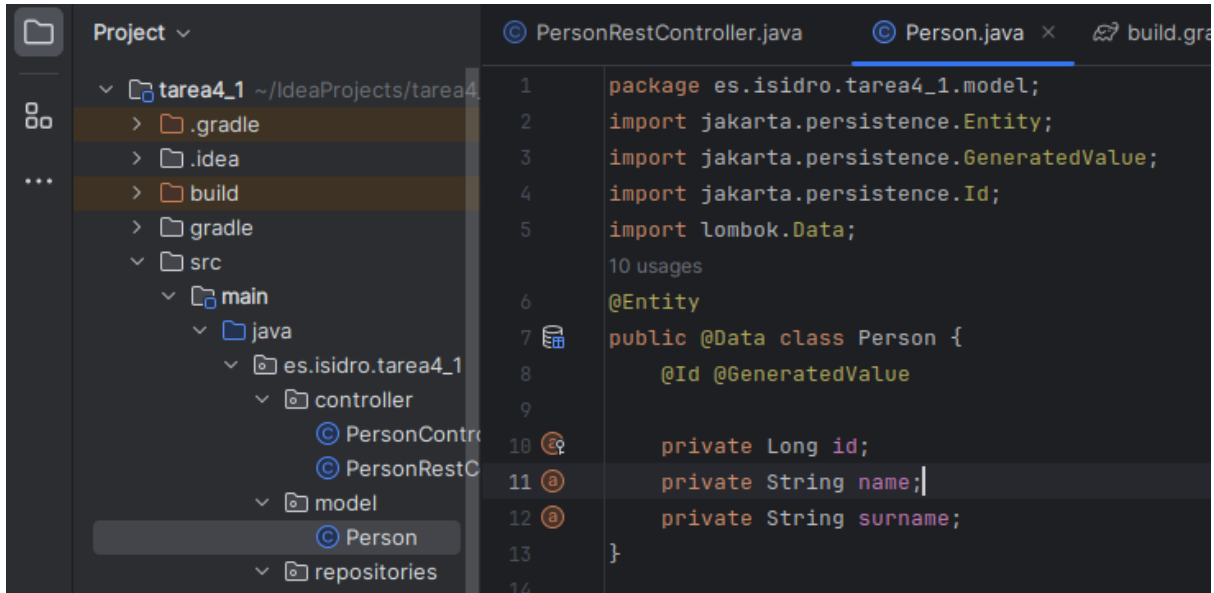
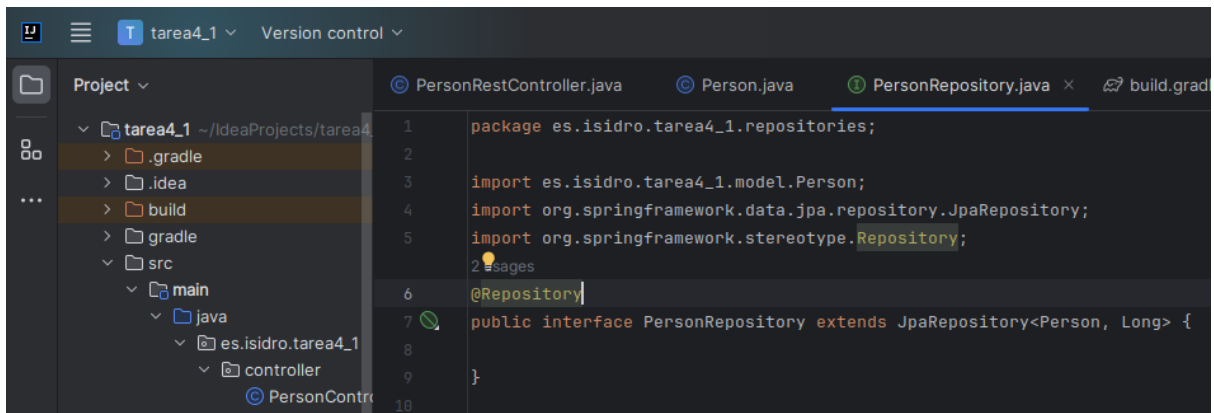


