

Clase 12

JHipster

Instalación

JHipster requiere una versión estable de nodeJS, y JAVA

NVM (<https://lenguajejs.com/npm/introduccion/instalacion-node-con-nvm/>)

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh  
| bash
```

Node version 16

```
nvm install 16
```

JHipster (<https://www.jhipster.tech/installation/>)

Instalamos la última versión estable

```
npm install -g generator-jhipster@7.9.3
```

Creación de aplicación

X jhipster

INFO! Using bundled JHipster

? Which **type** of application would you like to create? Monolithic application (recommended for simple projects)
? What is the base name of your application? demo1
? Do you want to make it reactive with Spring WebFlux? No
? What is your default Java package name? ar.edu.um.programacion2.fernando
? Which **type** of authentication would you like to use? JWT authentication (stateless, with a token)
? Which **type** of database would you like to use? SQL (H2, PostgreSQL, MySQL, MariaDB, Oracle, MSSQL)
? Which **production** database would you like to use? MariaDB
? Which **development** database would you like to use? H2 with disk-based persistence
? Which cache do you want to use? (Spring cache abstraction) Ehcache (local cache, for a single node)
? Do you want to use Hibernate 2nd level cache? Yes
? Would you like to use Maven or Gradle for building the backend? Maven
? Do you want to use the JHipster Registry to configure, monitor and scale your application? Yes
? Which other technologies would you like to use?
? Which **Framework** would you like to use for the client? React
? Do you want to generate the admin UI? Yes
? Would you like to use a Bootswatch theme (<https://bootswatch.com/>)? Default JHipster
? Would you like to enable internationalization support? Yes
? Please choose the native language of the application Spanish
? Please choose additional languages to install English, Italian
? Besides JUnit and Jest, which testing frameworks would you like to use?
? Would you like to install other generators from the JHipster Marketplace? No

Entidades

Página principal

<https://www.jhipster.tech/>

Creación de entidades

<https://www.jhipster.tech/creating-an-entity/>

Relaciones

<https://www.jhipster.tech/managing-relationships/>

JDL

<https://www.jhipster.tech/jdl/intro>

<https://www.jhipster.tech/jdl/relationships>

JDL Studio

<https://start.jhipster.tech/jdl-studio/>

JDL

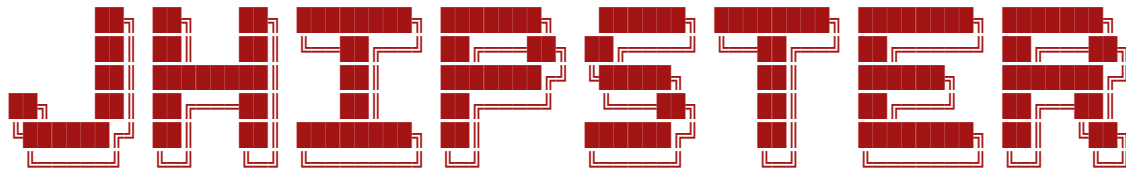
```
application {
  config {
    applicationType monolith
    authenticationType jwt
    baseName demol
    blueprints []
    buildTool maven
    cacheProvider ehcache
    clientFramework react
    clientPackageManager npm
    clientTheme none
    creationTimestamp 1693866284375
    databaseType sql
    devDatabaseType h2Disk
    dtoSuffix DTO
    enableGradleEnterprise false
    enableHibernateCache true
    enableSwaggerCodegen false
    enableTranslation true
    gradleEnterpriseHost ""
    jhiPrefix jhi
    jhipsterVersion "7.9.3"
    jwtSecretKey "MjE0ZmViZjFkMGZjNDJhYjZjOTEwYT..."
    languages [es, en, it]
    messageBroker false
    microfrontend false
    microfrontends []
    nativeLanguage es
```

```
    otherModules []
    packageName ar.edu.um.programacion2.fernando
    prodDatabaseType mariadb
    reactive false
    searchEngine false
    serverPort 8080
    serviceDiscoveryType eureka
    websocket false
    withAdminUi true
  }
  entities Automovil, Chofer
}
entity Automovil {
  marca String
  modelo String
  patente String pattern(/^[a-zA-Z]{2}[0-9]{3}[a-zA-Z]...)
}
entity Chofer {
  nombre String
  apellido String
  aniosChofer Integer
}
relationship ManyToOne {
  Automovil{chofer} to Chofer
}

paginate Automovil, Chofer with infinite-scroll
service Automovil, Chofer with serviceImpl
```

Crear / actualizar proyecto desde un JDL

