

# **PROYECTO PLANTAS: DOCUMENTACIÓN**

## **Introducción**

Se trata de una aplicación web de compra online especializada en flores. Con esta web se pretende construir un portal que sea capaz de mostrar una lista de los artículos disponibles permitiendo su adquisición a través de una orden de compra (carrito).

Este proyecto se ha construido en dos pilares fundamentales: un frontal web desarrollado en Angular y una capa de negocio (microservicio) construida en Java. Como base de datos se ha optado por usar una tipo no relacional, en concreto, Mongo DB.

## **Requisitos, instalación del entorno de desarrollo**

Para poner en funcionamiento la aplicación sería necesario un navegador web y una máquina virtual Java.

En cuanto a la puesta en marcha de la aplicación podría resumirse en:

- Un IDE para la parte web (por ejemplo Visual Studio)
- Otro IDE para la parte back end ( una opción sería Intelilij)

### ➤ Front-End:

La web se ha realizado con Angular como base, acompañado de los framework Bootstrap y PrimeNG.

Versión Angular: 15.2.6 (<https://angular.io/>)

Versión Bootstrap: 5.2.3 (<https://getbootstrap.com/>)

Versión PrimeNG: 14.2.5 (<https://primeng.org>)

### ➤ Parte Back-end:

El back-end se ha realizado como microservicio en Java, acompañado principalmente de SpringBoot.

Los datos utilizados para este proyecto son almacenados en una base de datos no relacional con MongoDB.

Versión Java: 17

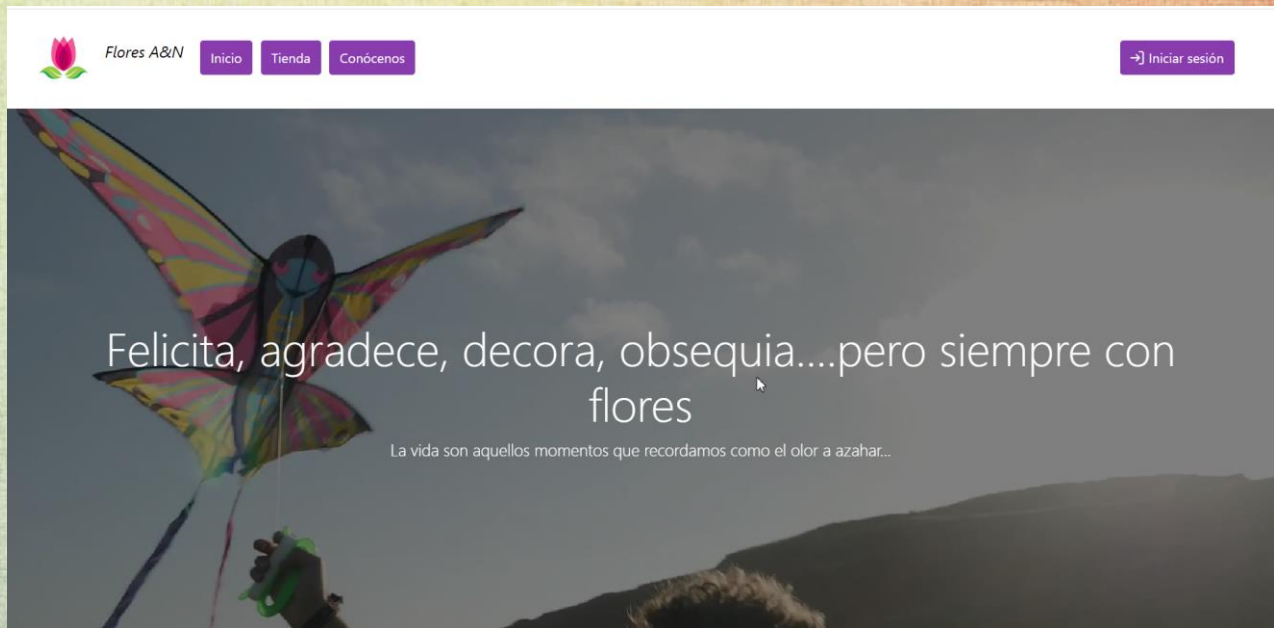
Versión SpringBoot: 2.7.3



## Funcionamiento WEB

Para llevar a cabo el proyecto hice una recopilación previa de lo que consideraba una unidad mínima viable para el producto final ya que disponía de un tiempo limitado y tenía la necesidad de formarme en tecnologías que no he usado.

