



CENTRO DE ESTUDIOS  
TECNOLÓGICOS Y SOCIALES  
Universidad Francisco de Vitoria · Madrid

# AGENDA

**Tomás Rodríguez-Mata Suárez**

**Proyecto Fin de Grado – Junio 2022**

**Desarrollo de Aplicaciones Multiplataforma**

**CETYS – Universidad Francisco de Vitoria**





## **Agradecimientos:**

*Agradezco de corazón, la labor de todos los docentes espectaculares que nos han enseñado durante estos dos años, en especial a Carmen por su metodología, a Inmaculada por su dedicación, a 'Nico' por su paciencia, y a Jorge, por darme ganas de aprender cada día más.*

*Agradecer también a mis compañeros de clase, en especial a Juan Pablo, Puertas, Helios, Adrián, Mulay, Alberto, Kike y Gustavo por hacer cada día más divertido.*

*Pero sobre todo a mi amigo Pablo, por guiarme y aguantarme en todo lo relacionado a la programación.*

*Gracias a todos por haberme ayudado a encontrar mi camino y otorgarme la confianza y la motivación que necesitaba desde hace tiempo. Muchas gracias.*

Tomás Rodríguez-Mata Suárez

Junio 2022

*“Sin vosotros no hubiese sido posible”*



## Índice

<b>1.</b>	<b>Acerca del proyecto .....</b>	<b>4</b>
1.1.	Origen de la idea .....	4
1.2.	Evolución de la idea .....	5
1.3.	Futuro del proyecto .....	6
<b>2.</b>	<b>Desarrollo del proyecto .....</b>	<b>7</b>
2.1.	Diseño de la base de datos .....	8
2.2.	Declaración rutas de los servicios .....	11
2.3.	Diseño de la interfaz .....	15
<b>3.</b>	<b>Tecnologías utilizadas .....</b>	<b>16</b>
3.1.	Frameworks empleados .....	16
3.2.	Lenguajes empleados .....	20
3.3.	Entorno de desarrollo (IDE) .....	23
3.4.	Otros programas .....	24
<b>4.</b>	<b>Guía de uso .....</b>	<b>27</b>
4.1.	Signup y Login .....	27
4.2.	Sección de Tareas .....	29
4.2.1.	Vistas de tareas .....	29
4.2.2.	Crear tarea .....	32
4.2.3.	Consulta de tareas .....	33
4.3.	Sección de Categorías .....	35
4.3.1.	Crear categoría .....	35
4.3.2.	Consulta de categorías .....	36
<b>5.</b>	<b>Metodología y Diseño previo .....</b>	<b>38</b>
5.1.	Agile Methodology .....	38
5.2.	Organización GitHub .....	39
<b>6.</b>	<b>Referencias .....</b>	<b>43</b>

## 1. Acerca del Proyecto

Esta aplicación web, permite crear y gestionar tareas en una agenda a la cual, se puede acceder desde cualquier dispositivo con una conexión a internet activa.

### 1.1. Origen de la idea

La idea de este proyecto, surge tres años atrás ante la búsqueda de una agenda online. En ese momento, necesitaba una agenda, con una interfaz minimalista y que fuese capaz de usarla desde todos mis dispositivos; para obtener mejora de la agenda que usaba en ese momento (la de papel tradicional). Tras una búsqueda exhaustiva, no encontré ninguna que satisficiera mis necesidades. La gran mayoría, implementaban una interfaz muy compleja y enrevesada con unos servicios de gestión bastante escasos. Sin embargo, las que cumplían con mis requisitos de interfaz y de gestión, eran aplicaciones de uso exclusivo para el móvil. Ante esta situación, pensé que sería un sueño poder diseñar mi propia agenda, que contara con todas las características que deseaba.



Hasta el día de hoy, en el que tengo los conocimientos y medios necesarios para poder desarrollar una versión que se acerque bastante al deseado proyecto de antaño.



## 1.2. Evolución de la idea

Tenía muy claro como quería que fuese la agenda: Debía ser una aplicación que permitiera a diferentes usuarios, gestionar y el almacenar la información relativa a una tarea, en una interfaz minimalista y funcional. Otro de los requisitos fundamentales era, que debía ser capaz de ejecutarse desde distintos dispositivos, con diferentes sistemas operativos con acceso a internet (especialmente en Windows, iOS y Linux) a la par de que debía usar algunas de las tecnologías aprendidas a lo largo del curso.

El proyecto del Veterinario realizado con C#, era el que más se asemejaba al mío, pero no podía usar dicho lenguaje debido a que únicamente ejecuta en máquinas Windows, dejando de lado los demás sistemas operativos. Mis principales opciones de lenguajes para desarrollar el proyecto eran Java y JavaScript, con los cuáles me sentía bastante capaz de desarrollar una aplicación de estas características.

- Tras varias semanas investigando frameworks que implementasen alguno de estos lenguajes y que pudiesen generar una aplicación compatible con distintos sistemas operativos. Decidí entonces generar una aplicación web (ejecutable en todos los sistemas operativos con acceso a internet) usando Spring para desarrollar el Backend y Angular para el Frontend. En primer lugar decidí utilizar Spring debido a que es una herramienta de código abierto para la plataforma Java, capaz de generar la base de datos con su correspondiente modelo relacional, a la par que los servicios de protocolo HTTP contra la base de datos. En segundo lugar decidí desarrollar la parte frontal de la web con Angular,



debido a que también es una herramienta de código abierto de Google desarrollada en TypeScript, muy similar a JavaScript. Una de las características que más me llamaban de este framework, es que implementa unas librerías HTML, capaces de generar nativamente componentes web muy útiles.

### **1.3. Futuro del proyecto**

En el futuro, seguiré ampliando y perfeccionando los servicios de la aplicación, junto a la colaboración de otros miembros que ayudarán a desarrollar el proyecto. Hasta que no se añadan significativas mejoras, se usará de forma local para personas allegadas al desarrollo del proyecto.

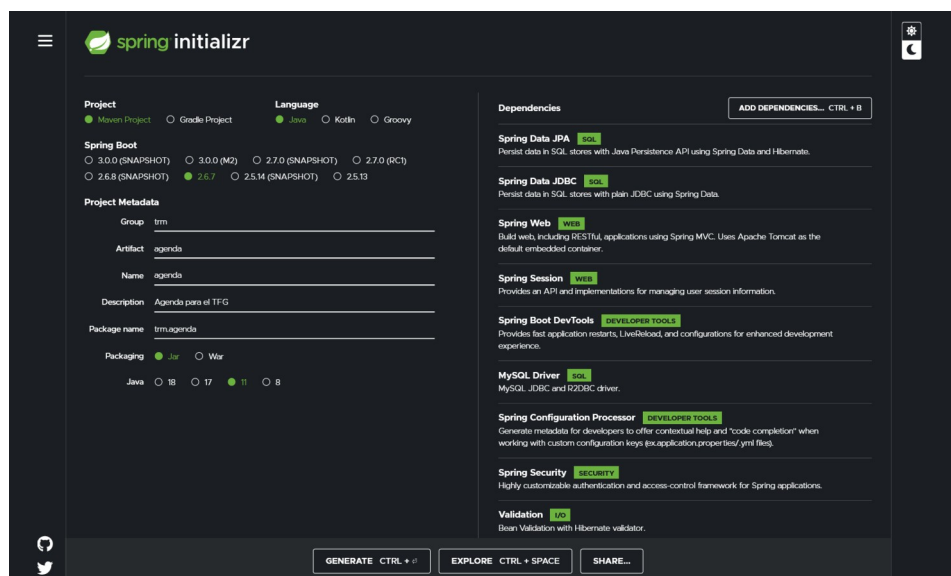
Cuando se añadan y testen las nuevas mejoras, se procederá a comprar un dominio y contratar un servidor pequeño que aloje la aplicación. Podrá ser utilizada de forma gratuita por cualquier persona con conexión a internet.