```
jhipster import-jdl pruebal.jdl
INFO! Switching to JHipster installed locally in current project's node repository (node modules)
(node:98439) [DEP0148] DeprecationWarning: Use of deprecated folder mapping "./lib/util/" in the "exports" field module
resolution of the package at
/Users/Villano/Documents/GitHub/UMProgramacion1-2023-ejemplos/programacion2-2023-ejemplos/jhipster-demo1/node_modules/yeo
man-environment/package.json.
Update this package.json to use a subpath pattern like "./lib/util/*".
(Use `node --trace-deprecation ...` to show where the warning was created)
```



<https://www.jhipster.tech>

Welcome to JHipster v7.9.3

```
INFO! Executing import-jdl pruebal.jdl
INFO! The JDL is being parsed.
INFO! Found entities: Perro6.
INFO! The JDL has been successfully parsed
INFO! Generating 1 application.
Application files will be generated in folder:
/Users/Villano/Documents/GitHub/UMProgramacion1-2023-ejemplos/programacion2-2023-ejemplos/jhipster-demo1
```

Relaciones - OneToMany bidireccional

```
/**
 * Bidirectional One to many relationship.
 * Persona (1) <-----> (*) Perro
 */
entity Persona {
    nombre String
    apellido String
}
entity Perro {
    nombre String
    sobreNombre String
}
relationship OneToMany {
    Persona{perro} to Perro{persona}
}
```

```
localhost:8080/api/personas/
[
  {
    "id": 1001,
    "nombre": "Fernando",
    "apellido": "Garcia",
    "perros": null
  }
]
```

```
localhost:8080/api/perros/
[
  {
    "id": 1051,
    "nombre": "Firulais",
    "sobreNombre": "Firulais",
    "persona": {
      "id": 1001,
      "nombre": "Fernando",
      "apellido": "Garcia"
    }
  }
]
```

Relaciones

Development

Demo10.0.1-SNAPSHOT

InicioEntidadesAdministraciónEspañolCuenta

Personas

Refrescar listaCrear nuevo Persona

ID	Nombre	Apellido
1001	Fernando	García

VistaEditarEliminar

Pie de página

Crear o editar Persona

Nombre

Apellido

Volver

Guardar

Development

Demo10.0.1-SNAPSHOT

InicioEntidadesAdministraciónEspañolCuenta

Perros

Refrescar listaCrear nuevo Perro

ID	Nombre	Sobre Nombre	Persona
1001	Firulais	Firulais	1001

VistaEditarEliminar

Pie de página

Crear o editar Perro

Nombre

Sobre Nombre

Persona

✓

1001

Volver

Guardar

Relaciones - ManyToOne bidireccional

```
/**
 * Bidirectional many to one relationship.
 * Persona (1) <-----> (*) Perro
 */
entity Personal {
    nombre String
    apellido String
}
entity Perro1 {
    nombre String
    sobreNombre String
}
relationship ManyToOne {
    Perro1{personal} to Personal{perro1}
}
```

```
localhost:8080/api/persona-1-s/
[
  {
    "id": 1601,
    "nombre": "Arturo",
    "apellido": "Galvez",
    "perro1s": null
  }
]
```

```
localhost:8080/api/perro-1-s/
[
  {
    "id": 1651,
    "nombre": "Sultán",
    "sobreNombre": "Sultán",
    "personal": {
      "id": 1601,
      "nombre": "Arturo",
      "apellido": "Galvez"
    }
  }
]
```

Relaciones - corregir

Dev Demo1

Persona 1 S

[Refrescar lista](#) [+ Crear nuevo Persona 1](#)

ID	Nombre	Apellido
1601	Arturo	Galvez

[Vista](#) [Editar](#) [Eliminar](#)

Pie de página

Crear o editar Persona 1

Nombre

Apellido

[← Volver](#)[Guardar](#)

Dev Demo1 0.0.1-SNAPSHOT

[Inicio](#) [Entidades](#) [Administración](#) [Español](#) [Cuenta](#)

Perro 1 S

[Refrescar lista](#) [+ Crear nuevo Perro 1](#)

ID	Nombre	Sobre Nombre	Persona 1
1651	Sultán	Sultán	1601

[Vista](#) [Editar](#) [Eliminar](#)

Pie de página

Crear o editar Perro 1

Nombre

Sobre Nombre

Persona 1

✓
1601

[← Volver](#)[Guardar](#)

Relaciones - ManyToOne unidireccional