Partiendo de esta base, la web cuenta con una página principal muy básica que con header con un menú principal y un botón para iniciar sesión:



El botón "Inicio" dirigirá siempre a esta pantalla de inicio.

El botón "Tienda" dará acceso a una página donde se muestran los productos disponibles para comprar.

Con el botón "Conócenos" tenía la intención de mostrar alguna información relativa a la página o autor, pero en este caso está sin realizar.

El botón de la parte derecha "Iniciar sesión", como su nombre indica, sirve para iniciar sesión si ya se dispone de una cuenta o darse de alta si no es así.



## Inicio de sesión

Flores A&N Inicio Tienda Conócenos → Iniciar sesión

INICIAR SESIÓN REGÍSTRATE

Introduce tu teléfono

Introduce tu contraseña

INICIAR SESIÓN

¿Olvidaste tu contraseña?

La pantalla de inicio de sesión es un formulario muy simple donde se deberá introducir el número de teléfono y la contraseña.

La validación de teléfono-contraseña se realiza en el back-end y se muestra un mensaje de error en la pantalla:

INICIAR SESIÓN REGÍSTRATE

321654987

.....

INICIAR SESIÓN

Teléfono o contraseña incorrectos

¿Olvidaste tu contraseña?

En la misma pantalla de inicio de sesión hay dos formularios: INICIAR SESIÓN y REGÍSTRATE.