En caso de que la aplicación web tenga una significativa cantidad de usuarios activos, y un alto tráfico de información, se procederá a mejorar la capacidad del servidor y a desarrollar nuevas funcionalidades. Una de ellas podría ser una versión Premium de la aplicación que contase con funcionalidades y servicios exclusivos para el cliente. Esta, se obtendría mediante una suscripción mensual con un precio inferior a los 5\$.



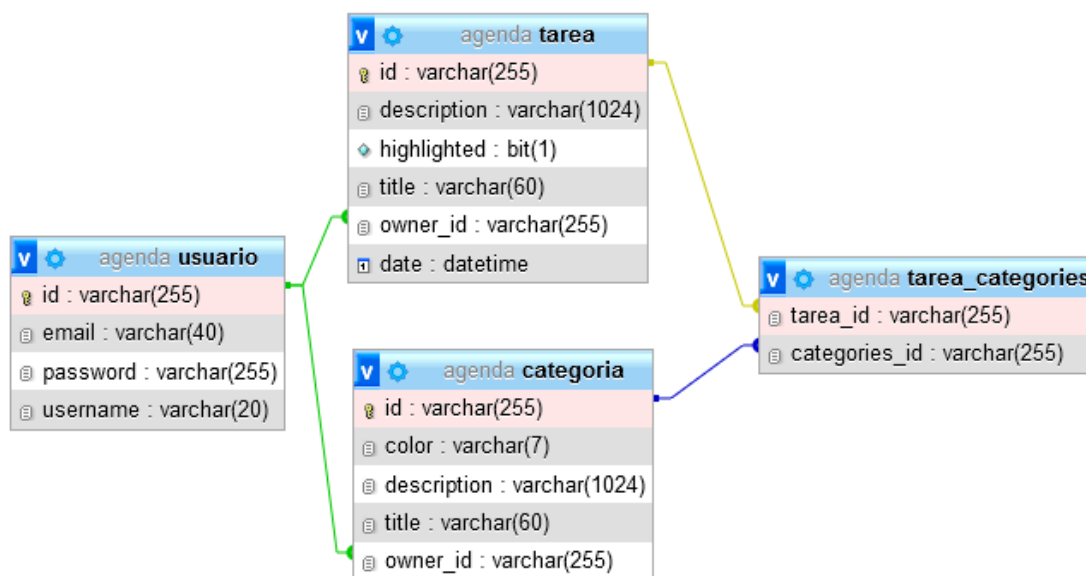
## 2. Desarrollo del proyecto

Lo primero que hice, fue crear el proyecto en GitHub y declarar las primeras tareas para la aplicación. A rasgos generales, podemos distinguir dos etapas en el desarrollo del proyecto: La primera es generar la base de datos y sus servicios mediante Spring. La base de datos MySQL se almacenará en un servidor de 'phpMyAdmin'.



## 2.1. Diseño de la Base de Datos y sus Servicios

El modelo de la base de datos relacional “agenda”, se compone de cuatro tablas. La tabla ‘usuario’ que almacena a los usuarios registrados, la tabla ‘tarea’ que almacena las tareas asociadas a un usuario, y la tabla ‘categoría’ que almacena las categorías creadas por el usuario. Estas tres tablas tienen el campo ‘id’ como Clave Primaria, que se autogenera al crear un elemento. La tabla ‘tarea\_categories’ alberga las categorías asociadas a las correspondientes tareas. Esta es necesario debido a la relación entre ‘tarea’ y ‘categoria’ (0 : N).





El modelo de datos de cada entidad, está declarado en Spring en sus correspondientes repositorios de cada entidad (usuario, tarea y categoría). A continuación, se ven las declaraciones de todos los modelos junto a estructura en las tablas:

## Entidad 'usuario'

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 	varchar(255)	utf8mb4_general_ci		No	Ninguna
2	email	varchar(40)	utf8mb4_general_ci		No	Ninguna
3	password	varchar(255)	utf8mb4_general_ci		No	Ninguna
4	username	varchar(20)	utf8mb4_general_ci		No	Ninguna



```
// Declaracion de campo id (PK)
@Id
@Type(type = "uuid-char")
@GeneratedValue
private UUID id;

// Declaracion de campo username
@NotNull
@Column(nullable = false, length = 20)
@Size(min = 8, max = 20)
@Pattern(regexp = "^([a-zA-Z0-9_-]){8,20}$")
private String username;

// Declaracion de campo password
@NotNull
@Column(nullable = false)
@JsonIgnore
private String password;

// Declaracion de campo email
@NotNull
@Column(nullable = false, length = 40)
@Size(min = 10, max = 40)
private String email;
```

## Entidad 'tarea'

```
// Declaración de campo id (PK)
@Id
@Type(type = "uuid-char")
@GeneratedValue
private UUID id;

// Declaración de campo title
@NotNull
@Size(max = 60)
private String title;

// Declaración de campo date
@FutureOrPresent
@Column(nullable = false)
@JsonFormat(shape = Shape.STRING)
@JsonIgnore(Include.NON_NULL)
private LocalDateTime date;

// Declaración de campo description
@Size(max = 1024)
@ColumnDefault("")
private String description;

// Declaración de campo Highlighted
@NotNull
@ColumnDefault("false")
private Boolean highlighted = false;

// Declaración de campo categorias
@ManyToMany()
private List<Categoria> categories;

// Declaración de campo propietario (owner) -> Usuario.id
@OneToOne(fetch = FetchType.LAZY, optional = true)
@JoinColumn(referencedColumnName = "id", updatable = false, name = "owner_id")
@Type(type = "uuid-char")
@JsonIgnoreProperties({ "hibernateLazyInitializer", "handler" })
private Usuario owner;
```

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
<input type="checkbox"/>	1 id 🔑	varchar(255)	utf8mb4_general_ci		No	Ninguna
<input type="checkbox"/>	2 description	varchar(1024)	utf8mb4_general_ci		Si	
<input type="checkbox"/>	3 highlighted	bit(1)			No	0
<input type="checkbox"/>	4 title	varchar(60)	utf8mb4_general_ci		No	Ninguna
<input type="checkbox"/>	5 owner_id 🔑	varchar(255)	utf8mb4_general_ci		Si	NULL
<input type="checkbox"/>	6 date	datetime			No	Ninguna

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	tarea_id 🔑	varchar(255)	utf8mb4_general_ci		No	Ninguna
2	categories_id 🔑	varchar(255)	utf8mb4_general_ci		No	Ninguna

## Entidad 'categoria'

```
// Declaración de campo id (PK)
@Id
@Type(type = "uuid-char")
@GeneratedValue
private UUID id;

// Declaración de campo title
@NotNull
@Size(max = 60)
private String title;

// Declaración de campo description
@Size(max = 1024)
@ColumnDefault("")
private String description;

// Declaración de campo color
@Column(nullable = false)
@Size(min = 4, max = 7)
@Pattern(regexp = "^#[a-fA-F0-9]{6}[a-fA-F0-9]{3}$")
private String color;

// Declaración de campo propietario (owner) -> Usuario.id
@OneToOne(fetch = FetchType.LAZY, optional = true)
@JoinColumn(referencedColumnName = "id", updatable = false, name = "owner_id")
@Type(type = "uuid-char")
@JsonIgnoreProperties({ "hibernateLazyInitializer", "handler" })
private Usuario owner;
```

