



# **Programación Orientada A Objetos II**

**Jaime**

**UAZ**

Examen 3

Crud con Spring y Angular

Jesús Adrián Arias Delgado

```

C:\Users\Adrian Arias>npm install -g @angular/cli
npm WARN deprecated @npmcli/move-file@2.0.1: This functionality has been moved to @npmcli/fs
added 256 packages in 31s

37 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New minor version of npm available! 9.5.1 -> 9.6.7
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.7
npm notice Run npm install -g npm@9.6.7 to update!
npm notice

```

Primero instalamos Angular y luego creamos el proyecto con el siguiente comando.

```

C:\Users\Adrian Arias\Desktop\programas\P00II\angular>ng new crud
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. No
Global setting: disabled
Local setting: No local workspace configuration file.
Effective status: disabled
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE crud/angular.json (2690 bytes)
CREATE crud/package.json (1035 bytes)
CREATE crud/README.md (1058 bytes)
CREATE crud/tsconfig.json (901 bytes)
CREATE crud/.editorconfig (274 bytes)
CREATE crud/.gitignore (548 bytes)
CREATE crud/tsconfig.app.json (263 bytes)
CREATE crud/tsconfig.spec.json (273 bytes)
CREATE crud/.vscode/extensions.json (130 bytes)
CREATE crud/.vscode/launch.json (470 bytes)
CREATE crud/.vscode/tasks.json (938 bytes)
CREATE crud/src/main.ts (214 bytes)
CREATE crud/src/favicon.ico (948 bytes)
CREATE crud/src/index.html (290 bytes)

```

Con el comando ng serve podemos correr el servidor en el puerto 4200

```

C:\Users\Adrian Arias\Desktop\programas\P00II\angular\crud>"C:\Program Files\nodejs\npm.cmd" run start
> crud@0.0.0 start
> ng serve

✓ Browser application bundle generation complete.

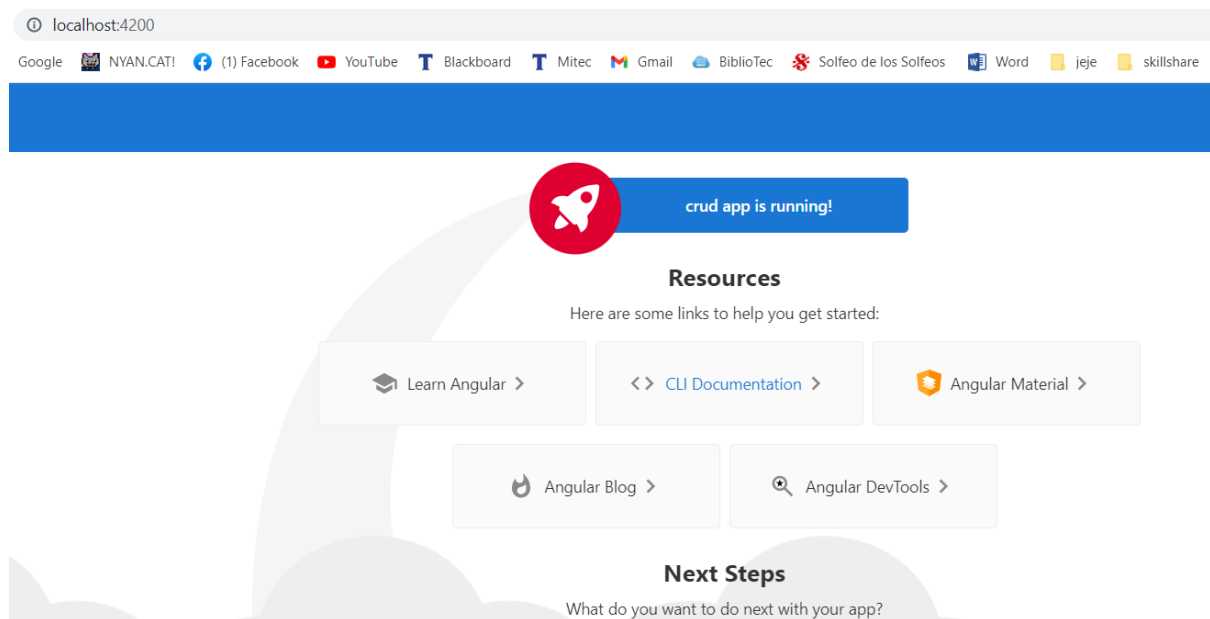
Initial Chunk Files | Names          | Raw Size
vendor.js           | vendor        | 2.26 MB |
polyfills.js        | polyfills     | 330.89 kB |
styles.css, styles.js | styles       | 228.32 kB |
runtime.js          | runtime       | 6.51 kB |
main.js             | main          | 6.47 kB |
                    | Initial Total | 2.82 MB

Build at: 2023-06-05T22:07:40.264Z - Hash: 45fe8107085f369c - Time: 1961ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/

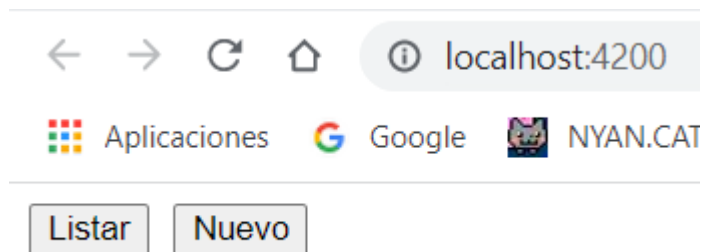
✓ Compiled successfully.
¿Desea terminar el trabajo por lotes (S/N)? S