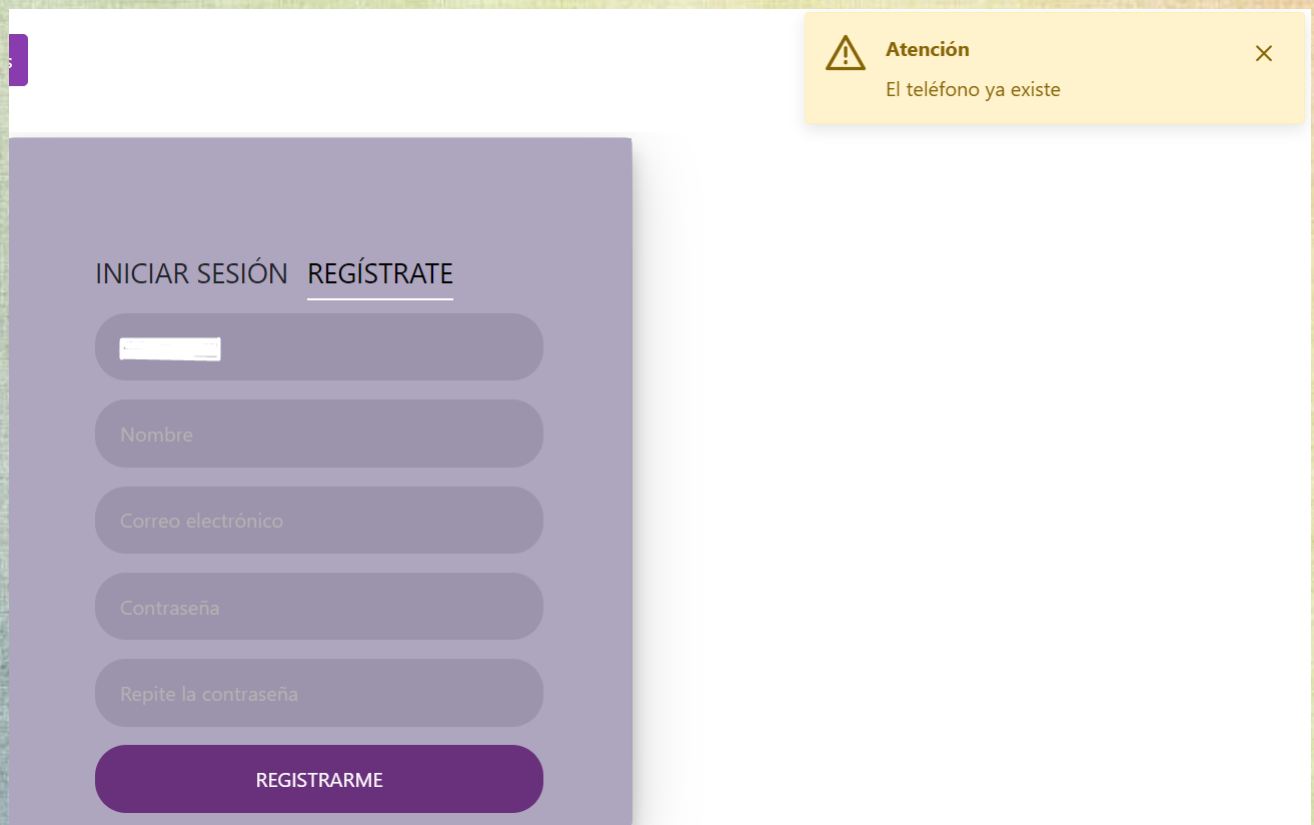


El formulario de REGÍSTRAME con tiene los campos:

- Teléfono (sirve como identificador único de un usuario)
- Nombre (Nombre con el que se dirigirá en los mensajes al usuario)
- Correo electrónico (dirección donde se enviará correos al usuario, deshabilitado)
- Contraseña
- Repite la contraseña


Este formulario también tiene algunas validaciones como:



- Número de teléfono ya existente: El back-end comprobará si existe ya dado de alta un número de teléfono, no dejando así dar de alta dos usuarios con el mismo teléfono.
- El correo electrónico deberá tener el formato típico de e-mail ([mail@mail.com](mailto:mail@mail.com))
- Los dos campos de contraseñas deben ser iguales (no se ha implementado validación de tipo de contraseña).



The image shows a web interface for user registration. At the top right, there is a yellow warning box with a triangle icon, the text "Atención", and the message "El teléfono ya existe" with a close button (X). Below this, on the left, is a registration form titled "INICIAR SESIÓN" and "REGÍSTRATE". The form has five input fields: "Teléfono" (with a small "Número" label), "Nombre", "Correo electrónico", "Contraseña", and "Repita la contraseña". At the bottom of the form is a purple button labeled "REGISTRARME".