```
/**
 * Unidirectional Many to One relationship.
 * Persona (1) <----- (*) Perro
 */
entity Persona2 {
    nombre String
    apellido String
}
entity Perro2 {
    nombre String
    sobreNombre String
}
relationship ManyToOne {
    Perro2{persona2} to Persona2
}
```

```
localhost:8080/api/persona-2-s/
[
  {
    "id": 1101,
    "nombre": "Juan",
    "apellido": "Gomez"
  }
]
```

```
localhost:8080/api/perro-2-s/
[
  {
    "id": 1151,
    "nombre": "Bobby",
    "sobreNombre": "Bobby",
    "persona2": {
      "id": 1101,
      "nombre": "Juan",
      "apellido": "Gomez"
    }
  }
]
```

Relaciones

Development

Demo1
0.0.1-SNAPSHOT

[Inicio](#) [Entidades](#) [Administración](#) [Español](#) [Cuenta](#)

Persona 2 S

[Refrescar lista](#) [+ Crear nuevo Persona 2](#)

ID	Nombre	Apellido
1101	Juan	Gomez

[Vista](#) [Editar](#) [Eliminar](#)

Pie de página

Crear o editar Persona 2

Nombre

Apellido

[← Volver](#)[Guardar](#)

Development

Demo1
0.0.1-SNAPSHOT

[Inicio](#) [Entidades](#) [Administración](#) [Español](#) [Cuenta](#)

Perro 2 S

[Refrescar lista](#) [+ Crear nuevo Perro 2](#)

ID	Nombre	Sobre Nombre	Persona 2
1151	Bobby	Bobby	1101

[Vista](#) [Editar](#) [Eliminar](#)

Pie de página

Crear o editar Perro 2

Nombre

Sobre Nombre

Persona 2

☒

1101

[← Volver](#)[Guardar](#)

Relaciones - ManyToMany bidireccional

```
/**
 * Bidirectional Many to Many relationship.
 * Persona (*) <-----> (*) Perro
 */
entity Persona3 {
    nombre String
    apellido String
}
entity Perro3 {
    nombre String
    sobreNombre String
}
relationship ManyToMany {
    Perro3{persona3} to Persona3{perro3}
}
```

```
localhost:8080/api/persona-3-s/
[
  {
    "id": 1201,
    "nombre": "Carlos",
    "apellido": "Alcalde",
    "perro3s": null
  },
  {
    "id": 1202,
    "nombre": "Juan",
    "apellido": "Robles",
    "perro3s": null
  }
]
```

```
localhost:8080/api/perro-3-s/
[
  {
    "id": 1251,
    "nombre": "Fluffy",
    "sobreNombre": "Fluffy",
    "persona3s": null
  }
]
```

Relaciones

Demo1 0.0.1-SNAPSHOT

Inicio Entidades Administración Español Cuenta

Persona 3 S

Refrescar lista + Crear nuevo Persona 3

ID	Nombre	Apellido	
1201	Carlos	Alcalde	Ver Editar Eliminar
1202	Juan	Robles	Ver Editar Eliminar

Pie de página

Crear o editar Persona 3

Nombre

Apellido

[← Volver](#)

[Guardar](#)

Demo1 0.0.1-SNAPSHOT

Inicio Entidades Administración Español Cuenta

Perro 3 S

Refrescar lista + Crear nuevo Perro 3

ID	Nombre	Sobre Nombre	Persona 3	
1251	Fluffy	Fluffy		Ver Editar Eliminar

Pie de página

Crear o editar Perro 3

Nombre

Sobre Nombre

Persona 3

1201

1202

Relaciones - OneToOne bidireccional

```
/**
 * Bidirectional One to One relationship.
 * Persona (1) <-----> (1) Perro
 */
entity Persona4 {
    nombre String
    apellido String
}

entity Perro4 {
    nombre String
    sobreNombre String
}

relationship OneToOne {
    Perro4{persona4} to Persona4{perro4}
}
```

```
localhost:8080/api/persona-4-s/
[
  {
    "id": 1301,
    "nombre": "Fernando",
    "apellido": "Alvarez",
    "perro4": {
      "id": 1351,
      "nombre": "Sultan",
      "sobreNombre": "Sultan"
    }
  }
]
```