```



Para editar el front end tenemos que modificar el archivo componente html que sirve como plantilla. Nótese que los métodos aparecen en rojo porque aun no han sido definidos.

```
<div class="content" role="main">
  <button (click)="Listar()" class="btn btn-info">Listar</button>
  <button (click)="Nuevo()" class="btn btn-info" style="margin-left:10px;">Nuevo</button>
</div>
<router-outlet></router-outlet>
```



Luego creamos los componentes del proyecto

```

C:\Users\Adrian Arias\Desktop\programas\P00II\angular\crud>ng g c Persona/listar

CREATE src/app/Persona/listar/listar.component.html (21 bytes)
CREATE src/app/Persona/listar/listar.component.spec.ts (559 bytes)
CREATE src/app/Persona/listar/listar.component.ts (202 bytes)CREATE src/app/Pers
ona/listar/listar.component.css (0 bytes)
UPDATE src/app/app.module.ts (483 bytes)

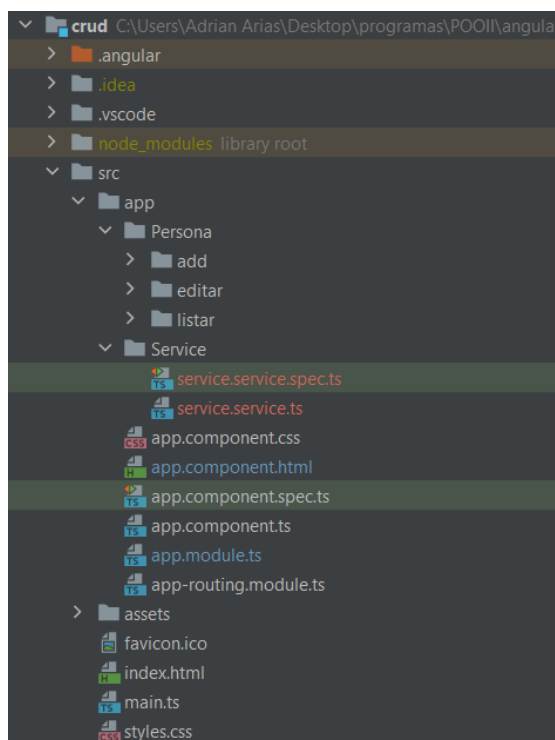
C:\Users\Adrian Arias\Desktop\programas\P00II\angular\crud>ng g c Persona/add
CREATE src/app/Persona/add/add.component.html (18 bytes)
CREATE src/app/Persona/add/add.component.spec.ts (538 bytes)