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 🔑	varchar(255)	utf8mb4_general_ci		No	Ninguna
2	color	varchar(7)	utf8mb4_general_ci		No	Ninguna
3	description	varchar(1024)	utf8mb4_general_ci		Si	
4	title	varchar(60)	utf8mb4_general_ci		No	Ninguna
5	owner_id 🔑	varchar(255)	utf8mb4_general_ci		Si	NULL

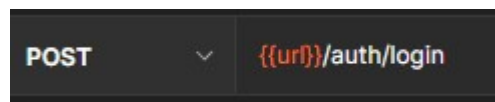


## 2.2. Declaración rutas de los servicios

- Cada entidad tiene asociado un controlador, que le permite interactuar con la aplicación para realizar servicios correspondientes a la entidad. A continuación veremos las rutas declaradas en el Controlador de Spring, junto a la ruta de cada servicio testeado desde Postman (imagen de abajo):
- **Controlador ‘usuario’:** La ruta padre de la entidad es **“/auth”** la cual tiene asociadas dos rutas:

```
@RestController
@RequestMapping("/auth")
public class AuthenticationController {
```

**“/login”** de tipo POST que se encarga de validar el inicio de sesión de un usuario



**“/signup”** de tipo PUT, se encarga de recoger los valores de un usuario, para registrarlo en la base de datos.



- **Controlador ‘tarea’:** La ruta padre la entidad es **“/tarea”** que tiene asociada 9 rutas:

```
@RestController
@RequestMapping("/tarea")
public class TareaController {
```

“/” de tipo GET, devuelve todas las tareas asociadas al usuario

GET	▼	{{url}}/tarea
-----	---	---------------

“/new” de tipo POST, que envía a la base de datos el registro de una nueva tarea con sus datos correspondientes.

POST	▼	{{url}}/tarea/new
------	---	-------------------

“/edit/{id}” de tipo PATCH, edita una tarea existente.

PATCH	▼	{{url}}/tarea/edit/:id
-------	---	------------------------

“/{id}” de tipo GET, devuelve una Tarea buscada por su ‘id’.

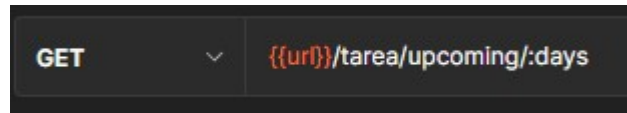
GET	▼	{{url}}/tarea/:id
-----	---	-------------------

“/highlighted” de tipo GET, devuelve las tareas destacadas

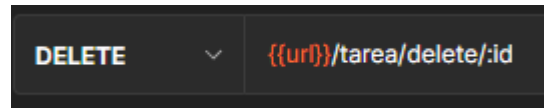
GET	▼	{{url}}/tarea/highlighted
-----	---	---------------------------



**“/upcoming/{days}”** de tipo GET, devuelve las Tareas próximas que se encuentren a tantos días de la fecha actual como se le indique en la ruta.



**“/delete/{id}”** de tipo DELETE, que elimina una Tarea, por su 'id' pasado en la ruta.



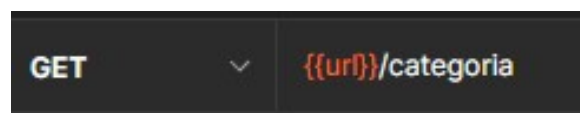
**“/search/category/{id}”** de tipo GET, que devuelve las Tareas que contengan la categoría pasada en la ruta.



- **Controlador 'categoria':** La ruta padre la entidad es **“/categoria/”** la cual tiene asociada cinco rutas

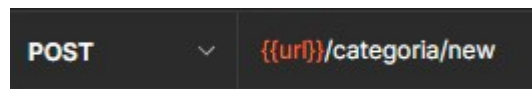
```
@RestController
@RequestMapping("/categoria")
public class CategoriaController {
```

**“/”** de tipo GET, devuelve todas las Categorías asociadas al usuario.

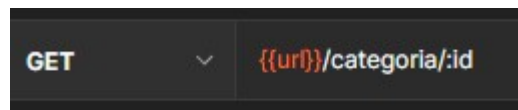




**“/new”** de tipo POST, que envía a la base de datos el registro de una nueva Categoría con sus datos correspondientes.



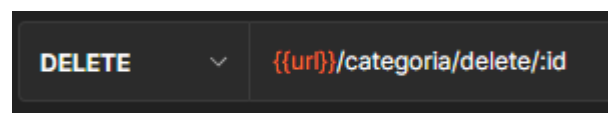
**“/{id}”** de tipo GET, devuelve una Categoría buscada por su 'id'.



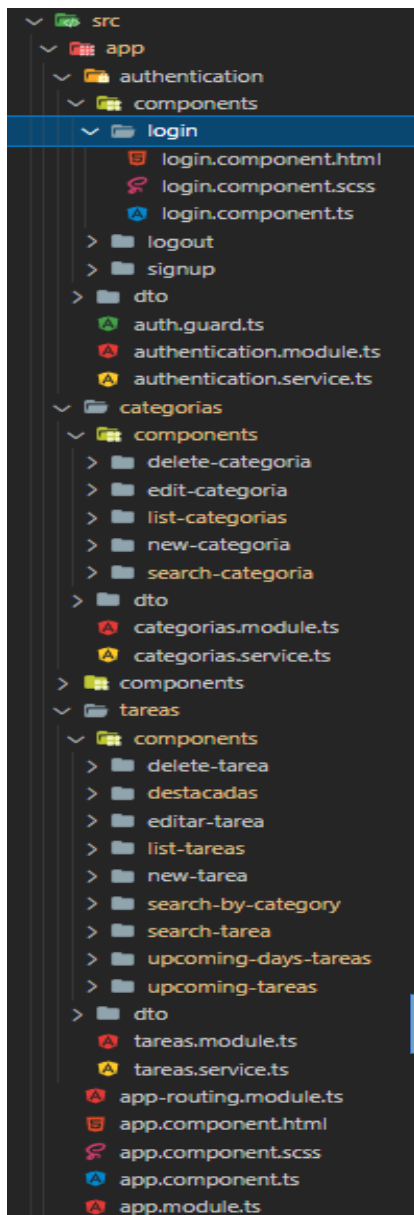
**“/edit/{id}”** de tipo PATCH, edita una Categoría existente.



**“/delete/{id}”** de tipo DELETE, que elimina una Categoría, por su 'id' pasado en la ruta.



## 2.3. Diseño de la interfaz



- Todo el diseño de la interfaz, se ha llevado a cabo con Angular, en especial, usando componentes web provenientes de la librería de 'AngularMaterial'. Cada entidad, tiene su módulo (agrupación de funcionalidad) y sus servicios (clases para comunicarse con la API). En cada entidad, están generados los componentes asociados a ella. Cada uno de ellos irá asociado a una de las rutas declaradas en el Controlador de Spring. Todos los componentes están compuestos por un HTML, un SCSS (con los estilos únicos del componente) y un TypeScript (servicios usados por el componente). Se cargarán en la ruta correspondiente de donde recogen el servicio de la base de datos.





### 3. Tecnologías utilizadas

A continuación, se van a describir todas las tecnologías usadas a lo largo del desarrollo del proyecto y como se han ido utilizando.

#### 3.1. Frameworks empleados

Como se ha decidido hacer una aplicación desacoplada, se han usado a grandes rasgos dos Frameworks. Por consiguiente uno de los Frameworks será para el Backend y el otro para el Frontend.