```
localhost:8080/api/perro-4-s/
[
  {
    "id": 1351,
    "nombre": "Sultan",
    "sobreNombre": "Sultan",
    "persona4": {
      "id": 1301,
      "nombre": "Fernando",
      "apellido": "Alvarez"
    }
  }
]
```

Relaciones

Development

Demo10.0.1-SNAPSHOT

[Inicio](#) [Entidades](#) [Administración](#) [Español](#) [Cuenta](#)

Persona 4 S

Refrescar lista

+ Crear nuevo Persona 4

ID	Nombre	Apellido	
1301	Fernando	Alvarez	<div>Vista Editar Eliminar</div>

Pie de página

Crear o editar Persona 4

Nombre

Apellido

[← Volver](#)

[Guardar](#)

Development

Demo10.0.1-SNAPSHOT

[Inicio](#) [Entidades](#) [Administración](#) [Español](#) [Cuenta](#)

Perro 4 S

Refrescar lista

+ Crear nuevo Perro 4

ID	Nombre	Sobre Nombre	Persona 4	
1351	Sultan	Sultan	1301	<div>Vista Editar Eliminar</div>

Pie de página

Crear o editar Perro 4

Nombre

Sobre Nombre

Persona 4

✓

1301

[← Volver](#)

[Guardar](#)

Relaciones - OneToOne unidireccional

```
/**
 * Unidirectional One to One relationship.
 * Persona (1) -----> (1) Perro
 */
entity Persona5 {
    nombre String
    apellido String
}
entity Perro5 {
    nombre String
    sobreNombre String
}
relationship OneToOne {
    Perro5{persona5} to Persona5
}
```

```
localhost:8080/api/persona-5-s/
[
  {
    "id": 1401,
    "nombre": "Roberto",
    "apellido": "Guillermo"
  }
]
```

```
localhost:8080/api/perro-5-s/
[
  {
    "id": 1451,
    "nombre": "Destructor",
    "sobreNombre": "Destructor",
    "persona5": {
      "id": 1401,
      "nombre": "Roberto",
      "apellido": "Guillermo"
    }
  }
]
```

Relaciones

Development

Demo10.0.1-SNAPSHOT

[Inicio](#) [Entidades](#) [Administración](#) [Español](#) [Cuenta](#)

Persona 5 S

Refrescar lista

+ Crear nuevo Persona 5

ID	Nombre	Apellido	
1401	Roberto	Guillermi	<div>Vista Editar Eliminar</div>

Pie de página

Development

Demo10.0.1-SNAPSHOT

[Inicio](#) [Entidades](#) [Administración](#) [Español](#) [Cuenta](#)

Perro 5 S

Refrescar lista

+ Crear nuevo Perro 5

ID	Nombre	Sobre Nombre	Persona 5	
1451	Destructor	Destructor	1401	<div>Vista Editar Eliminar</div>

Pie de página

Crear o editar Persona 5

Nombre

Apellido

[← Volver](#)

[Guardar](#)

Crear o editar Perro 5

Nombre

Sobre Nombre

Persona 5

✓

1401

[← Volver](#)

[Guardar](#)

Relaciones - ManyToOne bidireccional

```
/**
 * Bidirectional many to one relationship.
 * Persona (1) <-----> (*) Perro
 */
entity Persona6 {
    nombre String
    apellido String
}
entity Perro6 {
    nombre String
    sobreNombre String
}
relationship ManyToOne {
    Perro6{persona(nombre)} to
    Persona6{perro(sobreNombre)}
}
```

```
localhost:8080/api/persona-6-s/
[
  {
    "id": 1501,
    "nombre": "Alvaro",
    "apellido": "Gonzalez",
    "perros": null
  }
]
```