Apartado B

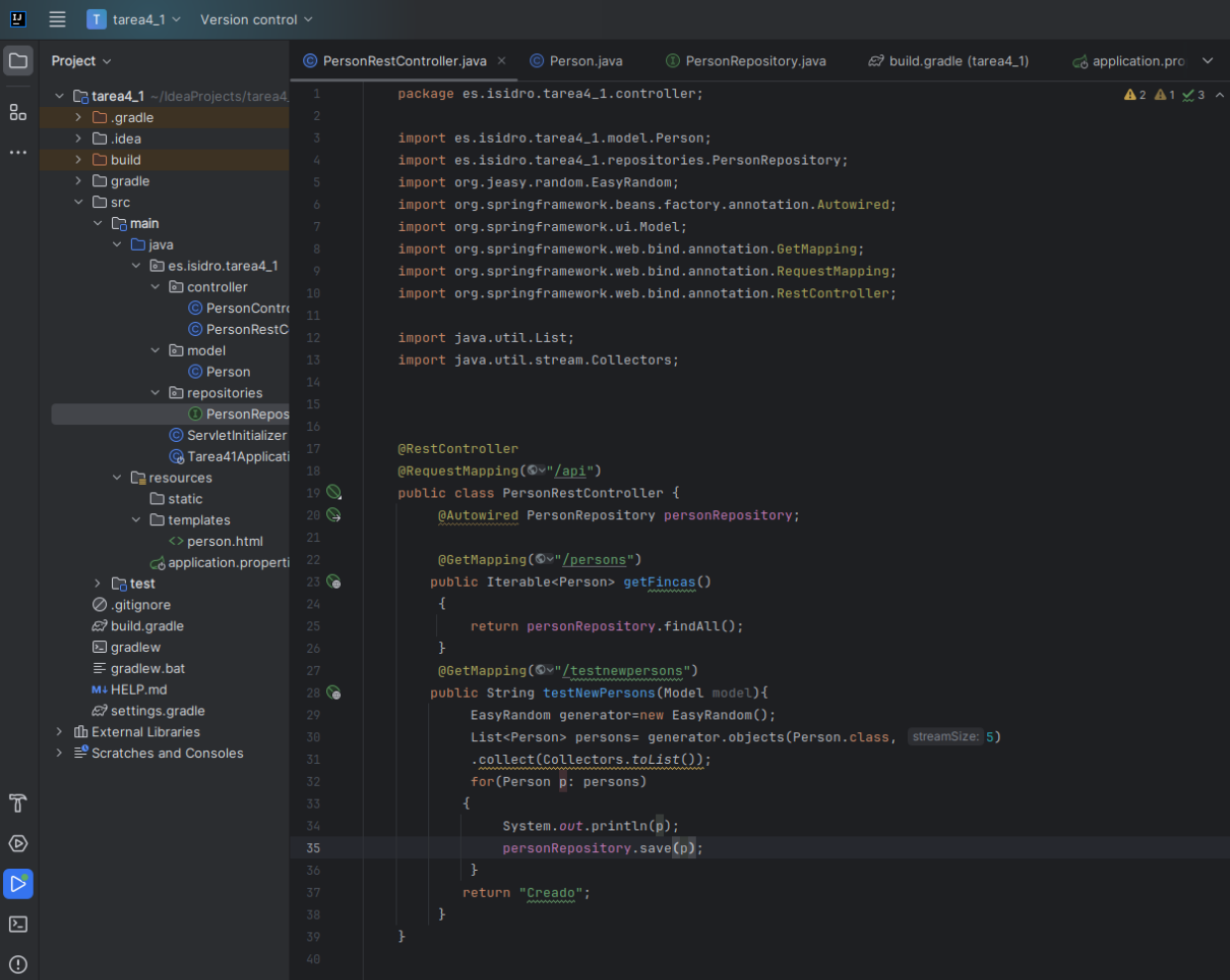
Una vez implementado el código del apartado a empezamos a modificar algunas cosas. En la clase person metemos el `@Entity`, el `@Id` y el `@GeneratedValue` para que sea primary key y el id se genere automáticamente:



Luego creamos una interfaz llamada `PersonRepository` y añadimos `@Repository`, después le indicamos el tipo de base y el primary key.



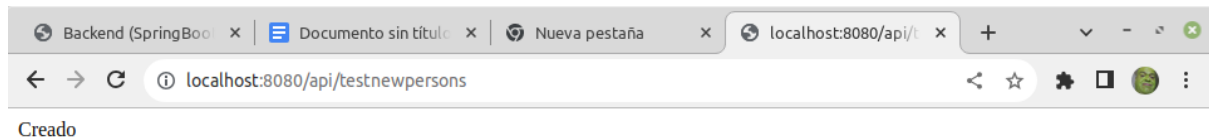
A continuación creamos un controlador llamado PersonRestController para probarlo