- Para el Backend se ha usado Spring.
- Para el Frontend se ha usado Angular.

##### **Spring**

Es un Framework para el desarrollo de aplicaciones basado en Java.

Aunque se puede usar como herramienta única para la

creación de aplicaciones, ya que permite hacer tanto la capa de base de datos como la capa de renderizado a través de JSP; se ha usado de forma puramente Backend para gestionar la conexión a base de datos y configurar los puntos de entrada a la aplicación (endpoints). Las librerías se gestionan con el gestor de paquetes "Maven".







Los componentes de Spring usados son:

- **Spring API REST:** Permite convertir el servidor web en una API, lo que genera unas configuraciones para intercambiar los datos en JSON y crear controladores REST
- **Hibernate:** Gestiona toda la capa de base de datos. Permite, a partir de clases Java estructurar y crear la base de datos haciendo las relaciones pertinentes entre las entidades, es decir, las tablas de la base de datos.
- **Spring Security:** Permite restringir el acceso a la aplicación, así como crear nuevas formas de autenticación, como el usado en esta aplicación: El JWT. Además permite restringir los orígenes de las peticiones.
- **Spring Web:** Es la capa que permite usar spring como servidor web

Adicionalmente, se han usado librerías adicionales para poder finalizar la integración

- **JWT:** Para poder generar tokens de acceso con una caducidad al usuario que se autentica. El usuario envía su usuario y contraseña y de ahí en adelante se autentica mandando el token generado en la cabecera.





## Angular

Angular es un Framework de desarrollo de aplicaciones Frontend basado en NodeJS.

Se programa en Typescript, que es un lenguaje de programación creado por Microsoft para dotar al lenguaje

JavaScript de tipado. De esta manera se puede hacer programación orientada a objetos más fácilmente. Al transpilar en JS y estar basado en él (pasando primero por NodeJS) sigue manteniendo la programación funcional que lo caracteriza.



Para gestionar las librerías se usa el gestor de paquetes "npm".

Angular es un framework muy estricto en cuanto a la estructura de código, ya que marca desde la documentación tanto la jerarquía de los ficheros dentro de la aplicación como la nomenclatura para el nombrado de los ficheros y tipos de clases que conforman la aplicación.

Se distinguen 3 niveles de profundidad básicos dentro de una aplicación:

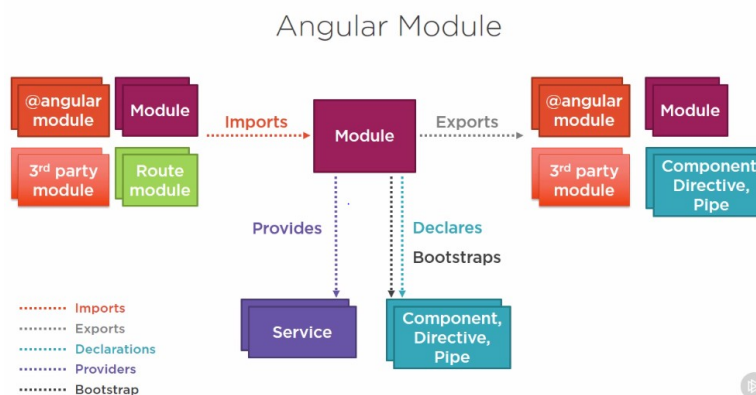
- **app:** Es el nivel superior a todos, la ruta base de los ficheros. En él está el AppModule, que es, por nomenclatura, el punto de entrada a la aplicación

- **module:** Se refiere a un conjunto de funcionalidad, como en este caso 'tareas'. Dentro de él estarán todos los componentes, servicios y otros elementos relacionados con ese conjunto de funcionalidad.

- **Elementos del módulo:**

- Servicios: Sirven para comunicarse con la API (Spring) tanto para enviar información contra la misma por ejemplo para crear o editar registros, como para consultar datos.
- Componentes: Son los elementos que construyen la interfaz gráfica. Tienen 3 partes: El HTML, el CSS y el TypeScript (que hace de JS una vez se transpila). Estos componentes pueden ser usados en los módulos donde se declaran, pero también por otros módulos y componentes, ya que por ejemplo para usar un componente de categorías en la parte de tareas habrá que exportarlo.

Para la parte gráfica se ha usado la librería de componentes Angular Material, que integra con la nomenclatura Angular un set de elementos que permiten crear las vistas más rápido, como por ejemplo los formularios de creación o edición.



## 3.2. Lenguajes empleados

Al usar dos Frameworks, se utilizan varios lenguajes de programación dependiendo de si es el Backend o el Frontend.

### Backend

En el Backend, desarrollado en Spring, se usa Java. Java es un lenguaje de programación multiplataforma que destaca por ser un lenguaje estrictamente tipado y orientado a objetos.



En esta aplicación ha usado ambas capacidades del lenguaje que están en él históricamente, pero también se han usado funcionalidades más nuevas como la programación funcional con los Streams y los datos opcionales, que son una capa de abstracción para objetos que pueden ser nulos.

Además, como se están usando versiones recientes de Spring se utilizan también las anotaciones, que sirven para dar funcionalidad adicional a clases, atributos o funciones.

Spring lee estas anotaciones que están creadas directamente en Java saber si por ejemplo una clase es un controlador o una entidad.

En el backend, además se usa XML y YAML.

XML se utiliza como índice de las librerías para que Maven las descargue.

YAML se utiliza para la configuración de Spring. En ella se configura por ejemplo el servidor y credenciales para la base de datos.



## Frontend

En el Frontend se usa HTML y TypeScript. Éste último se usa en vez de JavaScript.

TypeScript es un lenguaje que nace a partir de JavaScript para poder hacer el desarrollo más fácil al permitir tipado y poder generar clases. A diferencia de Java, el tipado no es estricto, aunque Angular, por su configuración lo exija. Esto quiere decir que a efectos prácticos, se podría programar en Typescript como si fuera Javascript directamente.

Angular, como Spring, también lee las anotaciones en las clases de TS para poder saber cuándo es, por ejemplo un componente o un servicio



- CSS: Se usa para modificar el diseño y colocación de los elementos en el HTML.

## Lenguajes comunes

JSON: Se usa para el intercambio de datos entre el Frontend y el Backend. Esto es tanto para enviar información como para recibirla. Es la única forma de comunicación entre ambas aplicaciones. Asimismo se usa como índice de los paquetes de NPM.



### 3.3. Entorno de Desarrollo (IDE)

El entorno de desarrollo empleado ha sido **VSCode**.

Este entorno, desarrollado por Microsoft es uno de los más utilizados últimamente por ser de los entornos más ligeros y rápidos.

Adicionalmente es Open Source, gratuito y tiene un amplio catálogo de extensiones para facilitar el desarrollo de cualquier aplicación.

Es un IDE relativamente reciente, y se nota tanto en su estructura como en su velocidad.

Está programado en TypeScript, con una estructura similar a la usada en esta aplicación, ya que tiene un servidor que se genera en la máquina del usuario y luego una aplicación que se conecta a él a modo de Frontend, que es la interfaz del IDE.





### 3.4. Otros programas

Se han usado varios programas para el desarrollo de esta aplicación.

- XAMPP
- PHPMyAdmin
- MySQL
- Postman
- Docker
- Docker Compose

#### **XAMPP**

Es una aplicación multiplataforma que instala en el ordenador del usuario un servidor Web con PHP, MySQL y PHPMyAdmin entre otros de Iso programa que engloba para poder tener un entorno de Base de datos y un lenguaje de Programación para poder crear



# XAMPP

aplicaciones.

