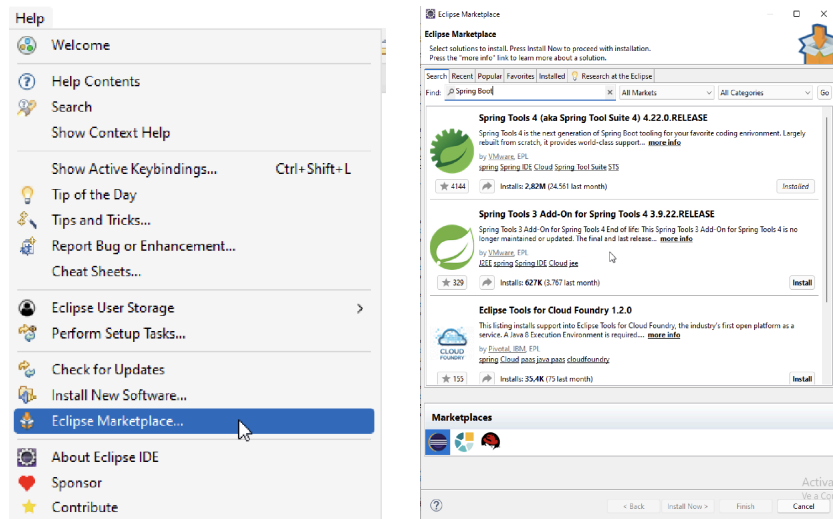


## Laboratorio 1

### Profundización Programación Orientada a Objetos II

Creación de un proyecto API Rest con el framework Spring Boot, motor de base de datos MySQL, JAVA, JPA y repositorio de dependencia Maven.

A través del IDE Eclipse se realizará la creación de la aplicación Laboratorio1 con las anteriores tecnologías mencionadas, para esto Eclipse deberá tener configurado Spring Boot la cual se puede realizar por la opción Help > Eclipse Marketplace para posteriormente buscar Spring Boot.

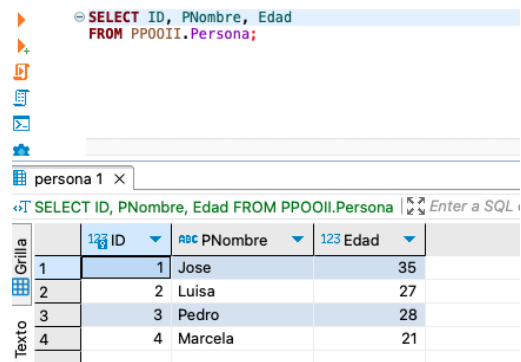


1. Creación de base de datos “ppooii” y la tabla “persona”, para el proceso se debe tener la base de datos con algunos datos referentes para la generación y visualización correspondiente.
  - La tabla persona se crea con tres campos
    - i. ID: identificador único de los registros de persona de tipo bigint configurado AUTO\_INCREMENT y con el constraint PRIMARY KEY
    - ii. PNOMBRE: representa el primer nombre de una persona y será de tipo varchar y no podrá ser null
    - iii. EDAD: representa la edad de la persona y será de tipo INT y puede ser null.

```
CREATE DATABASE `ppooii` /*!40100 DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
```

-- ppooii.persona definition

```
CREATE TABLE `persona` (  
  `ID` bigint NOT NULL AUTO_INCREMENT,  
  `PNOMBRE` varchar(100) CHARACTER SET utf8mb4 COLLATE  
utf8mb4_0900_ai_ci NOT NULL,  
  `EDAD` int DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;
```

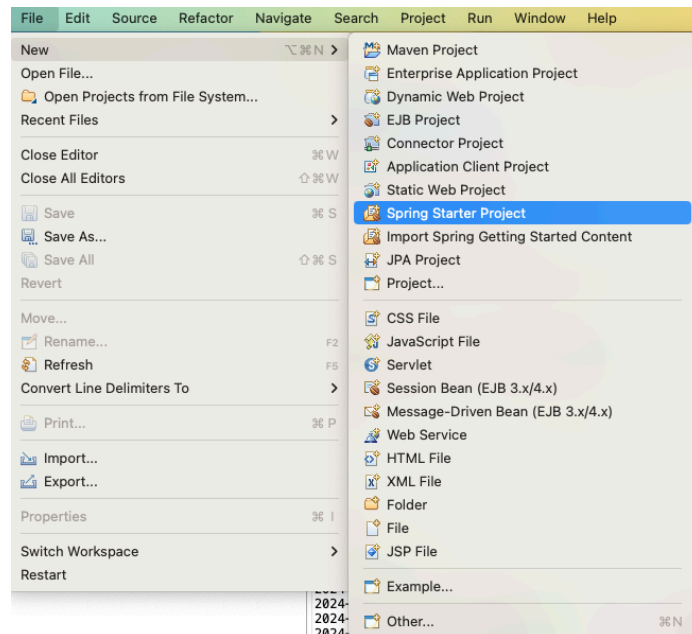


The screenshot shows a database IDE with a SQL query editor at the top and a results grid below. The query is: `SELECT ID, PNombre, Edad FROM PP00II.Persona;`. The results grid displays four rows of data.

	ID	PNombre	Edad
1	1	Jose	35
2	2	Luisa	27
3	3	Pedro	28
4	4	Marcela	21

## 2. Creación de proyecto Sprint Boot

- Crear un nuevo proyecto Spring con la opción “Spring Starter Project”



- Realizar las siguientes configuraciones iniciales del proyecto, como lo es el nombre, el tipo de repositorio de dependencias Maven y la definición del paquete como lo muestra la imagen.