 **Atención** 

Las contraseñas no coinciden

INICIAR SESIÓN REGÍSTRATE

666666666


Inma



mail@mail.com

Contraseña

Repite la contraseña

REGISTRARME



 **Atención** 

El mail no tiene el formato correcto

INICIAR SESIÓN REGÍSTRATE

666666666

Inma

mailmail.com

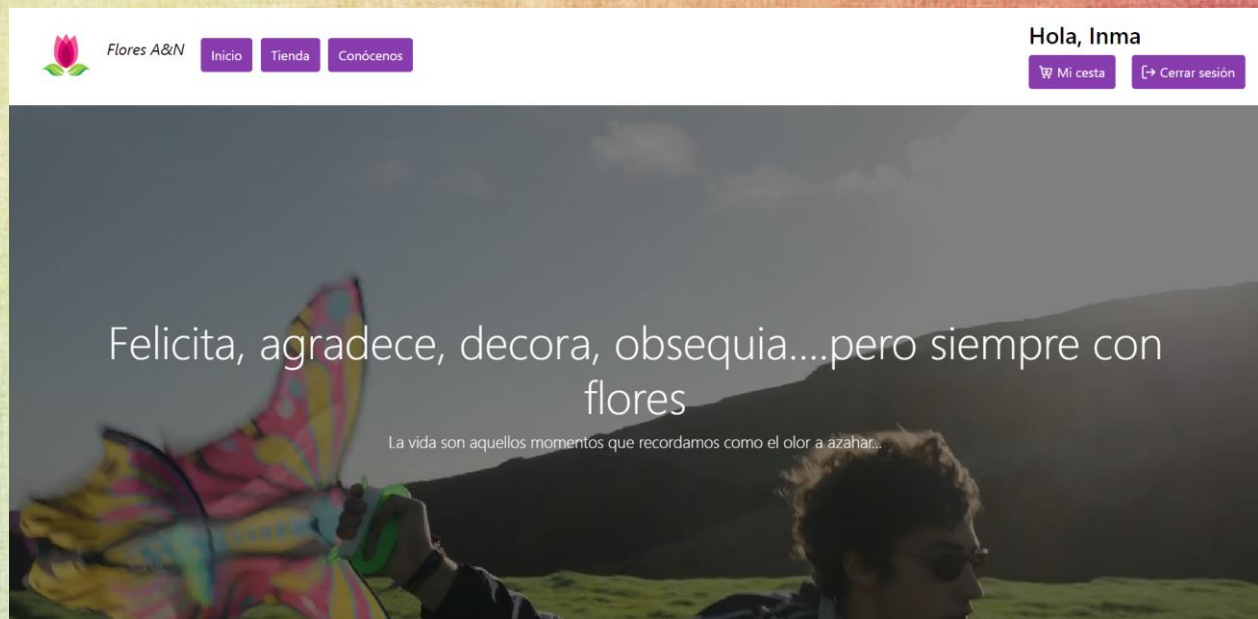
Contraseña

Repite la contraseña

REGISTRARME



Una vez dado de alta e iniciado sesión la pantalla de inicio quedará de la siguiente forma:



Aparecerá un botón nuevo “Mi cesta” que dará acceso a una pantalla donde aparecerá la lista de productos que se van a adquirir.

El botón “Iniciar sesión” se convertirá en un botón “Cerrar sesión”.

Además aparecerá un mensaje a modo de bienvenida y personalizado con el nombre del usuario.

### Acceso a “Tienda”

La página de la tienda mostrará en la parte superior 4 botones correspondiente a las tipologías de las flores a modo de filtro. En este caso, hay 3 tipologías de flores: Clásicas, Delicadas y Exóticas. El cuarto botón (Todas) sirve para listar todas las tipologías en la misma pantalla.

Por defecto, se mostrarán todas:

The screenshot shows the 'Tienda' page with a header similar to the homepage. Below the header, there are four filter buttons: 'Todas', 'Clásicas', 'Delicadas', and 'Exóticas'. The main content is a table listing various flower products.

Nombre	Foto	Precio	Categoría	Stock	Número de artículos	Añadir a la cesta
Rosa		€9.68	Clásicas	Sin stock	<input type="text"/>	<input type="button" value="+"/> <input type="button" value="🛒"/>
Hortensia		€9.48	Exóticas	En Stock	<input type="text"/>	<input type="button" value="+"/> <input type="button" value="🛒"/>
Corazón sangrante		€25.50	Exóticas	Stock bajo	<input type="text"/>	<input type="button" value="+"/> <input type="button" value="🛒"/>
Flor del cerezo		€21.41	Delicadas	Sin stock	<input type="text"/>	<input type="button" value="+"/> <input type="button" value="🛒"/>

En total hay 16 tipos de flores flores.



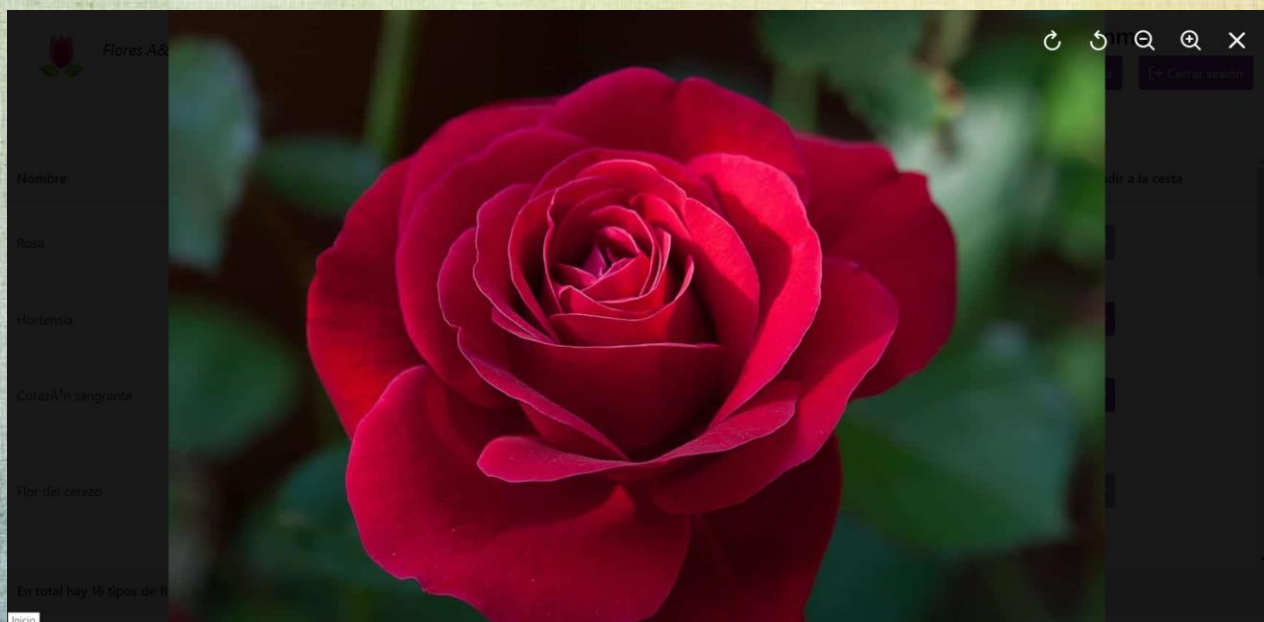
El cuerpo de esta pantalla es una tabla donde sus columnas son:

- Nombre de la flor
- Foto
- Precio por unidad
- Categoría a la que pertenece
- El stock de cada flor. (se ha optado por identificarlos con colores:
  - Rojo (sin stock): cero unidades disponibles
  - Verde (En stock): 5 o más unidades disponibles
  - Amarillo (Stock bajo): Entre 1 y 4 unidades
- Número de artículos: Componente que sirve para añadir unidades a comprar. Como mínimo será 0 y como máximo se podrá comprar el máximo de stock disponible)
- Añadir a la cesta: Se añadirá a la cesta del usuario el número de artículos seleccionados de la flor en concreto.

Tanto la columna “Número de artículos” como “Añadir a la cesta” podrán estar deshabilitados si no existen unidades disponibles.

En el pie de la tabla aparecerá el número de flores totales correspondientes al filtro seleccionado.

La foto de cada flor tiene la característica de que puede visualizarse a pantalla completa si se pulsa sobre ella:



Además se podrá alejar, acercar y rotar, así como cerrar el visualizador.



Al añadir artículos a la cesta, se mostrará un mensaje con el producto añadido:

The screenshot shows the Flores A&N website interface. At the top, there is a navigation bar with the logo and links for 'Inicio', 'Tienda', and 'Conócenos'. A green confirmation message in the top right corner states 'Añadido' (Added) and 'Se ha añadido 2 Peonias al carrito' (2 Peonies have been added to the cart). Below the navigation bar, there are category filters: 'Todas', 'Clasicas', 'Delicadas', and 'Exóticas'. The main content area displays a table of flower products with columns for 'Nombre', 'Foto', 'Precio', 'Categoría', 'Stock', 'Número de artículos', and 'Añadir a la cesta'. The table lists five items: Hortensia, Corazón sangrante, Flor del cerezo, and Peonias. Each item has a corresponding flower image, price, category, stock status (e.g., 'En Stock', 'Stock bajo', 'Sin stock'), and a quantity selector. A footer note indicates 'En total hay 16 tipos de flores flores.'

Nombre	Foto	Precio	Categoría	Stock	Número de artículos	Añadir a la cesta
Hortensia		€9.48	Exóticas	En Stock	2	
Corazón sangrante		€25.50	Exóticas	Stock bajo	4	
Flor del cerezo		€21.41	Delicadas	Sin stock		
Peonias		€13.48	Exóticas	En Stock	2	

En total hay 16 tipos de flores flores.

## Mi cesta

La pantalla de la cesta contiene un listado de los artículos que se han seleccionado para realizar la compra:

The screenshot shows the 'Mi cesta' (My cart) page on the Flores A&N website. The top navigation bar includes the logo and links for 'Inicio', 'Tienda', and 'Conócenos'. On the right, there is a greeting 'Hola, Inma' and buttons for 'Mi cesta' and 'Cerrar sesión'. The main content area features a table with columns for 'Nombre', 'Precio', 'Categoría', 'Número de artículos', and 'Eliminar producto/s'. The table lists three items: Corazón sangrante, Hortensia, and Peonias. Each item has a trash icon for removal. At the bottom of the table, the total order amount is displayed: 'El total de su pedido es: 147.92 €.' Below the table, there is a 'Finalizar compra' (Finalize purchase) button.

Nombre	Precio	Categoría	Número de artículos	Eliminar producto/s
Corazón sangrante	€25.50	Exóticas	4	
Hortensia	€9.48	Exóticas	2	
Peonias	€13.48	Exóticas	2	

El total de su pedido es: 147.92 €.