#### **PHPMyAdmin**

Es un Gestor de Base de Datos. Se puede usar con Bases de Datos como MySQL o MariaDB.

Se usa para poder ver los datos creados en la base de datos, así como poder ver que las relaciones entre las





tablas se han creado correctamente.  
Además permite generar un esquema relacional de las tablas de base de datos y gestionar permisos de los usuarios. Está hecho en PHP.

## **MySQL**

Es la Base de Datos que se ha usado en la aplicación.  
Es una base de datos relacional SQL.



## **Postman**

Es un programa de escritorio para hacer pruebas sobre REST APIs. Con este programa se ha probado que todo el flujo y endpoints funcionan antes de empezar con el desarrollo en Angular.



## **Docker**

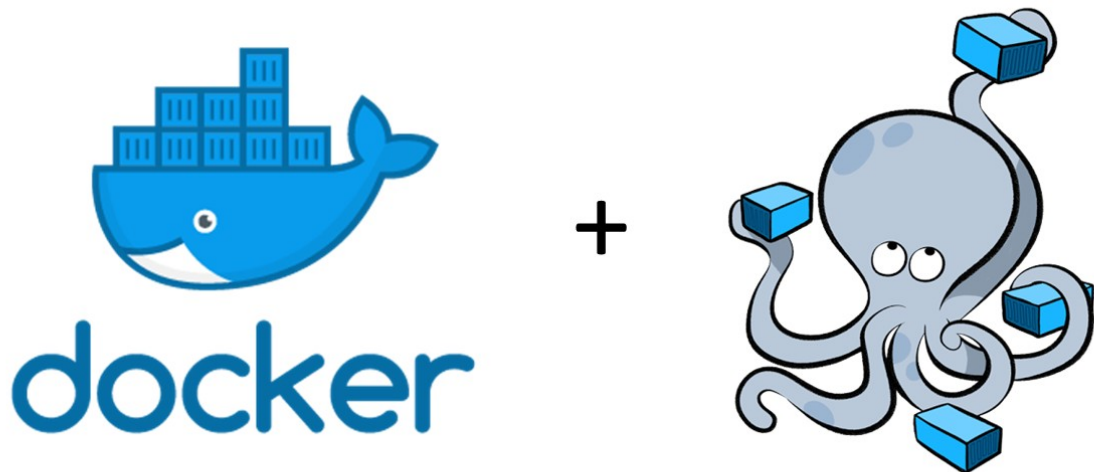
Es una herramienta que permite generar contenedores (como mini máquinas virtuales) para poder desarrollar y iniciar las aplicaciones.

Con ello se han construido a nivel local los servidores para poder usar tanto Angular como Spring.

## **Docker Compose**

Es una herramienta para hacer más fácil la gestión de los contenedores Docker.

Utiliza YAML como lenguaje de configuración.





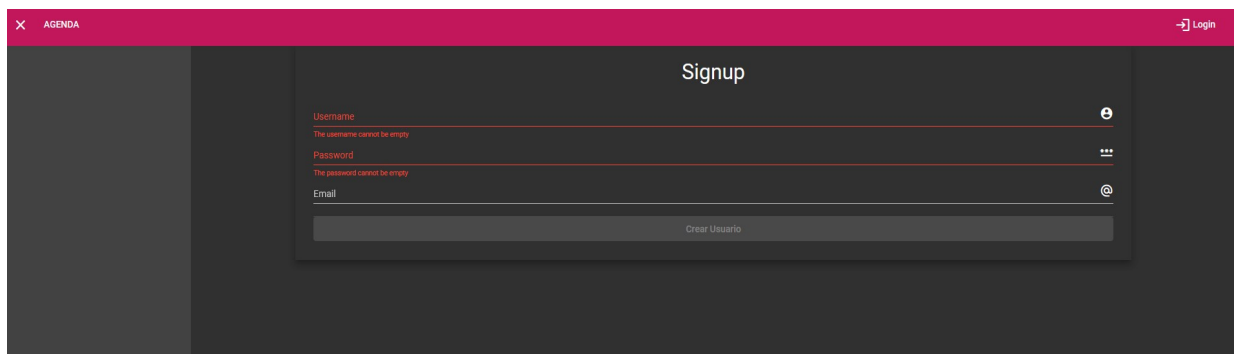
## 4. Guía de uso

A continuación se detallan una serie de capturas a modo de pequeñas instrucciones. Irán en orden al flujo inicial que tendría un nuevo usuario.

### 4.1. Signup y Login

En ninguna de las dos siguientes ventanas, se va a cargar el menú hamburguesa de la izquierda, debido a que el usuario debe identificarse previamente, antes de que le aparezcan los enlaces de dicho menú

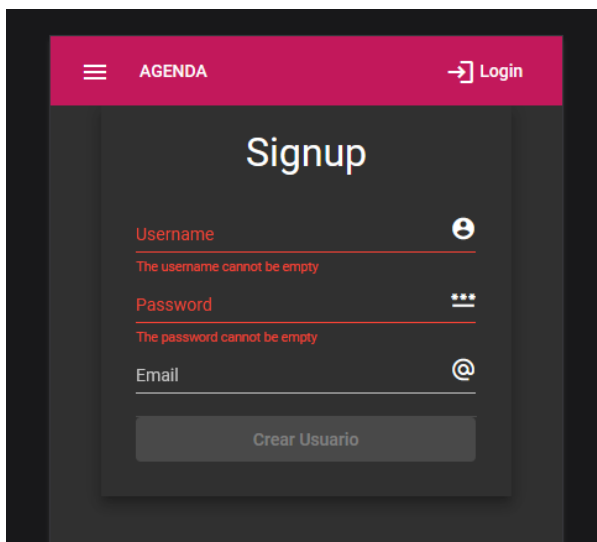
- **Signup**

A screenshot of a web application's signup form. The form is titled "Signup" and is set against a dark background. It contains three input fields: "Username" with a red error message "The username cannot be empty", "Password" with a red error message "The password cannot be empty", and "Email". Each field has a corresponding icon (person, lock, and email) on the right. Below the fields is a "Crear Usuario" button. The top of the page has a pink header with "AGENDA" on the left and a "Login" link on the right.

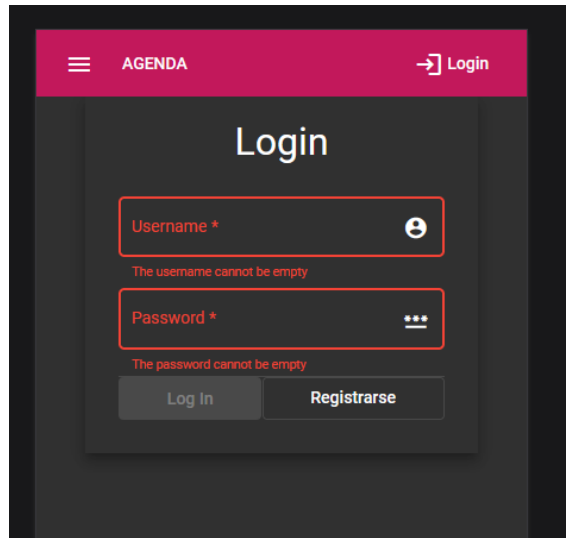
Como vemos en la imagen superior, para crear un nuevo usuario tenemos que rellenar tres campos: Nombre del usuario, contraseña y correo electrónico

- Nombre del usuario: Es obligatorio y tiene que tener una longitud de caracteres mayor a 7 y menos o igual que 20
- Contraseña: Es obligatoria y tiene que tener una longitud mínima de 8 caracteres.