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type:  Packaging:

Java Version:  Language:

Group:

Artifact:

Version:

Description:

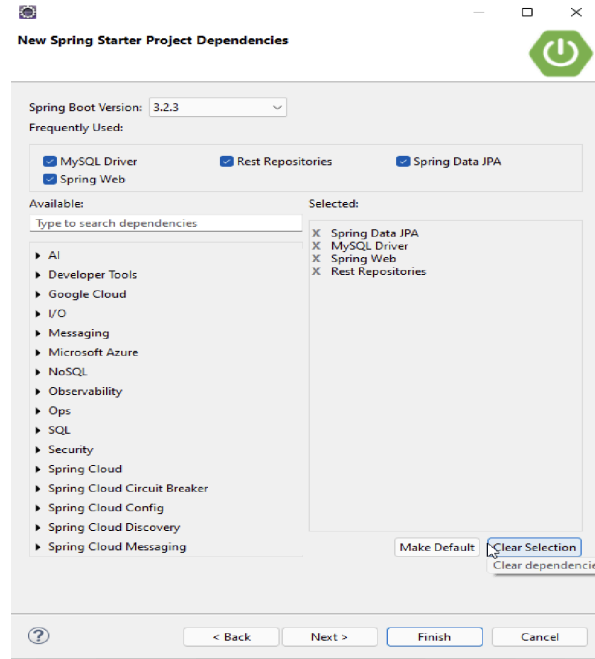
Package:

Working sets

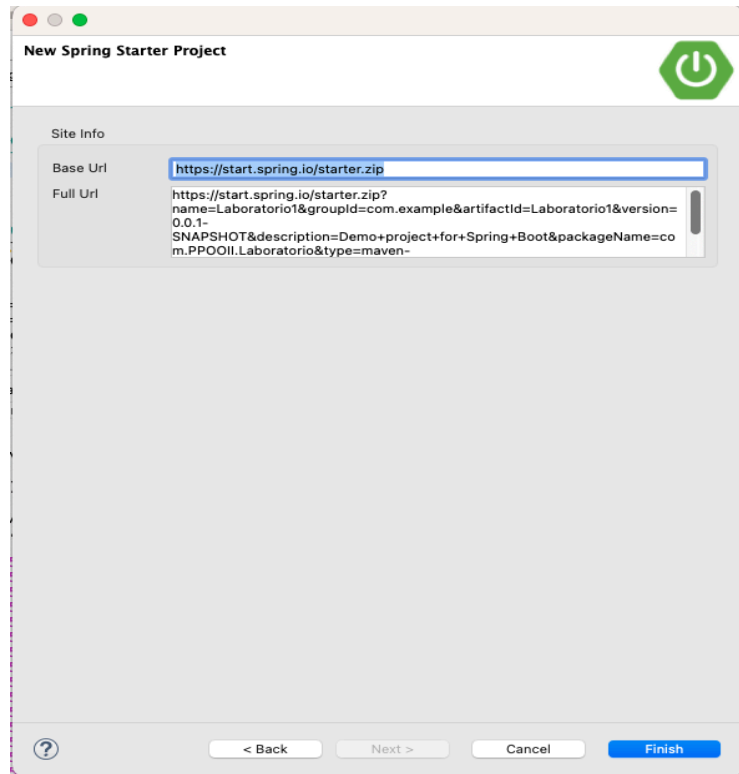
☐ Add project to working sets

Working sets:

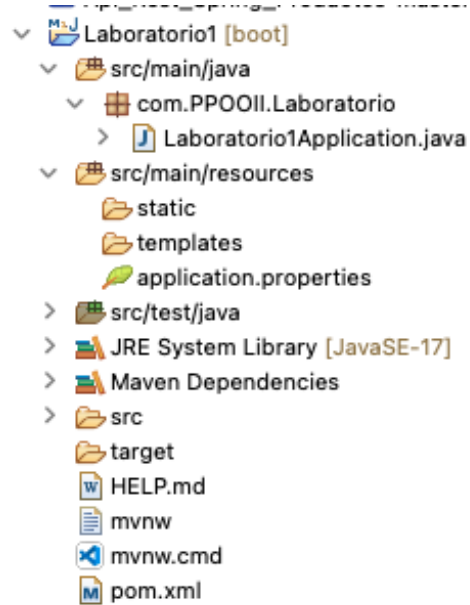
- Al dar siguiente habilitar las respectivas dependencias del proyecto como lo muestra la imagen
  - i. Spring Data JPA
  - ii. Spring Web
  - iii. MySQL Driver
  - iv. Rest Repositories



- Continuar con la configuración y dar finalizar



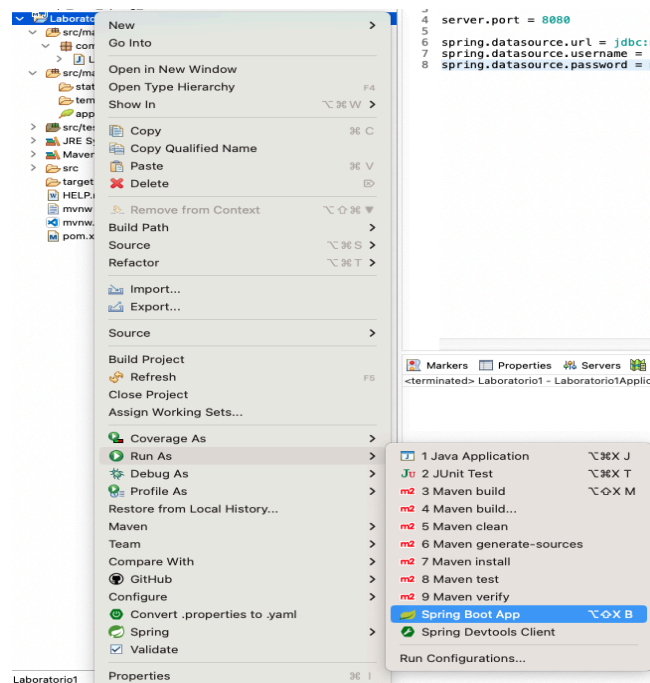
- Terminada la configuración se creará un proyecto con la siguiente estructura



- Se realiza la configuración de la base de datos creada en el punto 1 de la siguiente forma. Importante configurar un puerto que no esté ocupado por alguna aplicación previamente instalada.

```
application.proper... X Persona.java PersonaRepository.ja... IPers
1  spring.application.name=Laboratorio1
2
3
4  server.port = 9001
5
6  # Configurar la coneccion a la base de datos
7  spring.datasource.url = jdbc:mysql://localhost:3306/PP00II
8  spring.datasource.username = root
9  spring.datasource.password = mysqlroot
10 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
11
12 # Indica si debe mostrar el log de las consultas sql ejecutadas
13 # Bueno a la hora de depurar
14 spring.jpa.show-sql= true
15
16 # Configurar Hibernate
17 spring.jpa.hibernate.ddl-auto= none
18 spring.data.jpa.repositories.enabled=true
19
20
```

Seguida esta configuración corre el proyecto como una aplicación Sprint Boot.