```
localhost:8080/api/perro-6-s/
[
  {
    "id": 1551,
    "nombre": "Bobby",
    "sobreNombre": "Bobby",
    "persona": {
      "id": 1501,
      "nombre": "Alvaro",
      "apellido": "Gonzalez"
    }
  }
]
```

Relaciones

Development

Demo1 0.0.1-SNAPSHOT

[Inicio](#) [Entidades](#) [Administración](#) [Español](#) [Cuenta](#)

Persona 6 S

[Refrescar lista](#) [+ Crear nuevo Persona 6](#)

ID	Nombre	Apellido
1501	Alvaro	Gonzalez

[Vista](#) [Editar](#) [Eliminar](#)

Development

Demo1 0.0.1-SNAPSHOT

[Inicio](#) [Entidades](#) [Administración](#) [Español](#) [Cuenta](#)

Perro 6 S

[Refrescar lista](#) [+ Crear nuevo Perro 6](#)

ID	Nombre	Sobre Nombre	Persona
1551	Bobby	Bobby	Alvaro

[Vista](#) [Editar](#) [Eliminar](#)

Pie de página

Crear o editar Persona 6

Nombre

Apellido

[← Volver](#)

[Guardar](#)

Crear o editar Perro 6

Nombre

Sobre Nombre

Persona

☒

[Alvaro](#)

[← Volver](#)

[Guardar](#)

Fetching data strategy

Por defecto las relaciones tienen una estrategia por defecto para obtener los datos en las relaciones

- OneToMany: LAZY
- ManyToOne: EAGER
- ManyToMany: LAZY
- OneToOne: EAGER

Para cambiarlo hay que modificar la función que recupera los datos

```
@OneToMany(mappedBy = "parent", fetch = FetchType.EAGER)
```

```
private Set<Child> child = new HashSet<>();
```

A tener en cuenta a la hora del desarrollo

Creación del proyecto

Hay que crearlo en una carpeta vacía.

JHipster crea la estructura de directorios del proyecto completo con todos los archivos necesarios.

Listo para empezar a desarrollar

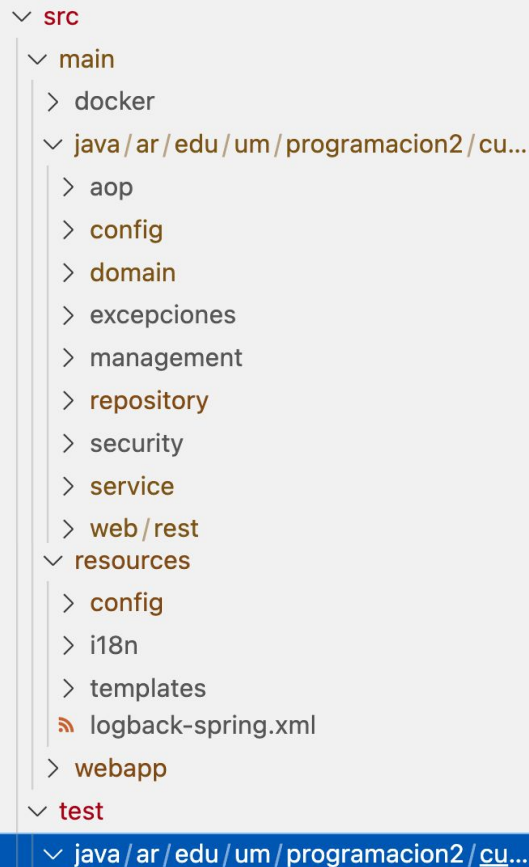
Crea automáticamente un repositorio git dentro de la carpeta.

Para crear entidades: **jhipster entity Persona**

Para crear controladores: **jhipster spring-controller Operaciones**

Para crear servicios: **jhipster service ManejoSesion**

A tener en cuenta a la hora del desarrollo



A screenshot of a file explorer window showing a project directory structure. The root directory is 'src', which is expanded to show 'main' and 'test'. The 'main' directory is further expanded to show 'docker', 'java/ar/edu/um/programacion2/cu...', 'aop', 'config', 'domain', 'excepciones', 'management', 'repository', 'security', 'service', 'web/rest', 'resources', and 'webapp'. The 'resources' directory is expanded to show 'config', 'i18n', 'templates', 'logback-spring.xml', and 'webapp'. The 'test' directory is also expanded to show 'java/ar/edu/um/programacion2/cu...'. The file explorer has a light gray background and a blue header bar.