[Finalizar compra](#)

En este caso, se ha optado por visualizar el nombre de la flor, el precio por unidad, la categoría y el número de artículos por cada tipología de flor.

Además se dispondrá un botón por cada artículo para dar la posibilidad de eliminarlo de la lista.

Al final de la tabla aparecerá el precio total de la compra a efectuar así como un botón para finalizar la compra.



Una vez se realice la compra, aparecerá un mensaje para confirmar la compra y se restará el número de artículos comprados de la base de datos.

## Back-end

En este caso he decidido hacer un símil a microservicio en el sentido de que vamos a tener un servicio que sirva de soporte a la web y sea capaz de realizar todo lo que necesite ésta.

La estructura del microservicio se ha realizado en arquitectura hexagonal.

¿En qué consiste la arquitectura hexagonal?

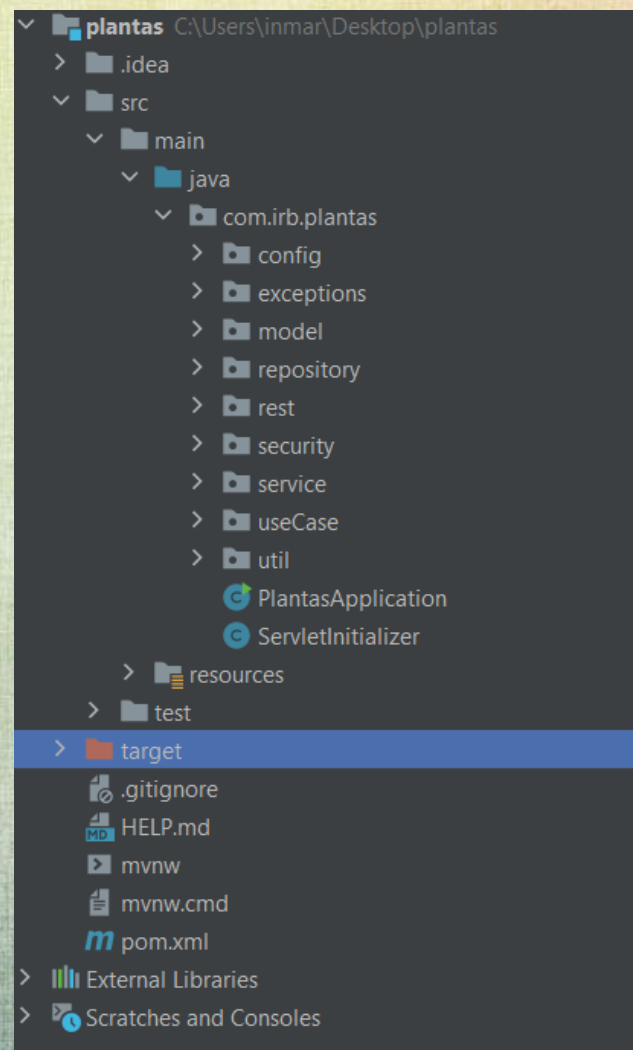
Se podría decir que es un modelo en el que entre la capa exterior (la que se expone) y la capa más profunda (la de acceso a datos) hay varias capas y entre ellas hay un total desacoplamiento, o al menos se pretende que lo haya de tal manera que si, imaginemos en un futuro se decide por optar por una base de datos relacional en vez de no relacional, solo se vería afectada la capa donde se implementa el acceso a ese tipo de base de datos.

Para empezar a desarrollar este microservicio con SpringBoot he hecho uso del inicializador de Spring (<https://start.spring.io/>), una herramienta en la cual se especifican las características principales del proyecto como: gestor de proyecto (Maven, gradle...), lenguaje (Java, Kotlin, Groovy), versión de Spring Boot, los metadatos del proyecto así como dependencias dentro de la lista de Spring.

Una vez detalladas todas estas características, puedes descargar el esqueleto ya formado y listo para empezar (este esqueleto ya es posible ejecutarlo).

En cuanto al IDE que he decidido usar en este caso es IntelliJ Idea (<https://www.jetbrains.com/es-es/idea/>) ya que considero es un IDE muy actual, avanzado y muy enfocado en el desarrollo Java.