Y en la consola de salida se podrá evidenciar la inicialización del proyecto Laboratorio y donde se verifica el puerto por donde los servicios Rest serán accedidos.

```

:: Spring Boot :: (v3.2.3)

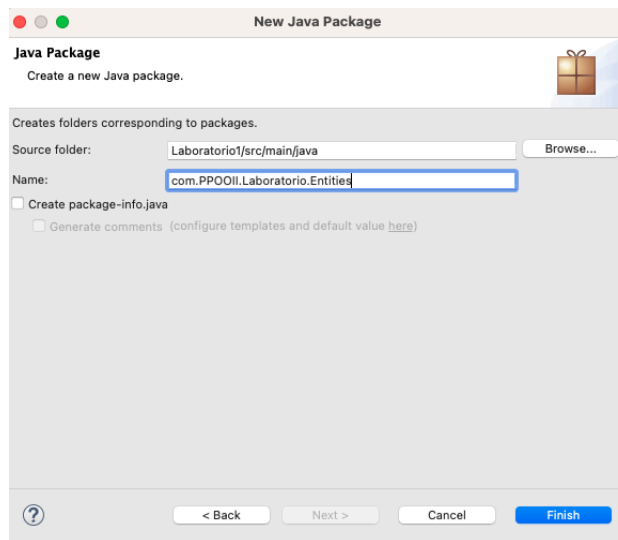
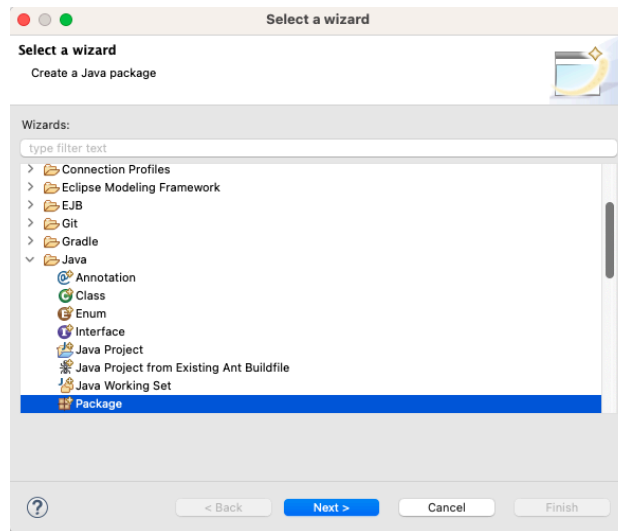
2024-03-19T11:14:57.980-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:00.184-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:00.321-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:01.660-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:01.697-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:01.700-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:01.854-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:01.855-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:02.337-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:02.466-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:02.545-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:02.545-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:02.595-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:03.065-05:00 WARN 4200 --- [Laboratorio1] [
2024-03-19T11:15:03.693-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:03.697-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:05.124-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:05.136-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:05.279-05:00 WARN 4200 --- [Laboratorio1] [
2024-03-19T11:15:07.320-05:00 INFO 4200 --- [Laboratorio1] [
2024-03-19T11:15:07.368-05:00 INFO 4200 --- [Laboratorio1] [

main] c.P.Laboratorio.Laboratorio1Application : Starting Laboratorio1Application
main] c.P.Laboratorio.Laboratorio1Application : No active profile set, falling ba
main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA rep
main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository c
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 9001
main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
main] o.s.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded Web
main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: init
Loading class 'com.mysql.jdbc.Driver'. This is deprecated. The new driver class is 'com.mysql.cj.jdbc.Driver'. The driver is automatically registered via
main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing Persistence
main] org.hibernate.Version : HHH000042: Hibernate ORM core ver
main] o.h.c.i.internal.RegionFactoryInitiator : No LoadTimeWeaver setup: ignoring
main] o.s.c.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring
main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
main] c.zaxxer.hikari.util.DriverDataSource : Registered driver with driverClass
main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection c
main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform availa
main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFact
main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enable
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 9001 (http
main] c.P.Laboratorio.Laboratorio1Application : Started Laboratorio1Application

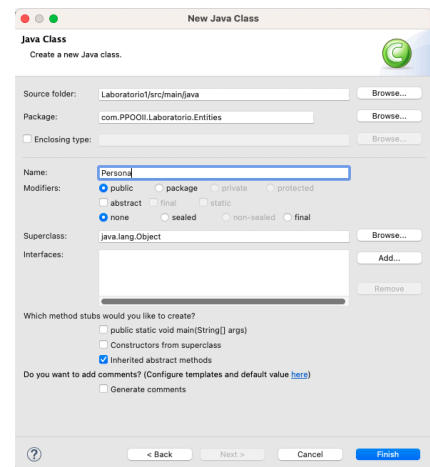
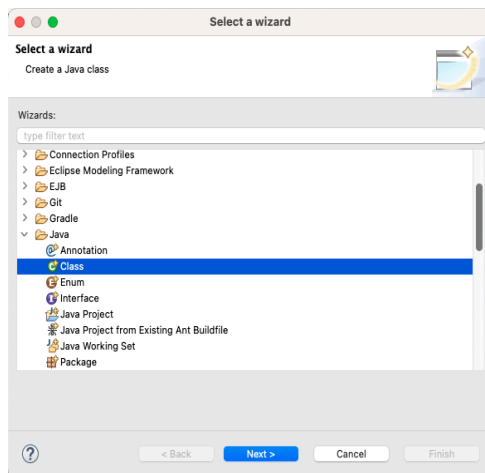
```

3. Creación de paquete, entidad persona, repositorio, servicio y controlador
  - Creación de la entidad “Persona”, la cual a través de la etiqueta @Entity se accederá a la persistencia de la tabla persona de la BD.