```
src
├── main
│   ├── docker
│   ├── java/ar/edu/um/programacion2/cu...
│   ├── aop
│   ├── config
│   ├── domain
│   ├── excepciones
│   ├── management
│   ├── repository
│   ├── security
│   ├── service
│   ├── web/rest
│   ├── resources
│   │   ├── config
│   │   ├── i18n
│   │   ├── templates
│   │   ├── logback-spring.xml
│   │   └── webapp
│   └── webapp
└── test
    ├── java/ar/edu/um/programacion2/cu...
```

Estructura de directorios

/src/main/java: proyecto JAVA

/src/main/java/.../config/: Clases de configuración

/src/main/java/.../domain/: Clases de entidades

/src/main/java/.../repository/: Clases de repositorios

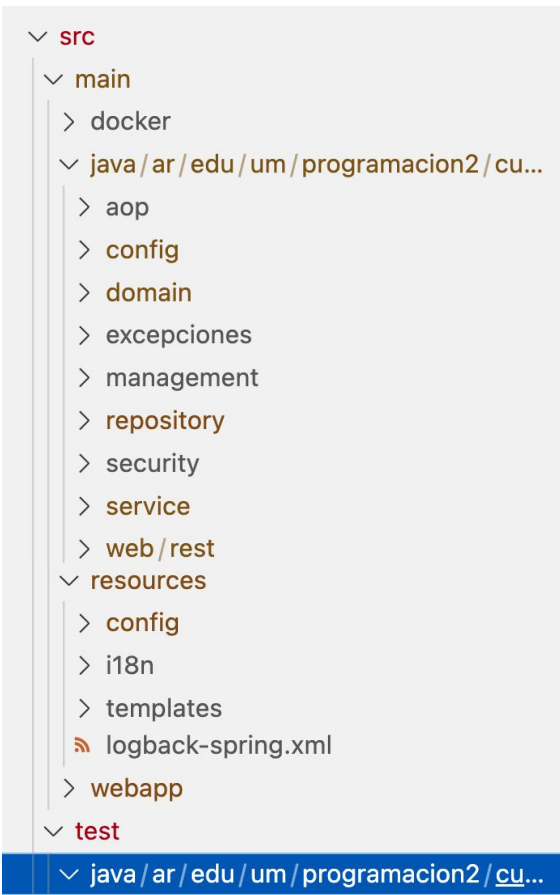
/src/main/java/.../service/: Clases servicios

/src/main/java/.../web/rest/: Clases recursos (web)

/src/main/resources/: Archivos de configuración

/test/java/.../: Clases de pruebas

A tener en cuenta a la hora del desarrollo



Estructura de directorios y archivos

./git: proyecto Git

./hipster: Directorio con los json que definen las entidades

./target/h2db: base de datos H2

./pom.xml: Archivo de configuración del proyecto

./mvnw: Archivo wrapper de mvn (maven) que permite hacer distintas tareas del proyecto (compilado, pruebas, arranque, etc)

A tener en cuenta a la hora del desarrollo

Cambios en las entidades

Ya sea con el comando para modificar las entidades o el `import-jdl`

Cambia:

- La clase de la entidad
- El servicio relacionado a la entidad
- Los repositorios
- Y varias cosas mas

Recomendación:

- Antes de hacer cambios, guardar todo en git
- Crear clase servicio helper que utilice la clase de servicio original y es donde se va a programar nuestros cambios.

A tener en cuenta a la hora del desarrollo

Logs

Si se necesita logear algo, usar el logger. Existen varios niveles de Logs.

```
@RestController
@RequestMapping("/api/authenticate")
public class AutenticateExtraResource {

    private final Logger log = LoggerFactory.getLogger(AutenticateExtraResource.class);

    public void metodo() {
        log.info("Mensaje para el canal INFO");
        log.warn("Mensaje para el canal WARN");
        log.debug("Mensaje para el canal DEBUG");
        log.trace("Mensaje para el canal TRACE");
        log.error("Mensaje para el canal ERROR");
    }
}
```

A tener en cuenta a la hora del desarrollo

Lectura de servicio externo (WebFlux)

Existen 2 formas: WebFlux o RestTemplate

```
public void lectura() {
    String url = "localhost:8080/api/clientes/" ;
    String token =
"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJmZXJuYW5kbyIsImF1dGgiOiJST0xFX1VTRVIiLCJleHAiOiE3." ;
    ClientesInDTO resultado;
    HttpHeaders headers = new HttpHeaders();
    headers.setBearerAuth(token);
    headers.setContentType(MediaType.APPLICATION_JSON);

    Mono<ClientesInDTO> clienteMono = WebClient.create()
        .get()
        .uri(url)
        .headers(h -> h.addAll(headers))
        .retrieve().bodyToMono(ClientesInDTO.class);

    resultado = clienteMono.share().block();
    String breakpoint = "";
}
```

A tener en cuenta a la hora del desarrollo

Lectura de servicio externo (WebFlux)

Agregar WebFlux al pom.xml

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-webflux</artifactId>  
</dependency>
```