CREATE src/app/Persona/add/add.component.ts (190 bytes)
CREATE src/app/Persona/add/add.component.css (0 bytes)
UPDATE src/app/app.module.ts (561 bytes)

C:\Users\Adrian Arias\Desktop\programas\P00II\angular\crud>ng g c Persona/editar
CREATE src/app/Persona/editar/editar.component.html (21 bytes)
CREATE src/app/Persona/editar/editar.component.spec.ts (559 bytes)
CREATE src/app/Persona/editar/editar.component.ts (202 bytes)CREATE src/app/Pers
ona/editar/editar.component.css (0 bytes)
UPDATE src/app/app.module.ts (651 bytes)

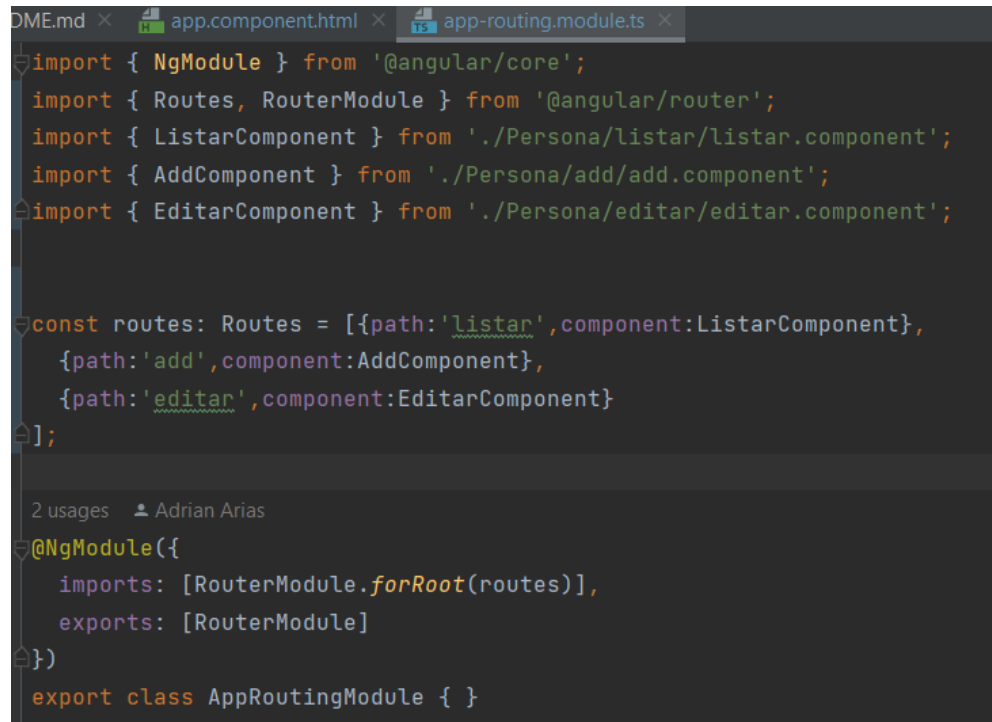
C:\Users\Adrian Arias\Desktop\programas\P00II\angular\crud>ng s Service/service
Error: Invalid values:
  Argument: project, Given: "Service/service", Choices: "crud"

C:\Users\Adrian Arias\Desktop\programas\P00II\angular\crud>ng g s Service/servic
e
CREATE src/app/Service/service.service.spec.ts (362 bytes)
CREATE src/app/Service/service.service.ts (136 bytes)

```



Para poder utilizar estos componentes los ponemos en el modulo de enrutamiento para poder hacer referencia a ellos.