Primero se crea el paquete “Entities”



A continuación, se crea la clase “Persona” que representa la entidad, respetando las anotaciones configuradas.



```

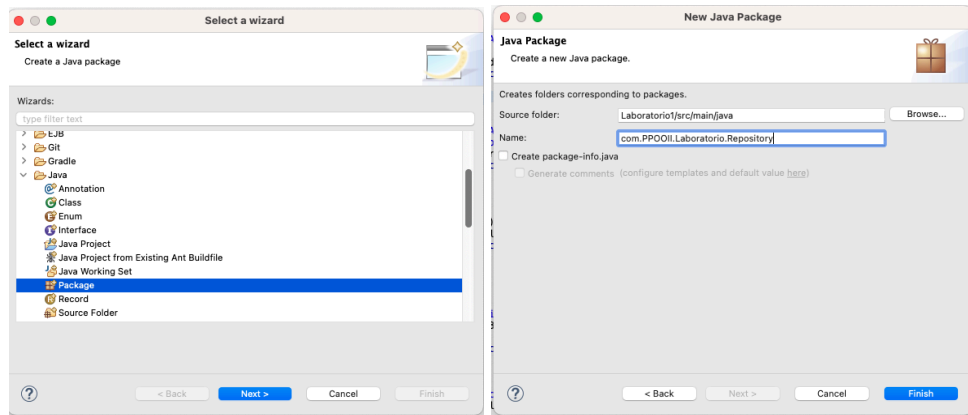
1 package com.PP00II.Laboratorio.Entities;
2
3 import java.io.Serializable;
4
11
12 @Entity
13 @Table(name = "Persona", schema = "PP00II")
14 public class Persona implements Serializable {
15     |
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     @Id
18     @Column(name = "ID")
19     public int Id;
20
21     @Column(name = "PNombre")
22     private String PNombre;
23
24     @Column(name = "Edad")
25     private int Edad;
26
27     public Persona () {}
28
29     public Persona(String PNombre, int Edad) {
30         super();
31         this.PNombre = PNombre;
32         this.Edad = Edad;
33     }
34
35     public Persona(int Id, String PNombre, int Edad) {
36         super();
37         this.Id = Id;
38         this.PNombre = PNombre;
39         this.Edad = Edad;
40     }
41
42     public int getId() {
43         return Id;
44     }
45
46     public void setId(int id) {
47         Id = id;
48     }
49
50     public String getPNombre() {
51         return PNombre;
52     }
53
54     public void setPNombre(String pNombre) {
55         PNombre = pNombre;
56     }
57
58     public int getEdad() {
59         return Edad;
60     }
61
62     public void setEdad(int edad) {
63         Edad = edad;
64     }
65
66     @Override
67     public String toString() {
68         return "Persona [id=" + this.Id + ", Primer Nombre=" + this.PNombre + ", Edad=" + this.Edad + "];
69     }
70 }
71

```

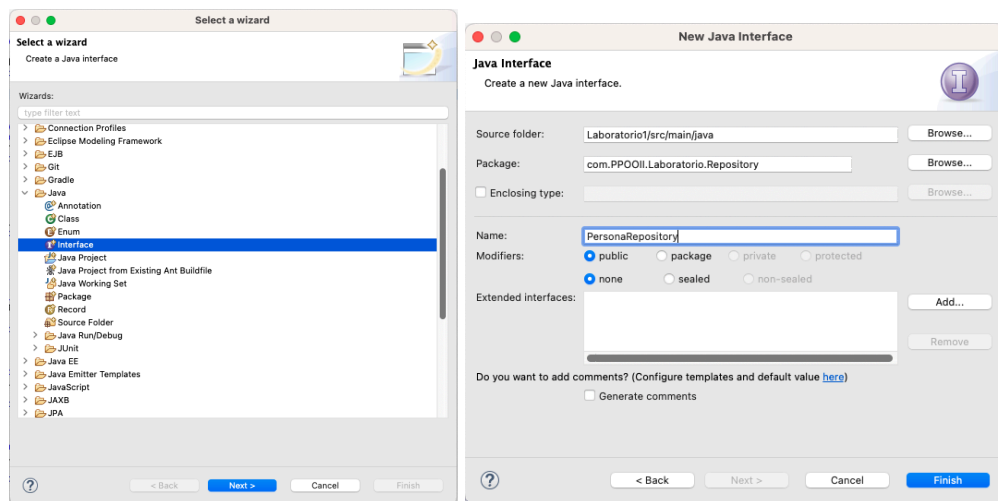
- Creación de la interfaz de repositorio en la cual se definen los métodos de acceso a la entidad, en este caso a los definidos por JpaRepository y CrudRepository con la anotación @Repository.

