optional;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.data.domain.Page;
7 import org.springframework.data.domain.Pageable;
8 import org.springframework.data.domain.Sort;
9 import org.springframework.stereotype.Service;
10 import edu.profesor.joseramon.dwes_futbol_rest.model.db.CiudadDb;
11 import edu.profesor.joseramon.dwes_futbol_rest.model.dto.CiudadInfo;
12 import edu.profesor.joseramon.dwes_futbol_rest.model.dto.CiudadList;
13 import edu.profesor.joseramon.dwes_futbol_rest.model.dto.PaginaDto;
14 import edu.profesor.joseramon.dwes_futbol_rest.repository.CiudadRepository;
15 import edu.profesor.joseramon.dwes_futbol_rest.srv.CiudadService;
16 import edu.profesor.joseramon.dwes_futbol_rest.srv.mapper.CiudadMapper;
17
18 @Service
19 public class CiudadServiceImpl implements CiudadService{
20     private final CiudadRepository ciudadRepository;
21
22     @Autowired
23     public CiudadServiceImpl(CiudadRepository ciudadRepository){
24         this.ciudadRepository=ciudadRepository;
25     }
26
27     public List<CiudadList> findAllCiudadList(){
28         return CiudadMapper.INSTANCE.ciudadesToCiudadList(ciudadRepository.findAll());
29     }
30
31     public List<CiudadList> findAllCiudadList(Sort sort){
32         return CiudadMapper.INSTANCE.ciudadesToCiudadList(ciudadRepository.findAll(sort));
33     }
34
35     public PaginaDto<CiudadList> findAllPageCiudadList(Pageable paging){
36         Page<CiudadDb> paginaCiudadDb=ciudadRepository.findAll(paging);
37         return new PaginaDto<CiudadList>{
38             //número de página solicitada
39             paginaCiudadDb.getNumber(), //tamaño de la página
40             paginaCiudadDb.getSize(), //total de elementos devueltos por la consulta sin paginación
41             paginaCiudadDb.getTotalElements(), //total páginas teniendo en cuenta el tamaño de cada página
42             CiudadMapper.INSTANCE.ciudadesToCiudadList(paginaCiudadDb.getContent()), //lista de elementos
43             paginaCiudadDb.getSort(); //ordenación de la consulta
44         };
45     }
46 }
```




UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

... continuación **Paginación, filtrado y ordenación:**

Y desde el Controller podemos llamar a ese método del servicio.

Fijate que igual que habíamos creado 'findAllPageCiudadList' en el servicio deberemos de **crear también en el servicio 'findByNombreContaining'**:

```
CiudadRestController.java X
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > controller > CiudadRestController.java > ...

38  /* DESACTIVAMOS EL ANTIGUO /ciudades:
39  @GetMapping("/ciudades")
40  public Collection<CiudadList> getCiudadesList() {
41      return ciudadService.findAllCiudadList();
42  } */
43
44  @GetMapping("/ciudades")
45  public ResponseEntity<Map<String, Object>> getAllCiudades(
46      @RequestParam(required = false) String nombre,
47      @RequestParam(defaultValue = "0") int page,
48      @RequestParam(defaultValue = "3") int size,
49      @RequestParam(defaultValue = "id,asc") String sort) {
50
51      try {
52          // Crear sorts (ordenación de los datos)
53          String[] orden = sort.split(",");
54          Sort sorts;
55          if (orden.length > 1)
56              sorts = Sort.by(Direction.fromString(orden[1]), orden[0]);
57          else // por defecto asc si no se dice nada
58              sorts = Sort.by(Direction.fromString("asc"), orden[0]);
59
60          // Crear solicitud de página nº 'page' de tamaño 'size'
61          // utilizando el orden 'sort'
62          Pageable paging = PageRequest.of(page, size, sorts);
63
64          PaginaDto<CiudadList> paginaCiudadesList;
65          if (nombre == null) // Sin filtrar por nombre
66              paginaCiudadesList = ciudadService.findAllPageCiudadList(paging);
67          else // filtrando nombre
68              paginaCiudadesList = ciudadService.findByNombreContaining(nombre, paging);
69
70          // Rellenar datos a devolver en el servicio REST
71          List<CiudadList> ciudades = paginaCiudadesList.getContent();
72          Map<String, Object> response = new HashMap<>();
73          response.put("data", ciudades);
74          response.put("currentPage", paginaCiudadesList.getNumber());
75          response.put("pageSize", paginaCiudadesList.getSize());
76          response.put("totalItems", paginaCiudadesList.getTotalElements());
77          response.put("totalPages", paginaCiudadesList.getTotalPages());
78          return new ResponseEntity<>(response, HttpStatus.OK);
79      } catch (Exception e) { // Si hay cualquier tipo de error
80          return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
81      }
82  }
```




UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

... continuación **Paginación, filtrado y ordenación:**

Si probamos las consultas siguientes resaltadas veremos que funcionan:

```
test_Rest.http x
test_Rest.http > ...
12  ###
13  ## getCiudadesByNombreContainingListOrderByNombre
    Send Request
14  GET http://localhost:8080/api/v1/ciudades/nombre/Val/orden/desc HTTP/1.1
15  Content-Type: application/json
16  ###
17  ## getAllCiudades
    Send Request
18  GET http://localhost:8080/api/v1/ciudades HTTP/1.1
19  Content-Type: application/json
20  ###
21  ## getAllCiudades
    Send Request
22  GET http://localhost:8080/api/v1/ciudades?size=5 HTTP/1.1
23  Content-Type: application/json
24  ###
25  ## getAllCiudades
    Send Request
26  GET http://localhost:8080/api/v1/ciudades?sort=nombre,desc&page=1&size=5 HTTP/1.1
27  Content-Type: application/json
28  ###
29  ## getAllCiudades
    Send Request
30  GET http://localhost:8080/api/v1/ciudades?nombre=Val&page=0&size=10 HTTP/1.1
31  Content-Type: application/json
32  ###
33  ## getAllCiudades
    Send Request
34  GET http://localhost:8080/api/v1/ciudades?sort=nombre,asc&nombre=Val&page=0&size=10 HTTP/1.1
35  Content-Type: application/json
36  ###
```



... continuación **Paginación, filtrado y ordenación:**

Solo nos queda la petición en la que le indicamos más de un método de ordenación (primero por habitantes y luego por id):

```
test_Rest.http x
test_Rest.http > ...
31 Content-Type: application/json
32 ###
33 ## getAllCiudades
Send Request
34 GET http://localhost:8080/api/v1/ciudades?sort=nombre,asc&nombre=Val&page=0&size=10 HTTP/1.1
35 Content-Type: application/json
36 ###
37 ## getAllCiudades
Send Request
38 GET http://localhost:8080/api/v1/ciudades?sort=habitantes,asc&sort=id,asc&page=4&size=10 HTTP/1.1
39 Content-Type: application/json
40 ###
41
```



¿Como especificamos más de un criterio de ordenación? ¿Que clases debemos cambiar?



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

... continuación **Paginación, filtrado y ordenación:**

Lo único que deberemos cambiar es el controller porque la clase 'Sort' nos permite especificar más de un criterio de ordenación gracias a la clase "Order" de org.springframework.data.domain.Sort. Fijate que el parámetro 'sort' del método ahora no es un String sino un vector (String[]):

```
CiudadRestController.java X
src > main > java > edu > profesor > joseramon > dwes_futbol_rest > controller > CiudadRestController.java > CiudadRestContro

46     @GetMapping("/ciudades")
47     public ResponseEntity<Map<String, Object>> getAllCiudades(
48         @RequestParam(required = false) String nombre,
49         @RequestParam(defaultValue = "0") int page,
50         @RequestParam(defaultValue = "3") int size,
51         @RequestParam(defaultValue = "id,asc") String[] sort) {
52
53         try {
54             // Crear sorts (ordenación de los datos)
55             List<Order> criteriosOrdenacion = new ArrayList<Order>();
56             // El primer criterio de ordenación siempre deberá de contener el orden(asc,desc)
57             if(sort[0].contains(",")){ // Hay más de un criterio de ordenación
58                 // Tenemos un vector de ordenaciones en 'sort' y debemos leerlos
59                 for (String criterioOrdenacion : sort) {
60                     String[] orden = criterioOrdenacion.split(",");
61                     if (orden.length > 1)
62                         criteriosOrdenacion.add(new Order(Direction.fromString(orden[1]), orden[0]));
63                     else // por defecto asc si no se dice nada
64                         criteriosOrdenacion.add(new Order(Direction.fromString("asc"), orden[0]));
65                 }
66             } else{ // Solo hay un criterio de ordenación
67                 // El primer elemento del vector de sort es la dirección y el segundo el campo
68                 criteriosOrdenacion.add(new Order(Direction.fromString(sort[1]), sort[0]));
69             }
70             Sort sorts = Sort.by(criteriosOrdenacion);
71
72             // Crear solicitud de página nº 'page' de tamaño 'size'
73             // utilizando el orden 'sorts'
74             Pageable paging = PageRequest.of(page, size, sorts);
```




... continuación Paginación, filtrado y ordenación:

Si lo probamos las 2 consultas devuelven lo mismo:

```
test_Rest.http x
test_Rest.http > ...
32 ###
33 ## getAllCiudades
34 GET http://localhost:8080/api/v1/ciudades?sort=nombre,asc&nombre=Val&page=0&size=10 HTTP/1.1
35 Content-Type: application/json
36 ###
37 ## getAllCiudades
38 GET http://localhost:8080/api/v1/ciudades?sort=habitantes,asc&sort=id,asc&page=4&size=10 HTTP/1.1
39 Content-Type: application/json
40 ###
41 ## getAllCiudades
42 GET http://localhost:8080/api/v1/ciudades?sort=habitantes,asc&sort=id&page=4&size=10 HTTP/1.1
43 Content-Type: application/json
44 ###
```

```
Response(335ms) x
1 HTTP/1.1 200
2 Content-Type: application/json
3 Transfer-Encoding: chunked
4 Date: Fri, 04 Feb 2022 08:05:18 GMT
5 Connection: close
6
7 {
8   "totalItems": 58,
9   "data": [
10    {
11      "id": 55,
12      "nombre": "Toledo",
13      "habitantes": null
14    },
15    {
16      "id": 59,
17      "nombre": "Zamora",
18      "habitantes": null
19    },
20    {
21      "id": 19,
22      "nombre": "Cornellà de Llobregat",
23      "habitantes": 85000
24    },
25    {
26      "id": 21,
27      "nombre": "Getafe",
28      "habitantes": 169000
29    },
30    {
31      "id": 47,
32      "nombre": "Sant Sebastià",
33      "habitantes": 186000
34    },
35    {
```




UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

... continuación **Paginación, filtrado y ordenación:**

Para afianzar los conceptos vistos en esta práctica se pide al alumno **ampliar el servicio rest con las mismas llamadas que en la tabla 'Ciudades' pero sobre la tabla 'Jornadas'** que el alumno puede encontrar en el DRIVE. En data.sql tenéis la definición de la tabla y los inserts:

```
test_Rest.http  data.sql  x
src > main > resources > data.sql
1  -- Copyright Jose Ramón Cebolla - 2022
2  -----
3  -- Table `ciudades`
4  -----
5  DROP TABLE IF EXISTS ciudades;
6  CREATE TABLE IF NOT EXISTS ciudades(
7  id IDENTITY,
8  nombre VARCHAR(50),
9  habitantes BIGINT,
10  CONSTRAINT pk_ciudades PRIMARY KEY(id));
11
12
13  -----
14  -- Table `jornadas`
15  -----
16  DROP TABLE IF EXISTS `jornadas`;
17
18  CREATE TABLE IF NOT EXISTS `jornadas` (
19  `num` INT(11) NOT NULL,
20  `fecha` VARCHAR(10) NULL DEFAULT NULL,
21  CONSTRAINT pk_jornadas PRIMARY KEY (`num`));
22
23  -- INSERCIÓN DE DATOS
24  -----
```



UD 3: Bases de datos y servicios REST

6.- Paginación en Spring Data JPA

Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com



EJERCICIO:

Sube la aplicación final al moodle.

Para ello:

1º Haz un “Maven Clean” para dejar solo los ficheros fuentes y quitar momentaneamente los necesarios para ejecutar la aplicación (dependencias).

2º Comprime la carpeta de tu aplicación y ponle como nombre al fichero comprimido UD3_practica6_nombreAlumno.tar.gz donde nombreAlumno es el nombre del alumno que entrega la práctica.

3º Súbela al moodle.

IMPORTANTE: No comprimir en RAR, porque Ubuntu no lo lee bien y en clase tenemos Ubuntu. Si tuviésemos Windows, podemos comprimir en ZIP.