Una vez tengo el esqueleto, lo siguiente fue pensar en las “capas” así como los paquetes que iba a contener mi proyecto:





## Explicación del código y procesos más complejos o interesantes

### ➤ Back-End:

Al tratarse de una estructura de código por capas comenzaríamos el camino por ellas:

- REST que contiene los diferentes endpoints (GET, POST, DELETE) necesarios para obtener, editar o eliminar los datos, es el canal de entrada por lo que esta sería la capa exterior y aquí tenemos una clase por cada controlador (en este caso las clases de plantas, usuarios, categoría y la orden de compra).
- USERCASE: esta sería la capa intermedia , aquí encontramos el código para las acciones que tiene cada una de las clases
- REPOSITORY: es la capa de acceso a datos, conecta con la base de datos y envía los datos solicitados a la capa UserCase anterior.

Se ha utilizado al librería de LOMBOK para el ahorro de la generación de código como son los constructores, getter/setter,...Para ello hemos he utilizado las anotaciones necesarias.

Para el apartado de configuración y seguridad me he apoyado, en prácticamente su totalidad de código, en guías y manuales para la creación de este tipo de aplicaciones.

En el TARGET está generado el .jar tras compilar el proyecto.

En el archivo POM.XML van todas las características del proyecto, versiones, dependencias...

Para completar la codificación se encuentran los modelos y el control de la excepciones.

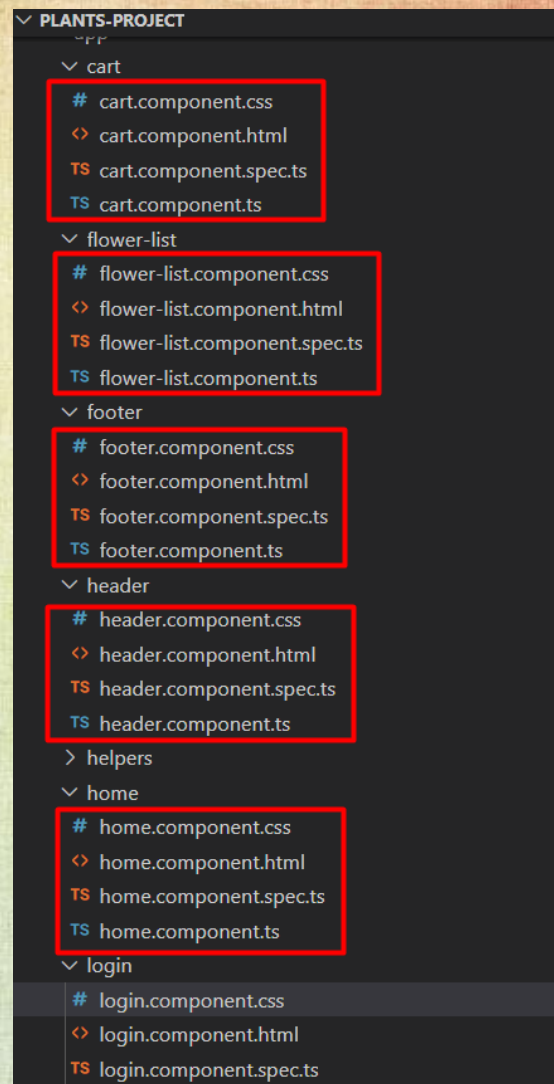
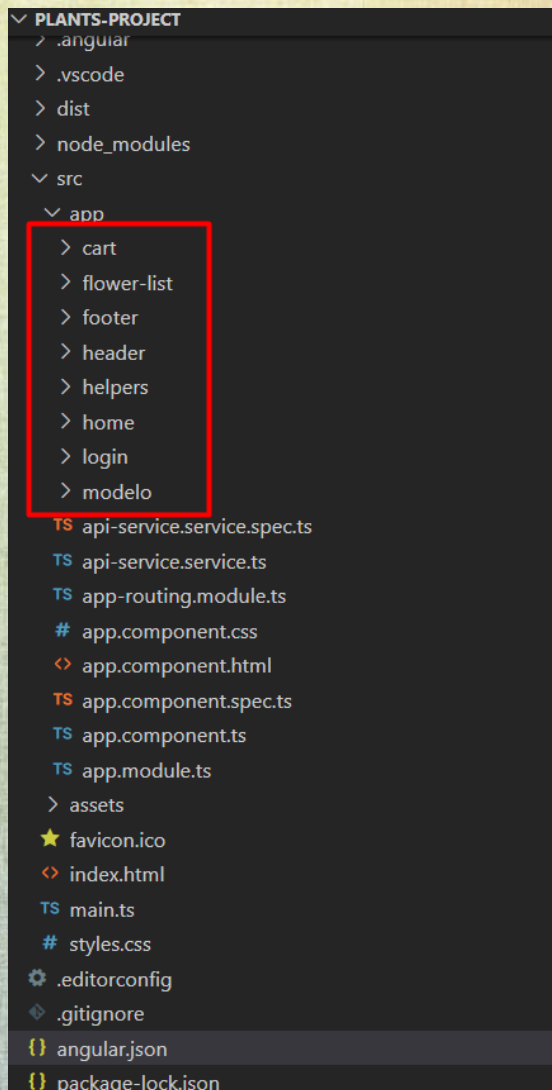
Todo ello construido con código Java.