Primero se crea el paquete “Repository”





A continuación, se crea la respectiva interfaz



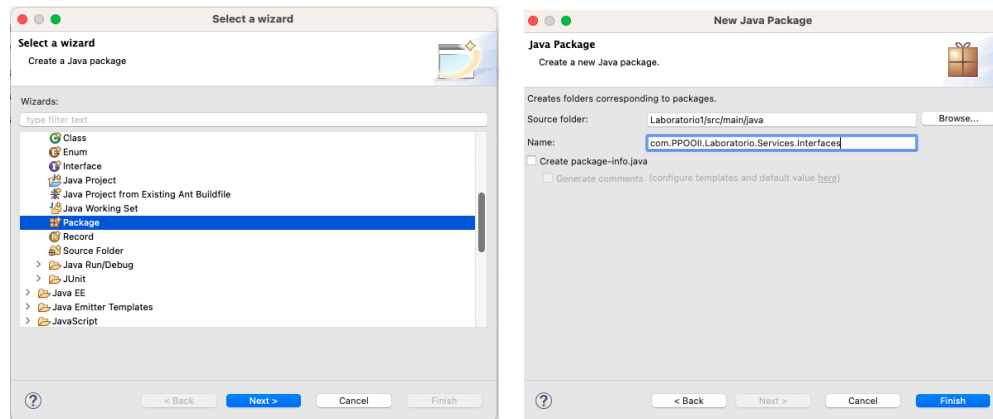
```

application.properties  Persona.java  LaboratorioI/pom.xml  PersonaRepository.java X
1 package com.PP00II.Laboratorio.Repository;
2
3 import java.io.Serializable;
4 import java.util.List;
5
6 import org.springframework.data.domain.Page;
7 import org.springframework.data.domain.Pageable;
8 import org.springframework.data.jpa.repository.JpaRepository;
9 import org.springframework.data.repository.PagingAndSortingRepository;
10 import org.springframework.stereotype.Repository;
11
12 import com.PP00II.Laboratorio.Entities.Persona;
13
14 @Repository("IPersonaRepo")
15 public interface PersonaRepository extends JpaRepository<Persona, Serializable>, PagingAndSortingRepository<Persona, Serializable> {
16
17     //Hay Métodos que JPA ya los tiene desarrollados, se pueden crear para tener
18     //una manipulación más específica a la hora de usarlos en el service
19
20     public abstract Persona findById(int id);
21
22     public abstract List<Persona> findByPNombre(String pnombre);
23
24     public abstract List<Persona> findByEdad(int edad);
25
26     public abstract Page<Persona> findAll(Pageable pageable);
27
28 }
29

```

- Creación de los servicios que serán accedidos desde el controlador, para esto se define a través, de una interfaz los servicios que se implementarán posteriormente en la clase “PersonaServiceImpl” la cual se definirá con la anotación @Service.

Primero se crea el paquete “Services”



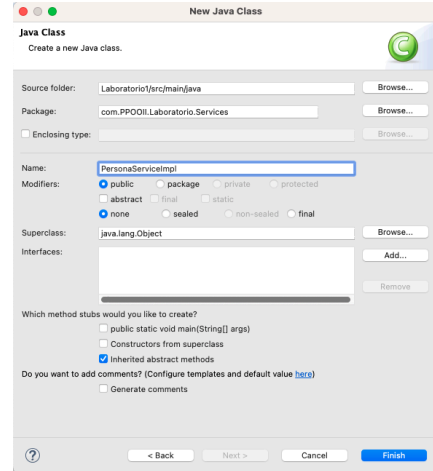
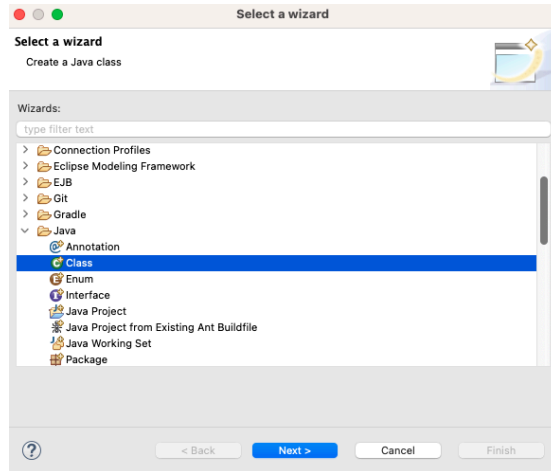
Se definen los métodos a implementar a través de la interfaz “IPersonaService”

```
application.properties Persona.java Laboratorio1/pom.xml PersonaRepository.java IPersonaService.java X
1 package com.PPOOII.Laboratorio.Services.Interfaces;
2
3 import java.util.List;
4
5
6 public interface IPersonaService {
7
8     //METODOS CRUD
9     public boolean guardar(Persona persona);
10    public boolean actualizar(Persona persona);
11    public boolean eliminar(int id);
12    public List<Persona> consultarPersona(Pageable pageable);
13
14    //LISTA DE PERSONA POR ID
15    public Persona findById(int id);
16
17    //LISTA DE PERSONA POR PRIMER NOMBRE
18    public List<Persona> findByNombre(String pnombre);
19
20    //LISTA DE PERSONA POR EDAD
21    public List<Persona> findByEdad(int edad);
22
23 }
24
25
26
27
28
```

Por último, se crea la clase “PersonaServiceImpl” la cual implementa los métodos definidos en la interfaz “IPersonaService”.