- Correo electrónico: Es obligatorio y tiene que tener formate de correo estandar, el cual filtramos mediante un expresión regular

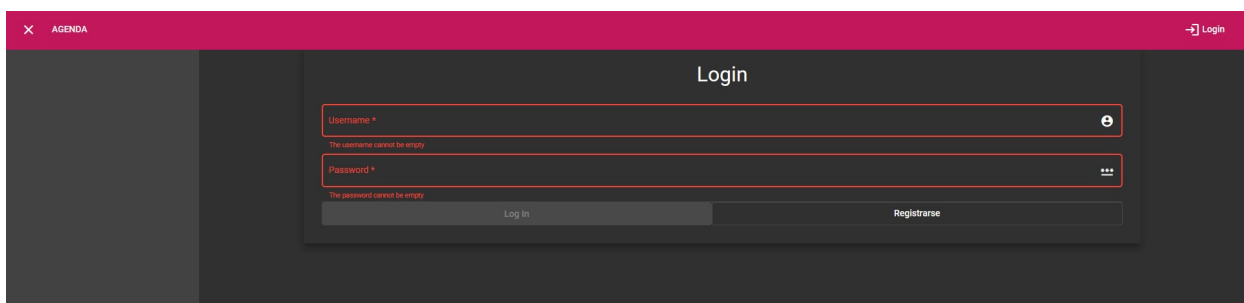


The Signup form is displayed on a dark background with a pink header. The header contains a menu icon, the word 'AGENDA', and a 'Login' button with a right-pointing arrow. The form itself is a light gray box with the title 'Signup'. It contains three input fields: 'Username' with a person icon, 'Password' with three dots, and 'Email' with an '@' icon. Each field has a red error message below it: 'The username cannot be empty', 'The password cannot be empty', and 'The password cannot be empty'. At the bottom of the form is a button labeled 'Crear Usuario'.



The Login form is displayed on a dark background with a pink header. The header contains a menu icon, the word 'AGENDA', and a 'Login' button with a right-pointing arrow. The form itself is a light gray box with the title 'Login'. It contains two input fields: 'Username \*' with a person icon and 'Password \*' with three dots. Each field has a red error message below it: 'The username cannot be empty' and 'The password cannot be empty'. At the bottom of the form are two buttons: 'Log In' and 'Registrarse'.

- **Login**



The Login form is displayed on a dark background with a pink header. The header contains a close button (X), the word 'AGENDA', and a 'Login' button with a right-pointing arrow. The form itself is a light gray box with the title 'Login'. It contains two input fields: 'Username \*' with a person icon and 'Password \*' with three dots. Each field has a red error message below it: 'The username cannot be empty' and 'The password cannot be empty'. At the bottom of the form are two buttons: 'Log In' and 'Registrarse'.

Para identificarse correctamente y acceder a la aplicación hay que rellenar 2 campos obligatorios: Nombre del usuario y su contraseña. También podemos acceder al Registro de usuario mediante el botón situado en la parte inferior a la dercha.



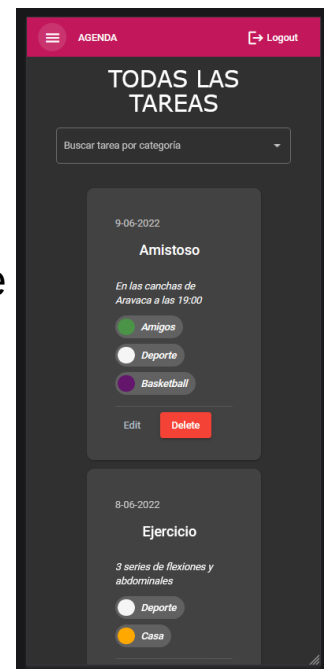
## 4.2. Sección de tareas

En esta sección veremos todo lo relacionado a como visualizar, buscar, crear y gestionar las tareas dentro de la aplicación.

### 4.2.1. Vistas de tareas

- **Todas las tareas (Home)**

Se puede acceder también desde el menú hamburguesa en el apartado “Tareas”. En esta vista podemos visualizar todas las tareas del usuario. Podemos editar y eliminar la tarea pinchando en los botones inferiores de cada tarea. Se puede consultar la información individual de las categorías asociadas a la tarea, haciendo click sobre el nombre de la categoría. También podemos acceder a la información individual de cada tarea haciendo click sobre su título.





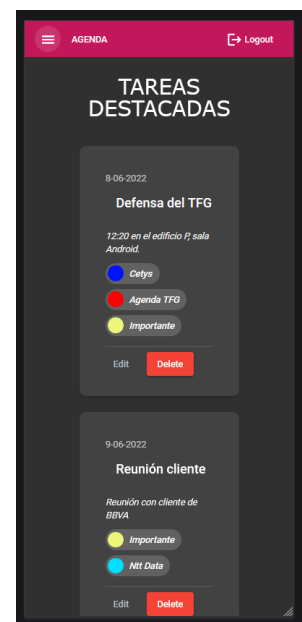
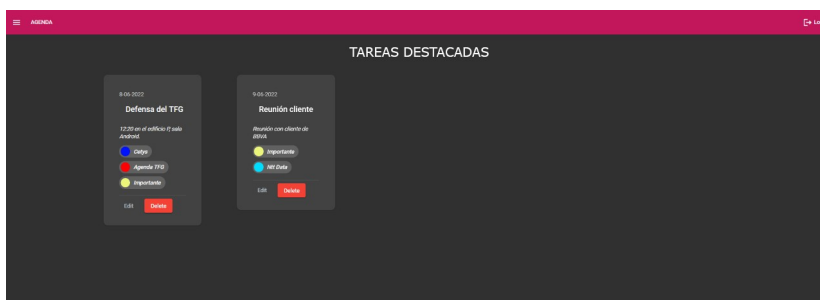
- **Eliminar Tarea:**

Se accede desde los botones individuales de cada Tarea. Este componente es un Diálogo, en el que si aceptamos, se eliminará la Tarea seleccionada y sino, se regresará al apartado “Tareas”



- **Tareas Destacadas**

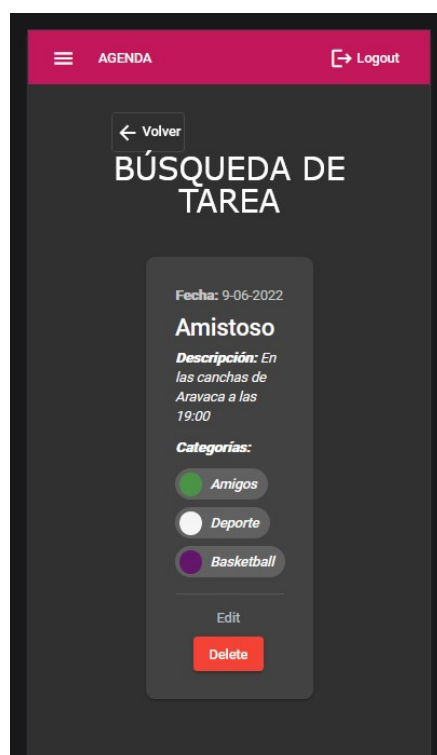
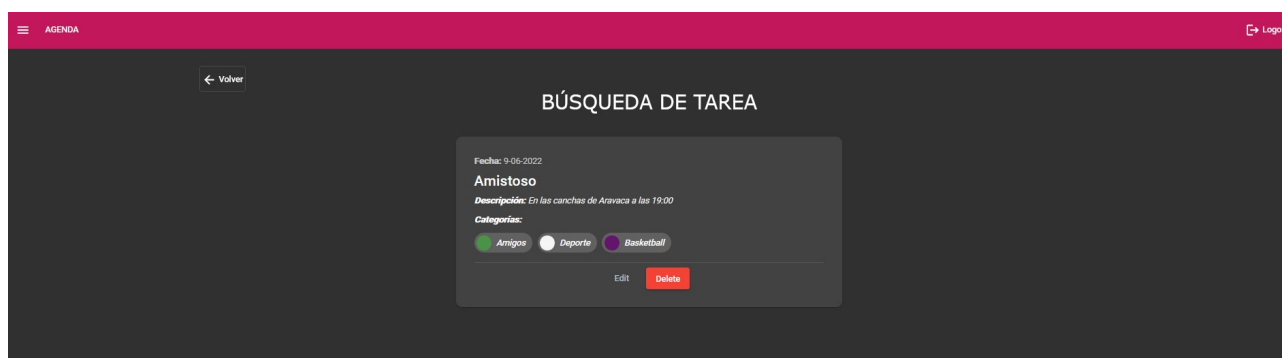
Se puede acceder también desde el menú hamburguesa en el apartado “Tareas Destacadas”. En esta se visualizan todas las tareas destacadas del usuario activo. Se encuentran las mismas funcionalidades que en el apartado “Tareas”