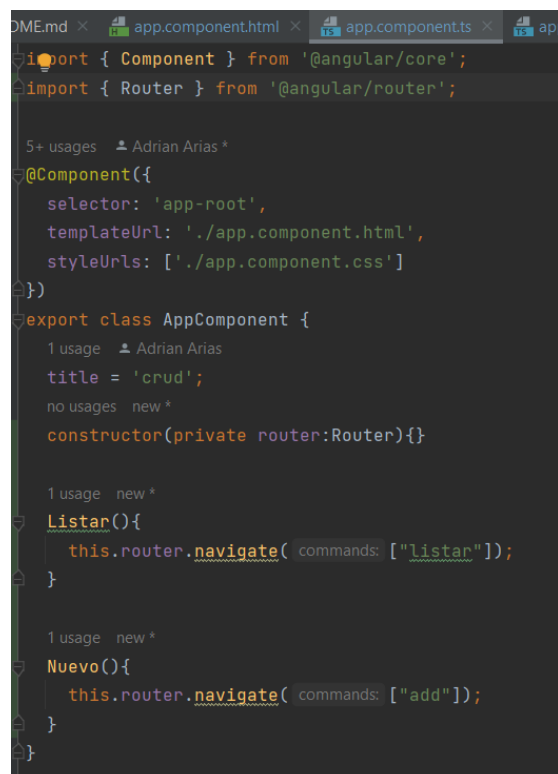


```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { ListarComponent } from '../Persona/listar/listar.component';
import { AddComponent } from '../Persona/add/add.component';
import { EditarComponent } from '../Persona/editar/editar.component';

const routes: Routes = [{path:'listar',component:ListarComponent},
  {path:'add',component:AddComponent},
  {path:'editar',component:EditarComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Una vez que los tenemos en la clase router podemos hacer uso de estos componentes desde una instancia creada en app.components.ts



```
import { Component } from '@angular/core';
import { Router } from '@angular/router';

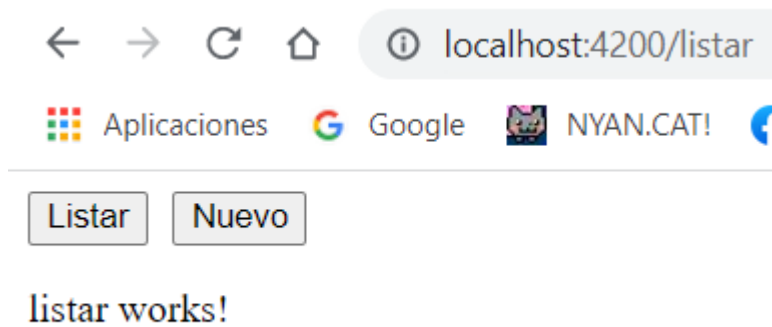
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'crud';

  constructor(private router:Router){}

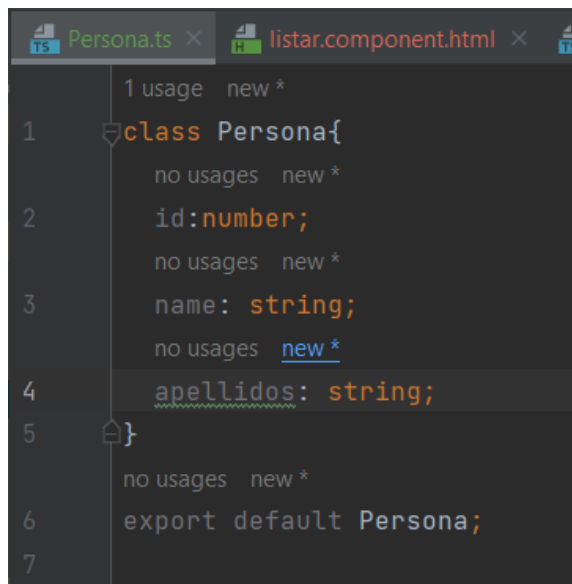
  Listar(){
    this.router.navigate( commands: ["listar"]);
  }