En esta clase se puede identificar la declaración de un atributo de tipo

“PersonaRepository”, el cual permitirá a los métodos implementados del servicio acceder a la capa de persistencia realizando operaciones de extracción de datos, creación, actualizaciones y eliminación.



```

application.properties  Persona.java  LaboratorioI/pom.xml  PersonaRepository.java  IPersonaService.java  PersonaServiceImpl.java x
1 package com.PP00II.Laboratorio.Services;
2
3 import java.util.List;
4
14
15 @Service("PersonaService")
16 public class PersonaServiceImpl implements IPersonaService{
17     // ===== INYECCIÓN DE DEPENDENCIAS =====
18     @Autowired
19     @Qualifier("IPersonaRepo")
20     private PersonaRepository IPersonaRepository;
21     //===== LOGS =====
22     //LOGS DE ERROR
23     private static final Logger logger = org.apache.logging.log4j.LogManager.getLogger(PersonaServiceImpl.class);
24     //INSERT
25     @Override
26     public boolean guardar(Persona persona) {
27         try {
28             if (persona == null) {
29                 logger.error("ERROR AGREGAR_PERSONA: LA PERSONA ES NULO!");
30                 return false;
31             }
32             else {
33                 IPersonaRepository.save(persona);
34                 return true;
35             }
36         }catch(Exception e) {
37             logger.error("ERROR AGREGAR_PERSONA: LA PERSONA NO SE HA GUARDADO!");
38             return false;
39         }
40     }
41     //UPDATE
42     @Override
43     public boolean actualizar(Persona persona) {
44         try {
45             if ((persona == null) || (persona.getId() == 0)) {
46                 logger.error("ERROR EDITAR_PERSONA: LA PERSONA ES NULO O EL ID ES 0!");
47                 return false;
48             }
49             else {
50                 IPersonaRepository.save(persona);
51                 return true;
52             }
53         }catch(Exception e) {
54             logger.error("ERROR EDITAR_PERSONA: LA PERSONA NO SE HA EDITADO!");
55             return false;
56         }
57     }
58     //DELETE
59     @Override
60     public boolean eliminar(int id) {
61         try {
62             if ((id == 0)) {
63                 logger.error("ERROR ELIMINAR_PERSONA: EL ID DEL PERSONA ES 0!");
64                 return false;
65             }
66             else {
67                 Persona persona = IPersonaRepository.findById(id);
68                 IPersonaRepository.delete(persona);
69                 return true;
70             }
71         }catch(Exception e) {
72             logger.error("ERROR ELIMINAR_PERSONA: LA PERSONA NO SE HA ELIMINADO!");
73             return false;
74         }
75     }
76     //LISTA DE PRODUCTOS
77     @Override
78     public List<Persona> consultarPersona(Pageable pageable) {
79         return IPersonaRepository.findAll(pageable).getContent();
80     }
81 }

```

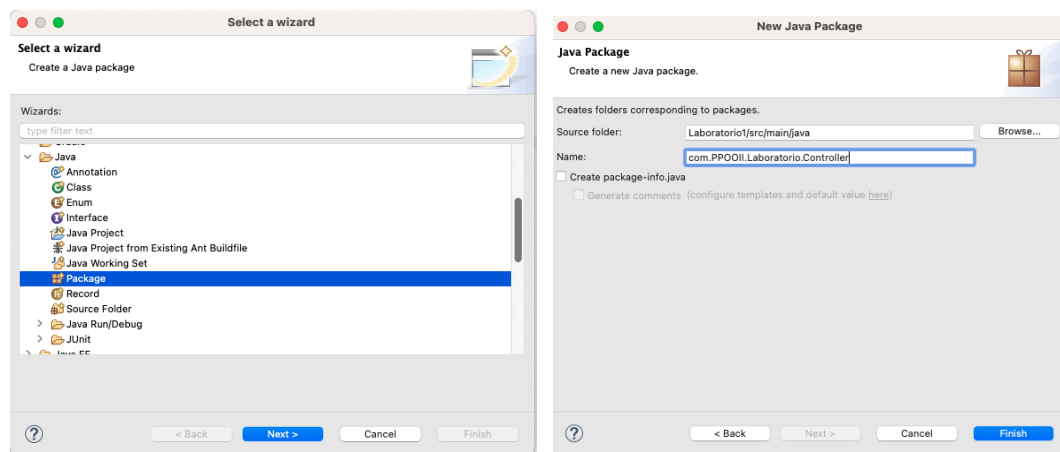
```

1 package com.PP00II.Laboratorio.Services;
2
3 import java.util.List;
4
14 @Service
15 public class PersonaServiceImpl implements IPersonaService{
16     // ===== INYECCIÓN DE DEPENDENCIAS =====
17     @Autowired
18     @Qualifier("IPersonaRepo")
19     private PersonaRepository IPersonaRepository;
20     //===== LOGS =====
21     //LOGS DE ERROR
22     private static final Logger logger = org.apache.logging.log4j.LogManager.getLogger(PersonaServiceImpl.class);
23     //INSERT
24     public boolean guardar(Persona persona) {}
25     //UPDATE
26     public boolean actualizar(Persona persona) {}
27     //DELETE
28     public boolean eliminar(int id) {}
29     //LISTA DE PRODUCTOS
30     public List<Persona> consultarPersona(Pageable pageable) {}
31
32     //===== METODOS DE BUSQUEDA =====
33     //PERSONA POR ID | VALOR UNICO
34     @Override
35     public Persona findById(int id) {
36         return IPersonaRepository.findById(id);
37     }
38
39     //LISTA DE PERSONAS POR NOMBRE
40     @Override
41     public List<Persona> findByNombre(String pnombre) {
42         return IPersonaRepository.findByPNombre(pnombre);
43     }
44
45     //LISTA DE PERSONAS POR EDAD
46     @Override
47     public List<Persona> findByEdad(int edad) {
48         return IPersonaRepository.findByEdad(edad);
49     }
50 }
51
52

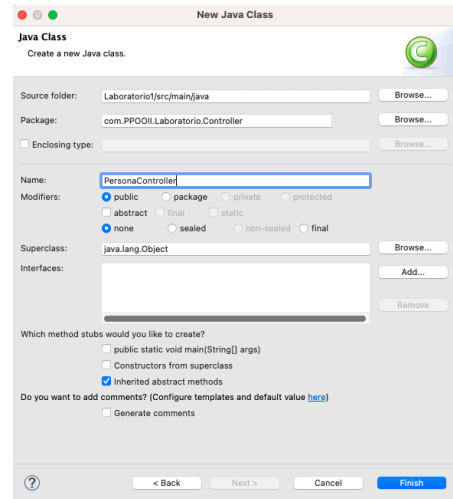
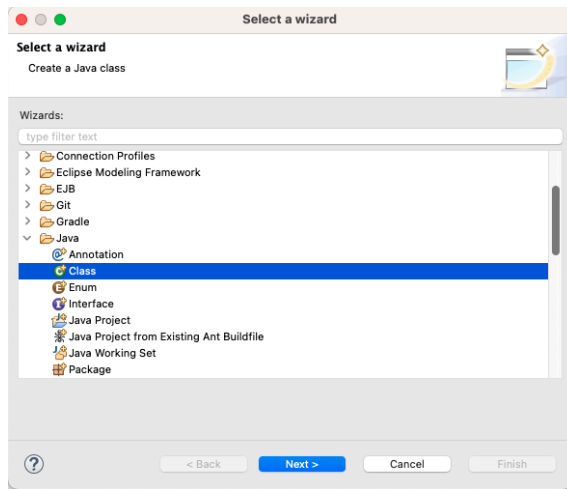
```

- Creación del controlador, el cual definirá las formas de acceso a los servicios implementados usando la anotación `@RestController`

Primero se crea el paquete “Controller”



Seguido a lo anterior se crea la clase “PersonaController”, en esta clase se crea un atributo de tipo “PersonaServiceImpl” con el cual se accede a los métodos implementados del servicio de persona. También se definen los diferentes métodos (GET, POST, PUT, DELETE) de acceso a los servicios.