### ➤ Front-End:

Aquí encontramos como Angular como Framework conteniendo los lenguajes html, CSS y TypeScript, utilizando también BootStrap y PrimeNG

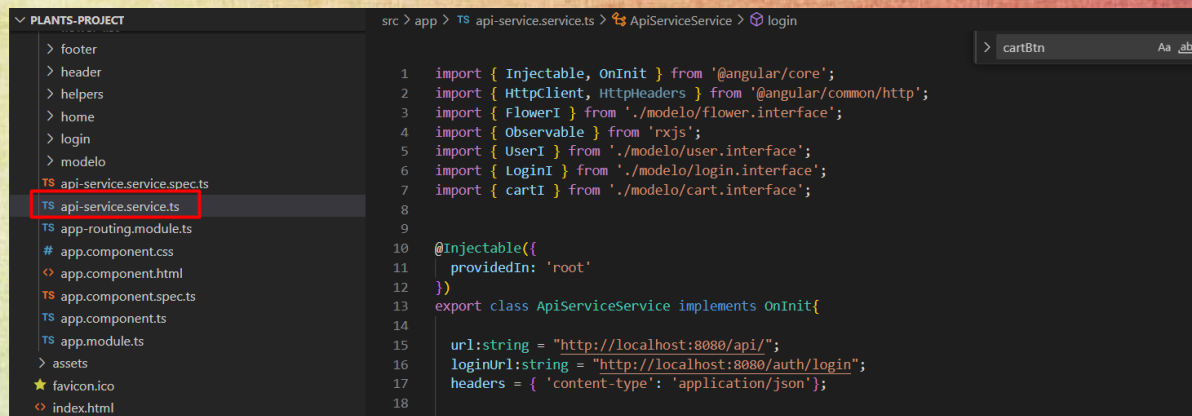


Tenemos los siguientes componentes en el proyecto, conteniendo cada uno de ellos su **CSS**, **Html**, el archivo **.ts** donde se encuentra el código necesario para todos las acciones a realizar en la vista de la aplicación y el archivo **spec.ts** que se autogenera con cada componente.



El archivo `api-service.service.ts` es la clase que conecta con el microservicio, aquí se encuentran todos los métodos necesarios para obtener los datos desde el microservicio a la web.





```
src > app > TS api-service.service.ts > ApiServiceService > login

1  import { Injectable, OnInit } from '@angular/core';
2  import { HttpClient, HttpHeaders } from '@angular/common/http';
3  import { FlowerI } from '../modelo/flower.interface';
4  import { Observable } from 'rxjs';
5  import { UserI } from '../modelo/user.interface';
6  import { LoginI } from '../modelo/login.interface';
7  import { cartI } from '../modelo/cart.interface';
8
9
10 @Injectable({
11   providedIn: 'root'
12 })
13 export class ApiServiceService implements OnInit{
14
15   url:string = "http://localhost:8080/api/";
16   loginUrl:string = "http://localhost:8080/auth/login";
17   headers = { 'content-type': 'application/json'};
18
```

## Dificultades y retos

Sin duda alguna, enfrentarme a tecnologías que no había visto con anterioridad ha sido agotador, la necesidad de recopilar información, trabajar con ejemplos, tutoriales, solicitar asesoría a profesionales cuando no encontraba información suficiente para completar el funcionamiento de algún apartado...Además he elegido una opción que parece no tener límite en las funcionalidades a crear, podría seguir durante mucho tiempo incluyendo ideas nuevas que completen y mejoren la funcionalidad y diseño de la aplicación