  Nuevo(){
    this.router.navigate( commands: ["add"]);
  }
}
```

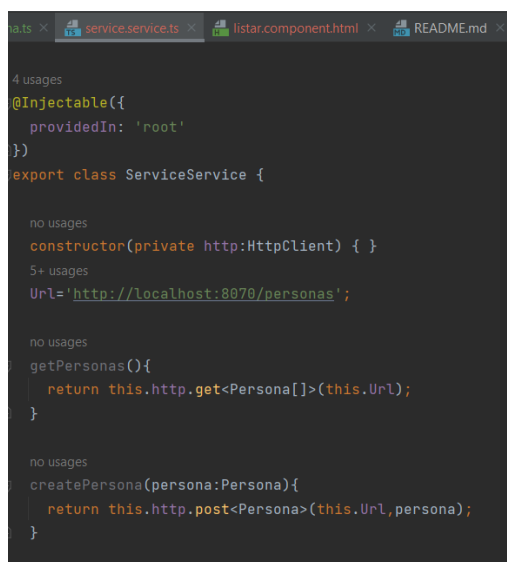
Con esto ya deberían de funcionar los botones



Ahora crearemos una carpeta llamada Modelo que tendrá un archivo Persona.ts que nos indica la estructura de Persona



Este modelo lo podremos usar en el servicio.servicio.ts que nos permitirá hacer las solicitudes http.



Creamos nuestra base de datos

```
mysql> CREATE DATABASE crudo;
Query OK, 1 row affected (0.04 sec)

mysql> Use crudo
Database changed
mysql> CREATE TABLE persona(
  -> id INT PRIMARY KEY,
  -> nombre VARCHAR(45),
  -> apellidos VARCHAR(45));
Query OK, 0 rows affected (0.06 sec)

mysql>
```

Luego utilizaremos Spring para conectarlo a la base de datos para crear el backend, así que crearemos un proyecto usando la siguiente página.

The screenshot shows the start.spring.io web interface. On the left, under 'Project', 'Maven' is selected. Under 'Language', 'Java' is selected. Under 'Spring Boot', '3.1.0' is selected. The 'Project Metadata' section has fields for Group (com.adri), Artifact (crud), Name (crud), Description (Demo project for Spring Boot), and Package name (com.adri.crud). The 'Packaging' is set to 'Jar' and the 'Java' version is '17'. On the right, under 'Dependencies', 'Spring Web' (WEB), 'Spring Data JPA' (SQL), and 'MySQL Driver' (SQL) are listed.

Editamos las propiedades de la app de Spring para que se conecte a la base de datos

```
application.properties
1  server.contextPath=/rest
2  spring.datasource.url=jdbc:mysql://localhost:3306/crudo?serverTimezone=UTC
3  spring.datasource.username=root
4  spring.datasource.password=
5  spring.datasource.driver-class-name=com.mysql.jdbc.Driver
6  server.port = 8070
7
```

Añadimos la clase Persona con sus respectivas anotaciones de Spring

```
application.properties Persona.java Controlador.java PersonaRepositorio.java PersonaService.java
package com.adri.crud;
import javax.persistence.*;

28 usages
@Entity
@Table(name="persona")
public class Persona {

    2 usages
    @Id @Column @GeneratedValue(strategy=GenerationType.IDENTITY) private int id;

    2 usages
    @Column private String name;

    2 usages
    @Column private String apellidos;

    no usages
    public int getId() {
        return id;
    }

    1 usage
    public void setId(int id) {
        this.id = id;
    }

    no usages
    public String getName() {
        return name;
    }

    no usages
    public void setName(String name) {
```

Controlador

```
application.properties Persona.java Controlador.java PersonaRepositorio.java PersonaService.java
@RestController
@CrossOrigin(origins = "http://localhost:4200",maxAge = 3600)
@RequestMapping({"/personas"})
public class Controlador {

    5 usages
    @Autowired
    PersonaService service;

    no usages
    @GetMapping
    public List<Persona> listar(){
        return service.listar();
    }

    no usages
    @PostMapping
    public Persona agregar(@RequestBody Persona p){
        return service.add(p);
    }

    no usages
    @GetMapping(path={"/{id}"})
    public Persona listarID(@PathVariable("id") int id){
        return service.listarId(id);
    }

    no usages
    @PutMapping(path = {"/{id}"})
    public Persona editar(@RequestBody Persona p,@PathVariable("id") int id){
        p.setId(id);
        return service.edit(p);
    }

    no usages
```



## Persona repositorio

```
package com.adri.crud;