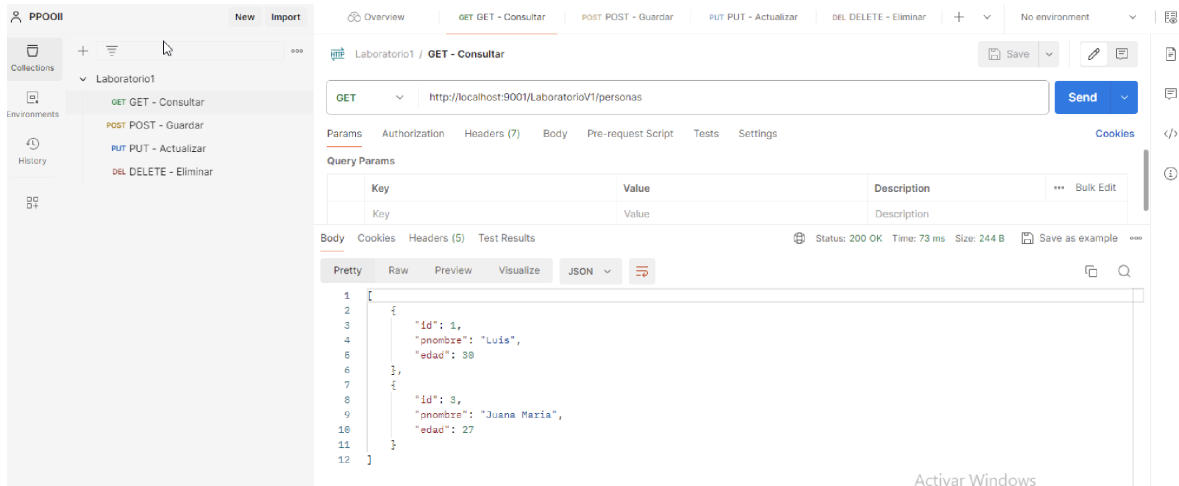


```

1 package com.PP00II.Laboratorio.Controller;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.beans.factory.annotation.Qualifier;
7 import org.springframework.data.domain.Pageable;
8 import org.springframework.validation.annotation.Validated;
9 import org.springframework.web.bind.annotation.DeleteMapping;
10 import org.springframework.web.bind.annotation.GetMapping;
11 import org.springframework.web.bind.annotation.PathVariable;
12 import org.springframework.web.bind.annotation.PostMapping;
13 import org.springframework.web.bind.annotation.PutMapping;
14 import org.springframework.web.bind.annotation.RequestBody;
15 import org.springframework.web.bind.annotation.RequestMapping;
16 import org.springframework.web.bind.annotation.RestController;
17
18 import com.PP00II.Laboratorio.Entities.Persona;
19 import com.PP00II.Laboratorio.Services.PersonaServiceImpl;
20
21 @RestController
22 @RequestMapping("/LaboratorioV1")
23 public class PersonaController {
24
25     // ===== INYECCION DEL SERVICE =====
26     @Autowired
27     @Qualifier("PersonaService")
28     PersonaServiceImpl personaService;
29
30     // ===== METODOS HTTP =====
31     // METODO POST
32     @PostMapping("/persona")
33     public boolean agregarPersona(@RequestBody @Validated Persona persona) {
34         return personaService.guardar(persona);
35     }
36     // METODO PUT
37     @PutMapping("/persona")
38     public boolean editarPersona(@RequestBody @Validated Persona persona) {
39         return personaService.actualizar(persona);
40     }
41     // METODO DELETE
42     @DeleteMapping("/persona/{id}")
43     public boolean eliminarPersona(@PathVariable("id") int id) {
44         return personaService.eliminar(id);
45     }
46     // METODO GET
47     @GetMapping("/personas")
48     public List<Persona> listadoPersona(Pageable pageable) {
49         return personaService.consultarPersona(pageable);
50     }
51
52     // ===== METODOS HTTP DE BÚSQUEDA =====
53     // ---GET---
54     @GetMapping("/persona/{id}/{id}")
55     public Persona getById(@PathVariable("id") int id) {
56         return personaService.findById(id);
57     }
58     // ---GET---
59     @GetMapping("/persona/pnombre/{pnombre}")
60     public List<Persona> getByNombre(@PathVariable("pnombre") String pnombre) {
61         return personaService.findByName(pnombre);
62     }
63     // ---GET---
64     @GetMapping("/persona/edad/{edad}")
65     public List<Persona> getByEdad(@PathVariable("edad") int edad) {
66         return personaService.findByEdad(edad);
67     }
68 }
69
70