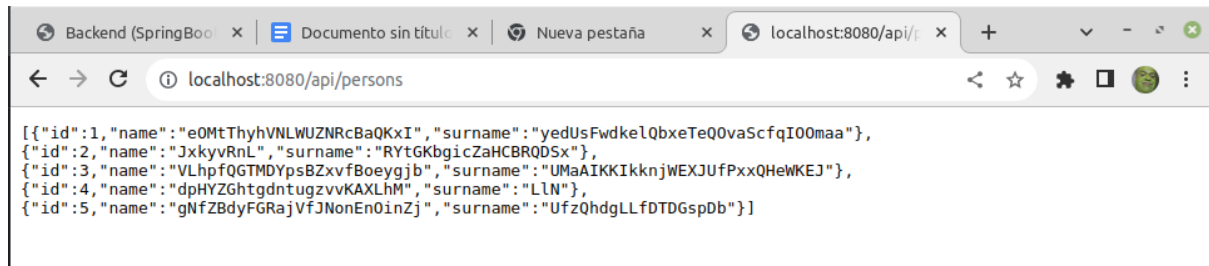
The screenshot shows an IDE with a project named 'tarea4_1'. The left sidebar displays the project structure, including folders like 'src/main/java' and 'src/test'. The main editor area shows the code for 'PersonRestController.java'. The code includes package declarations, imports for Spring and other utilities, and three methods: 'getFincas()', 'testNewPersons()', and a return statement.

```
1 package es.isidro.tarea4_1.controller;
2
3 import es.isidro.tarea4_1.model.Person;
4 import es.isidro.tarea4_1.repositories.PersonRepository;
5 import org.jeasy.random.EasyRandom;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.ui.Model;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RestController;
11
12 import java.util.List;
13 import java.util.stream.Collectors;
14
15 @RestController
16 @RequestMapping("/api")
17 public class PersonRestController {
18     @Autowired PersonRepository personRepository;
19
20     @GetMapping("/persons")
21     public Iterable<Person> getFincas()
22     {
23         return personRepository.findAll();
24     }
25
26     @GetMapping("/testnewpersons")
27     public String testNewPersons(Model model){
28         EasyRandom generator=new EasyRandom();
29         List<Person> persons= generator.objects(Person.class, streamSize: 5)
30             .collect(Collectors.toList());
31         for(Person p: persons)
32         {
33             System.out.println(p);
34             personRepository.save(p);
35         }
36         return "Creado";
37     }
38 }
39
40
```

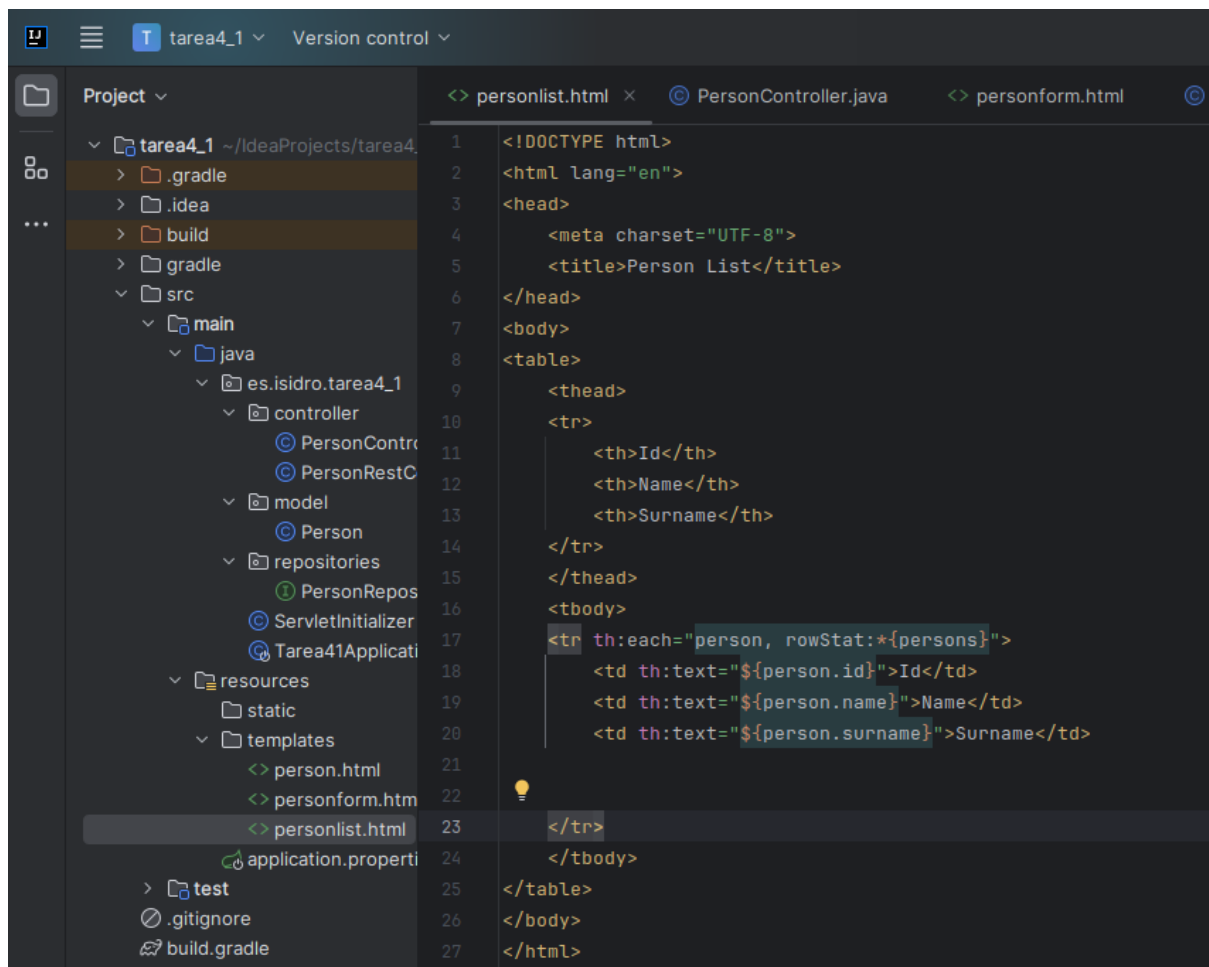
Cuando lo probemos se verá así:

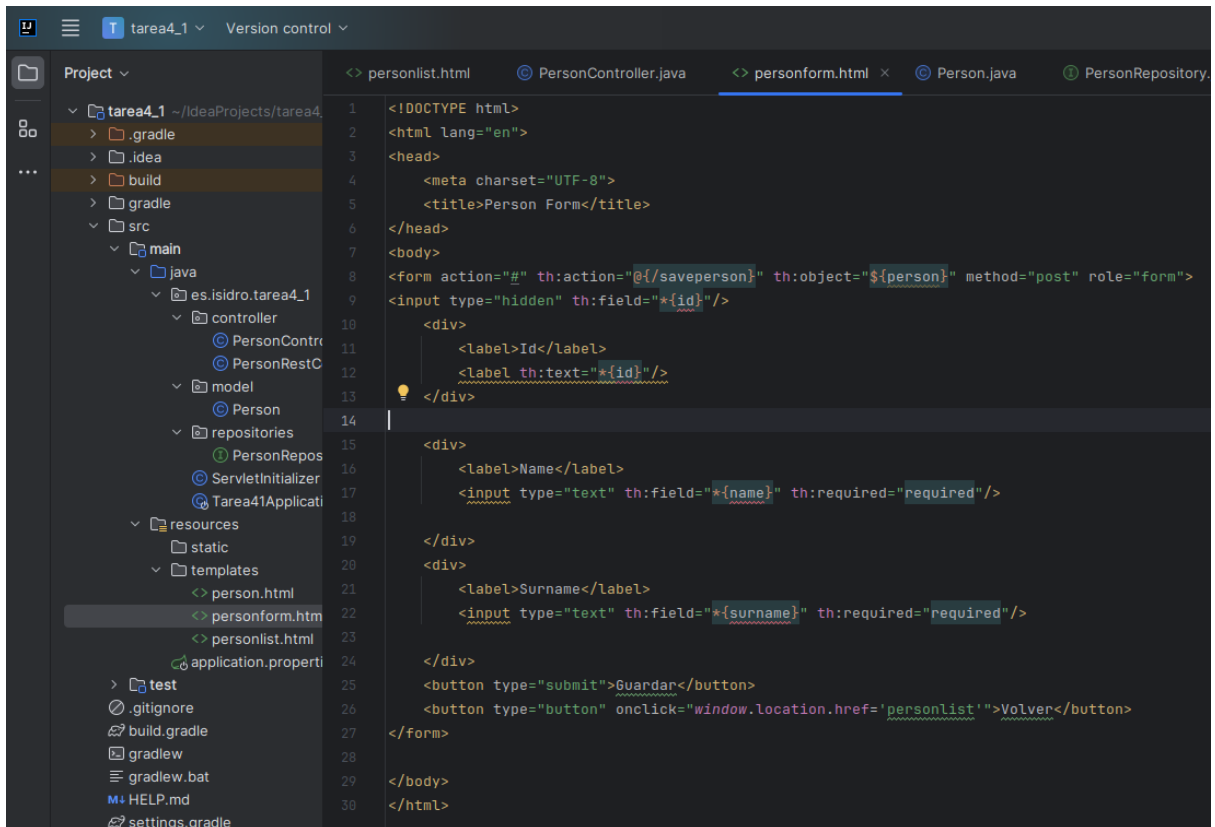


Y una vez que se haya accedido a esta página, vamos al /persons y se habrán creado 5 personas:



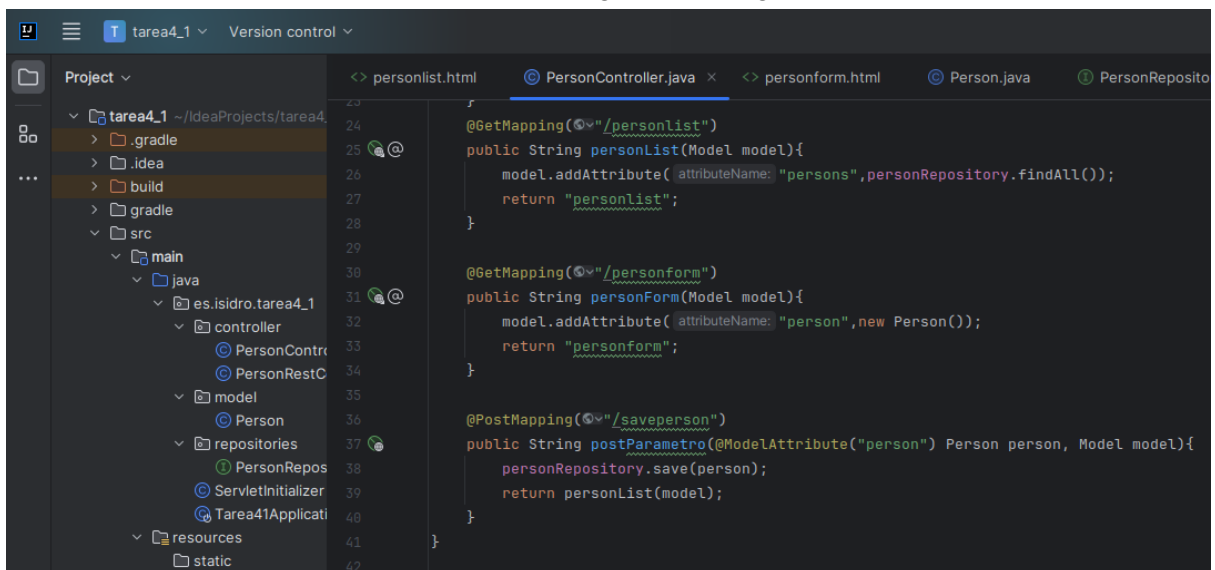
Ahora vamos a crear dos html llamados personlist y personform con el siguiente contenido para tener unas tablas con las personas:





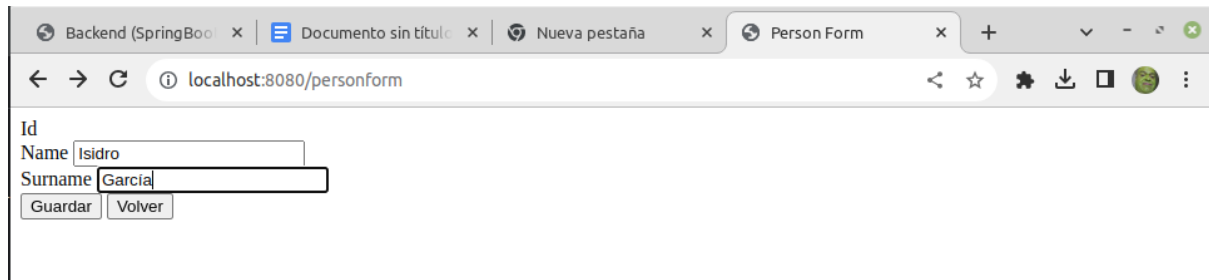
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Person Form</title>
6 </head>
7 <body>
8     <form action="#" th:action="@{/saveperson}" th:object="${person}" method="post" role="form">
9     <input type="hidden" th:field="*{id}"/>
10     <div>
11         <label>Id</label>
12         <label th:text="*{id}"/>
13     </div>
14
15     <div>
16         <label>Name</label>
17         <input type="text" th:field="*{name}" th:required="required"/>
18
19     </div>
20     <div>
21         <label>Surname</label>
22         <input type="text" th:field="*{surname}" th:required="required"/>
23
24     </div>
25     <button type="submit">Guardar</button>
26     <button type="button" onclick="window.location.href='personlist'">Volver</button>
27 </form>
28
29 </body>
30 </html>
```

Y ahora añadimos en el personController el siguiente código:



```
24
25
26 @GetMapping("/personlist")
27 public String personList(Model model){
28     model.addAttribute("persons", personRepository.findAll());
29     return "personlist";
30 }
31
32 @GetMapping("/personform")
33 public String personForm(Model model){
34     model.addAttribute("person", new Person());
35     return "personform";
36 }
37
38 @PostMapping("/saveperson")
39 public String postParametro(@ModelAttribute("person") Person person, Model model){
40     personRepository.save(person);
41     return personList(model);
42 }
```

Cuando lo probemos, si entramos a personform se verá esto



Backend (SpringBoot) x Documento sin título x Nueva pestaña x Person Form x + -

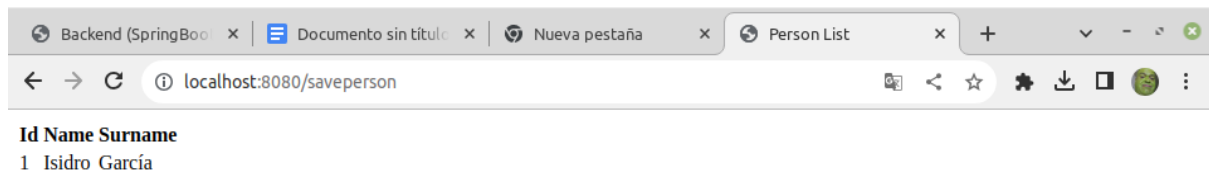
localhost:8080/personform

Id

Name

Surname

Y al darle a guardar nos mandara a saveperson y se verá esto:

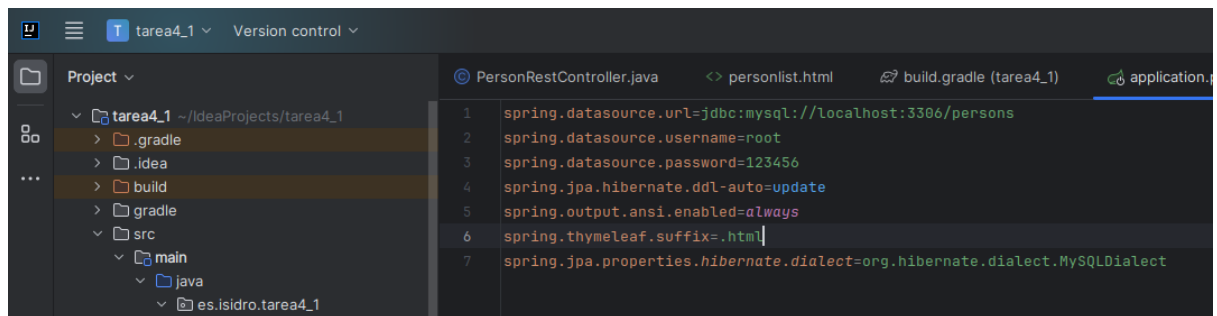


Backend (SpringBoot) x Documento sin título x Nueva pestaña x Person List x + -

localhost:8080/saveperson

Id	Name	Surname
1	Isidro	García

Ahora añadimos estas lineas al application.properties

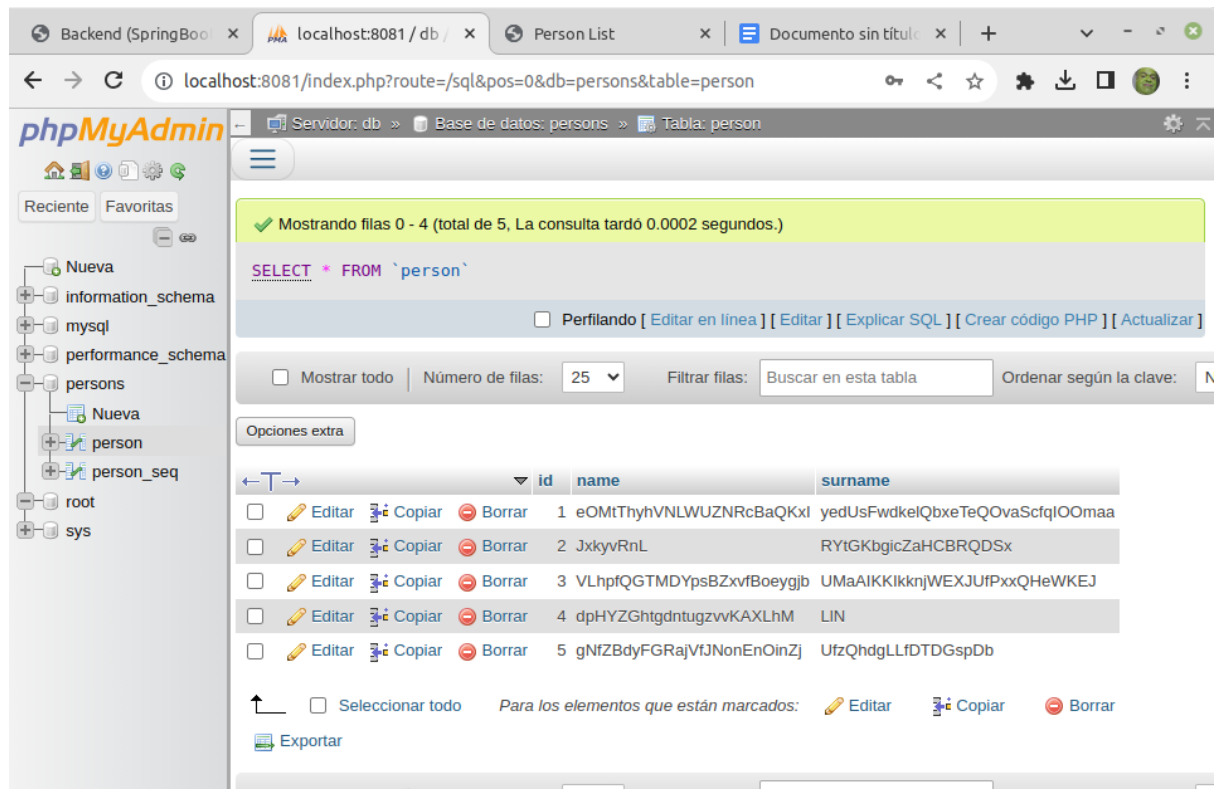