- **Tarea Individual**

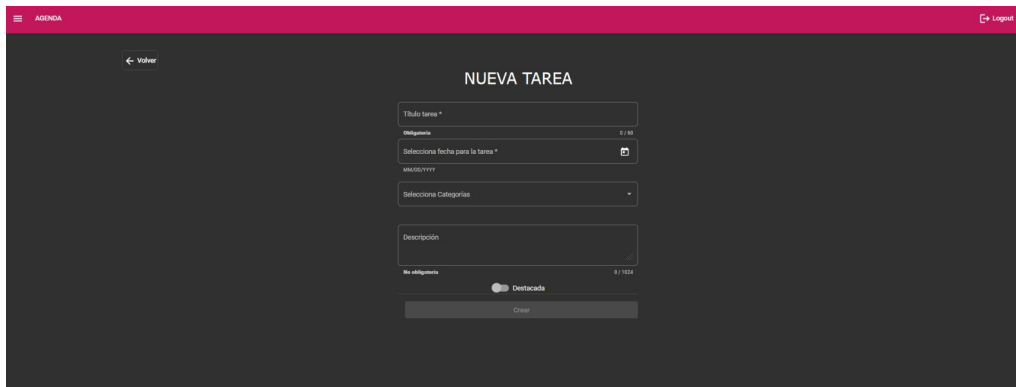
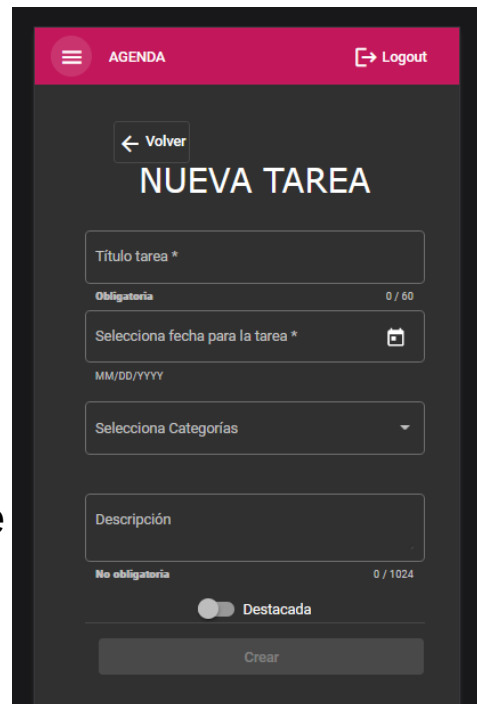
A este apartado se accede haciendo click sobre el título de la tarea correspondiente. Se visualizan todos los campos de la Tarea, junto al nombre del campo.



## 4.2.2. Crear tarea

- **Nueva Tarea**

Se puede acceder también desde el menú hamburguesa en el apartado “Nueva Tarea”. En este apartado es donde se crea una nueva Tarea, definiendola en los apartados que vemos en la imagen. Como hemos visto anteriormente en el modelo de datos, es necesario introducir valor sobre los campos de Título de Tarea y Fecha, ya que el campo Tarea destacada viene inicializado como “false”. Hasta que no se introducen los campos obligatorios, no se activa el evento sobre el botón “Crear”.



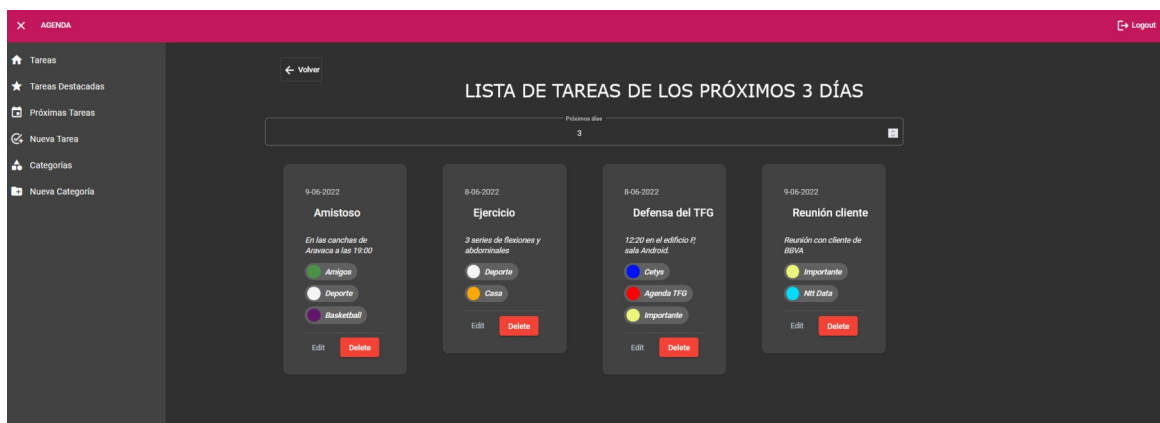
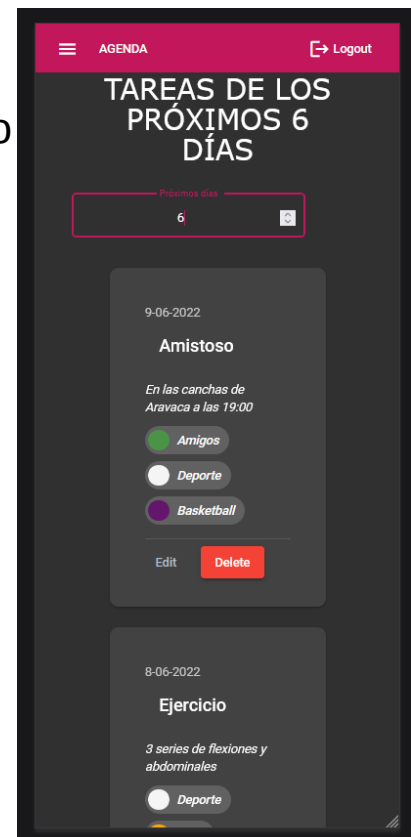




### 4.2.3. Consulta de tareas

- **Próximas Tareas**

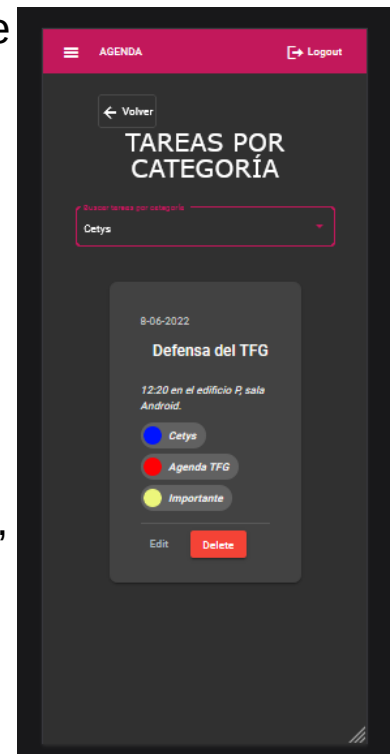
Se puede acceder también desde el menú hamburguesa en el apartado “Próximas Tareas”. En este apartado se buscan las tareas próximas a la fecha actual. En el input situado abajo del título, se introducen la cantidad de próximas días para realizar la consulta. Las tareas se recargan automáticamente al variar del número de días. El campo solamente acepta números enteros mayores que 0.