```

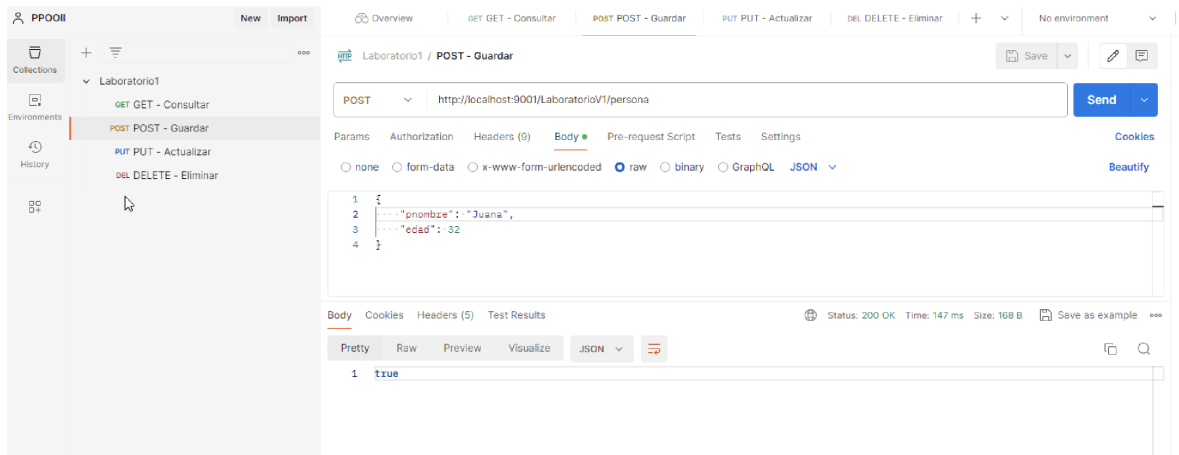
## Verificación acceso a los servicios con su respectivo método



Postman interface showing a GET request to `http://localhost:9001/LaboratorioV1/personas`. The response is a JSON array of two objects:

```
1 {
2   {
3     "id": 1,
4     "nombre": "Luis",
5     "edad": 38
6   },
7   {
8     "id": 2,
9     "nombre": "Juana Maria",
10    "edad": 27
11  }
12 }
```

Status: 200 OK Time: 73 ms Size: 244 B



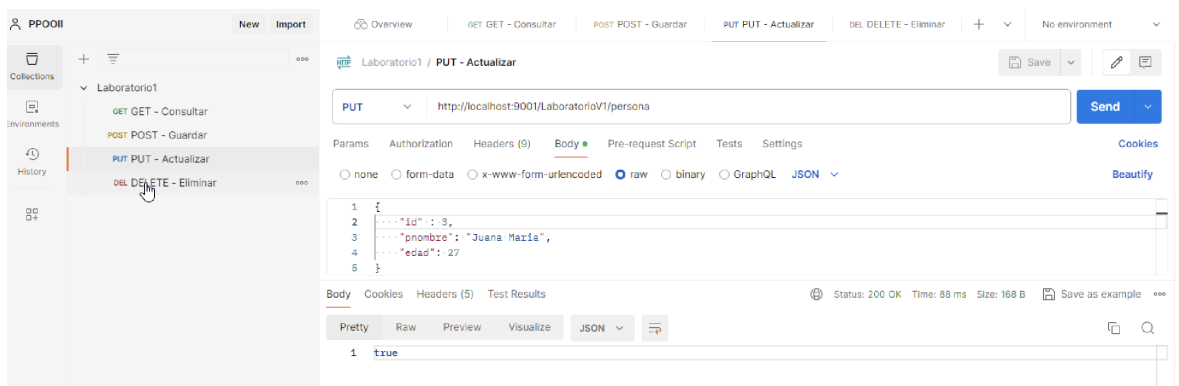
Postman interface showing a POST request to `http://localhost:9001/LaboratorioV1/persona`. The request body is a JSON object:

```
1 {
2   "nombre": "Juana",
3   "edad": 32
4 }
```

The response is a JSON object with the value `true`.

```
1 true
```

Status: 200 OK Time: 147 ms Size: 168 B



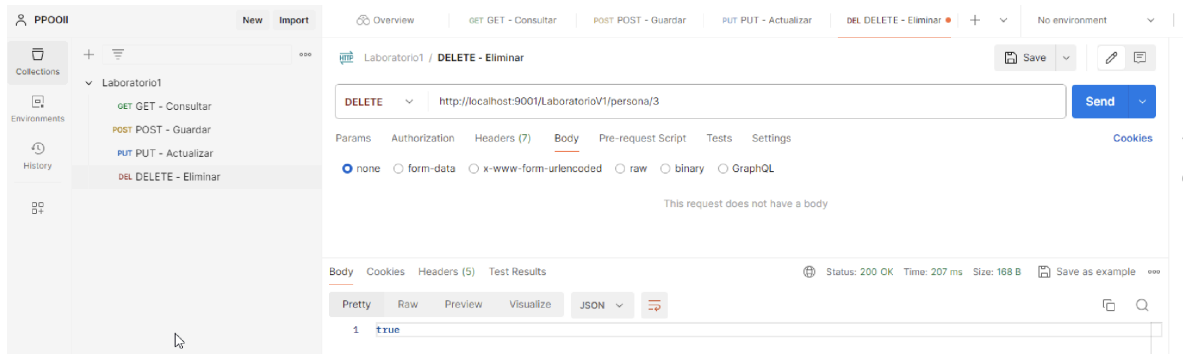
Postman interface showing a PUT request to `http://localhost:9001/LaboratorioV1/persona`. The request body is a JSON object:

```
1 {
2   "id": 3,
3   "nombre": "Juana Maria",
4   "edad": 27
5 }
```

The response is a JSON object with the value `true`.

```
1 true
```

Status: 200 OK Time: 88 ms Size: 168 B



Configurar y validar los demás servicios de consulta configurados en el controlador

```
// =====MÉTODOS HTTP DE BÚSQUEDA =====  
// ---GET---  
@GetMapping("/persona/id/{id}")  
public Persona getById(@PathVariable("id") int id) {  
    return personaService.findById(id);  
}  
// ---GET---  
@GetMapping("/persona/pnombre/{pnombre}")  
public List<Persona> getByPNombre(@PathVariable("pnombre") String pnombre) {  
    return personaService.findByName(pnombre);  
}  
// ---GET---  
@GetMapping("/persona/edad/{edad}")  
public List<Persona> getByEdad(@PathVariable("edad") int edad) {  
    return personaService.findByEdad(edad);  
}
```