```
1 spring.datasource.url=jdbc:mysql://localhost:3306/persons
2 spring.datasource.username=root
3 spring.datasource.password=123456
4 spring.jpa.hibernate.ddl-auto=update
5 spring.output.ansi.enabled=always
6 spring.thymeleaf.suffix=.html
7 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

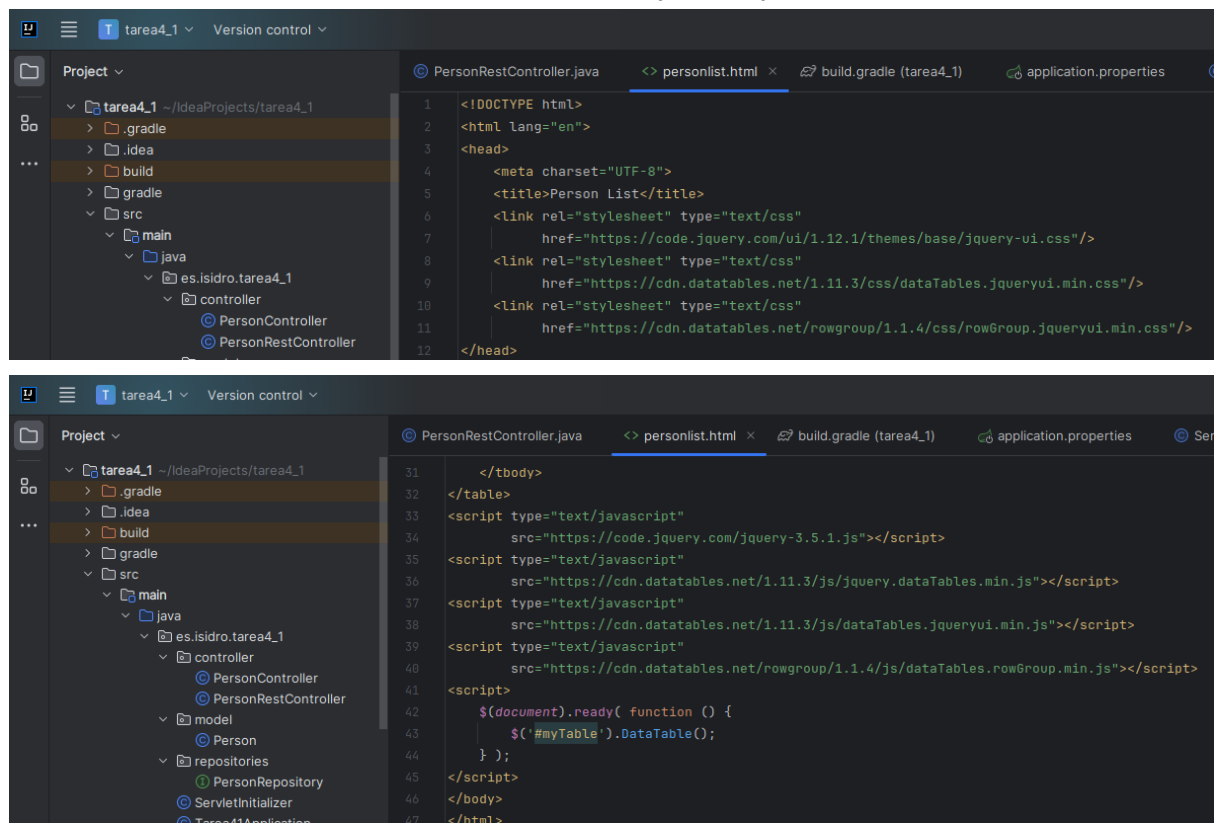
Project

- tarea4_1 ~//IdeaProjects/tarea4_1
 - .gradle
 - .idea
 - build
 - gradle
 - src
 - main
 - java
 - es.isidro.tarea4_1

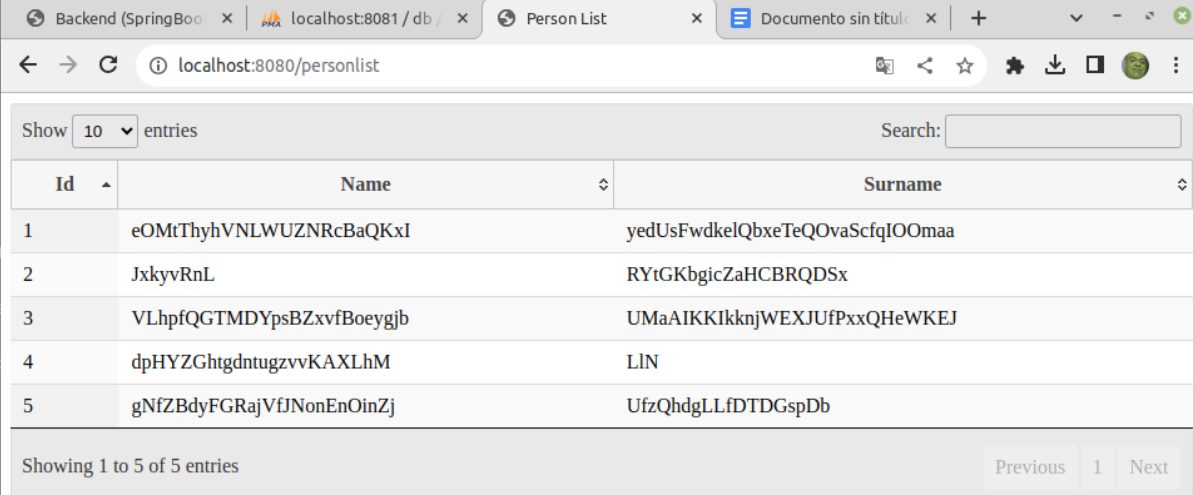
Una vez tengamos phpmyadmin y mariadb instalados y creemos una base de datos llamada persons, cuando generemos usuarios se podrán observar desde aquí:



Por ultimo vamos a añadir esto al html en el head y el body



Y ahora cuando entremos a el /personlist se verá de esta forma gracias a las líneas de css que hemos añadido:



Show	10	entries	Search:	<input type="text"/>
Id	Name	Surname		
1	eOMtThyhVNLWUZNRcBaQKxI	yedUsFwdkelQbxTeQOvaScfqIOOmaa		
2	JxkyvRnL	RYtGKbgicZaHCBRQDSx		
3	VLhpfQGTMDYpsBZxvfBoeygjb	UMaAIKKIknjWEXJUfPxxQHeWKEJ		
4	dpHYZGhtgdntugzvKAXLhM	LIN		
5	gNfZBdyFGRajVfJNonEnOinZj	UfzQhdgLLfDTDGspDb		
Showing 1 to 5 of 5 entries			Previous	1 Next