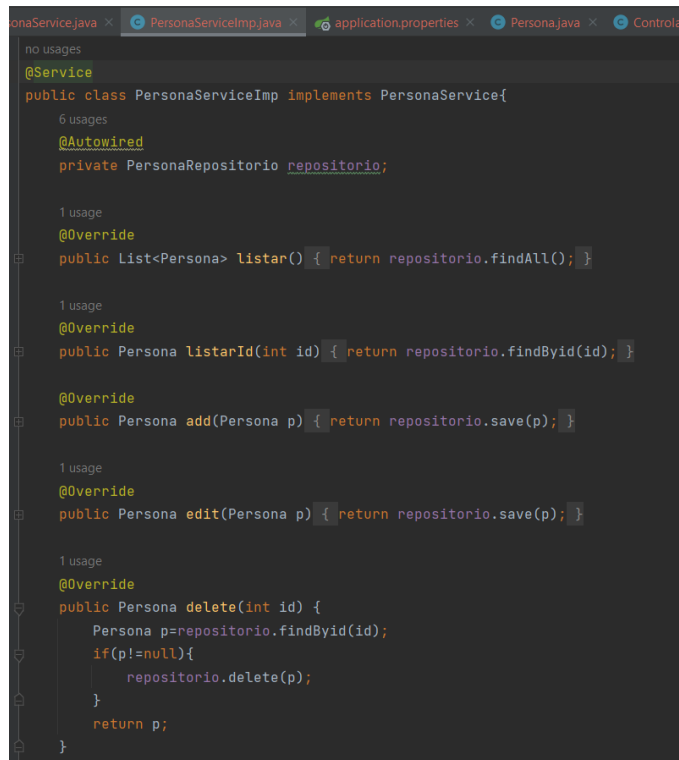
import java.util.List;
import org.springframework.data.repository.Repository;

1 usage
public interface PersonaRepositorio extends Repository<Persona, Integer>{
    //para listar todas las personas
    1 usage
    List<Persona>findAll();
    //para listar una persona
    2 usages
    Persona findById(int id);
    //para guardar cambios nuevos o actualizar
    2 usages
    Persona save(Persona p);
    //para eliminar
    1 usage
    void delete(Persona p);
}
```

## Persona Service

```
1 package com.adri.crud;
2 import java.util.List;
3
4 2 usages 1 implementation
5 public interface PersonaService {
6     // objeto de tipo List
7     1 usage 1 implementation
8     List<Persona>listar();
9     // objeto de tipo persona
10    1 usage 1 implementation
11    Persona listarId(int id);
12    1 implementation
13    Persona add(Persona p);
14    1 usage 1 implementation
15    Persona edit(Persona p);
16    1 usage 1 implementation
17    Persona delete(int id);
18 }
```

## Y PersonaServiceImp



The screenshot shows an IDE window titled 'PersonaServiceImp.java' with the following Java code:

```
no usages
@Service
public class PersonaServiceImp implements PersonaService{

    6 usages
    @Autowired
    private PersonaRepositorio repositorio;

    1 usage
    @Override
    public List<Persona> listar() { return repositorio.findAll(); }

    1 usage
    @Override
    public Persona listarId(int id) { return repositorio.findById(id); }

    @Override
    public Persona add(Persona p) { return repositorio.save(p); }

    1 usage
    @Override
    public Persona edit(Persona p) { return repositorio.save(p); }

    1 usage
    @Override
    public Persona delete(int id) {
        Persona p=repositorio.findById(id);
        if(p!=null){
            repositorio.delete(p);
        }
        return p;
    }
}
```

The code implements the `PersonaService` interface. It includes an `@Autowired` field `repositorio` of type `PersonaRepositorio`. The methods `listar()`, `listarId(int id)`, `add(Persona p)`, `edit(Persona p)`, and `delete(int id)` are all overridden to delegate the logic to the `repositorio` object. The `delete` method includes a null check before attempting to delete the entity.