- **Buscar por Categorías**

Desde el apartado “Tareas”, se puede observar debajo del título del apartado de página, un input desplegable en el que viene escrito “Buscar tareas por categoría”. Al desplegar el campo, aparecen en forma de lista, todas las categorías del usuario. Al seleccionar una de ellas, automáticamente se hace la consulta de las Tareas. En el botón situado en la parte superior izquierda, redirige al apartado “Tareas”.





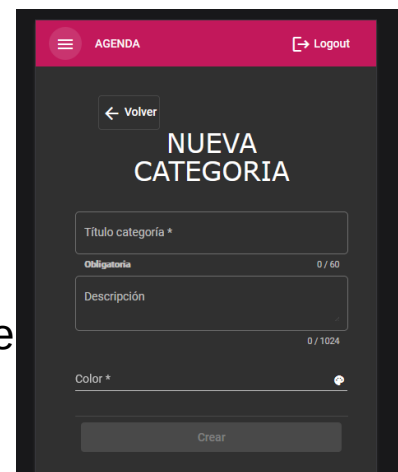
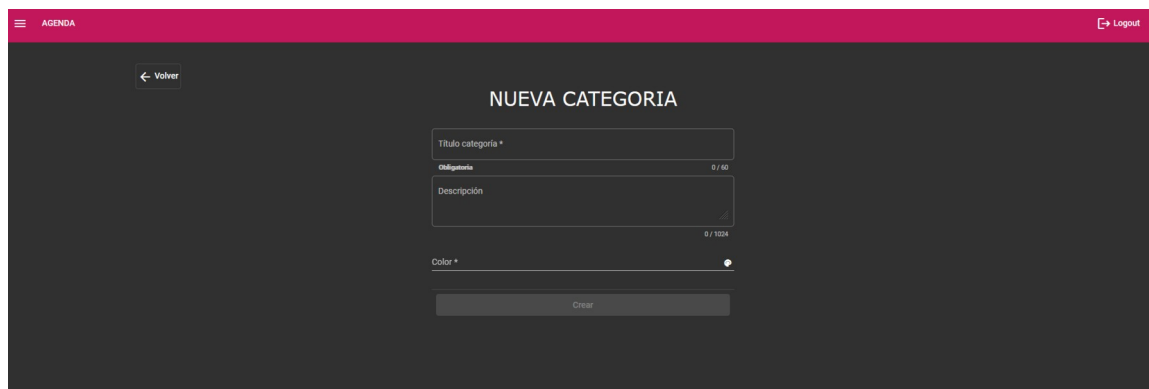
## 4.3. Sección de Categorías

En esta sección veremos todo lo relacionado a como visualizar, buscar, crear y gestionar las categorías dentro de la aplicación.

### 4.3.1. Crear categorías

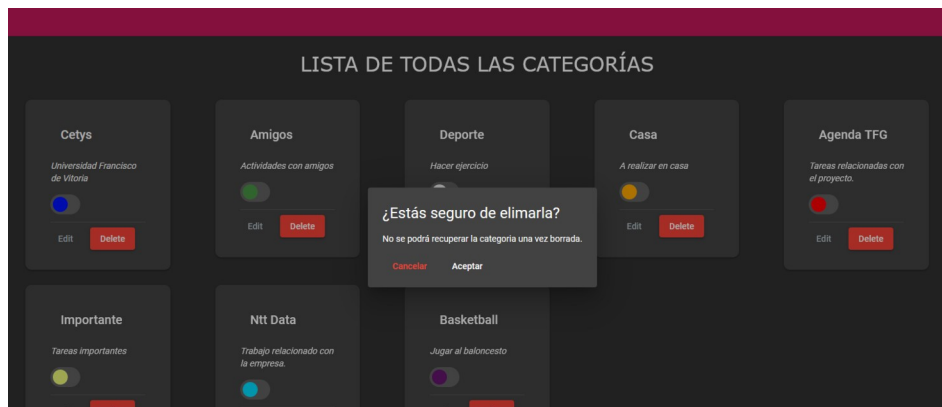
- **Nueva Categoría**

Se puede acceder también desde el menú hamburguesa en el apartado “Nueva Categoría”. Como se ve anteriormente en el modelo de datos, los campos necesarios para crear una nueva Categoría son el Título y el Color. Si alguno de los campos obligatorios no se rellena con un valor válido, no se activa la función de crear una nueva Categoría del botón “Crear”.

A screenshot of a mobile application interface for creating a new category. The screen has a dark background with a pink header bar. The header bar contains a hamburger menu icon, the word 'AGENDA', and a 'Logout' button. Below the header, there is a 'Volver' button with a left arrow. The main title is 'NUEVA CATEGORIA'. There are three input fields: 'Título categoría \*' (with a character count of 0 / 60), 'Descripción' (with a character count of 0 / 1024), and 'Color \*' (with a color picker icon). A 'Crear' button is at the bottom.A screenshot of a desktop application interface for creating a new category. The screen has a dark background with a pink header bar. The header bar contains a hamburger menu icon, the word 'AGENDA', and a 'Logout' button. Below the header, there is a 'Volver' button with a left arrow. The main title is 'NUEVA CATEGORIA'. There are three input fields: 'Título categoría \*' (with a character count of 0 / 60), 'Descripción' (with a character count of 0 / 1024), and 'Color \*' (with a color picker icon). A 'Crear' button is at the bottom.

- **Eliminar Categoría:**

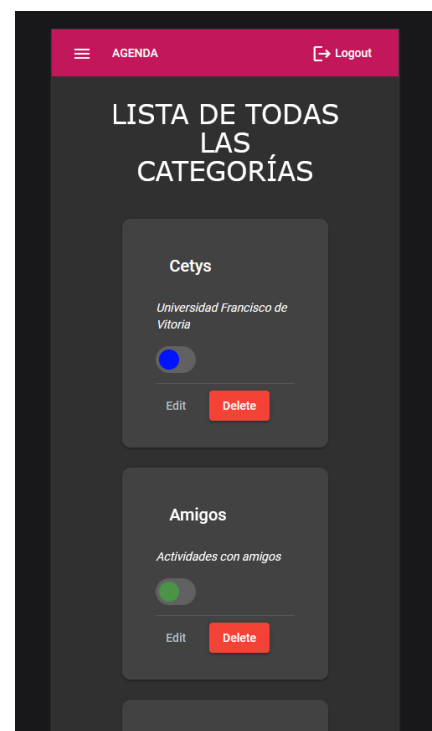
Se accede desde los botones individuales de cada Categoría. Este componente es un Diálogo, en el que si aceptamos, se eliminará la Categoría seleccionada y sino, se regresará al apartado “Categorías”



#### 4.3.2. Consulta de categorías

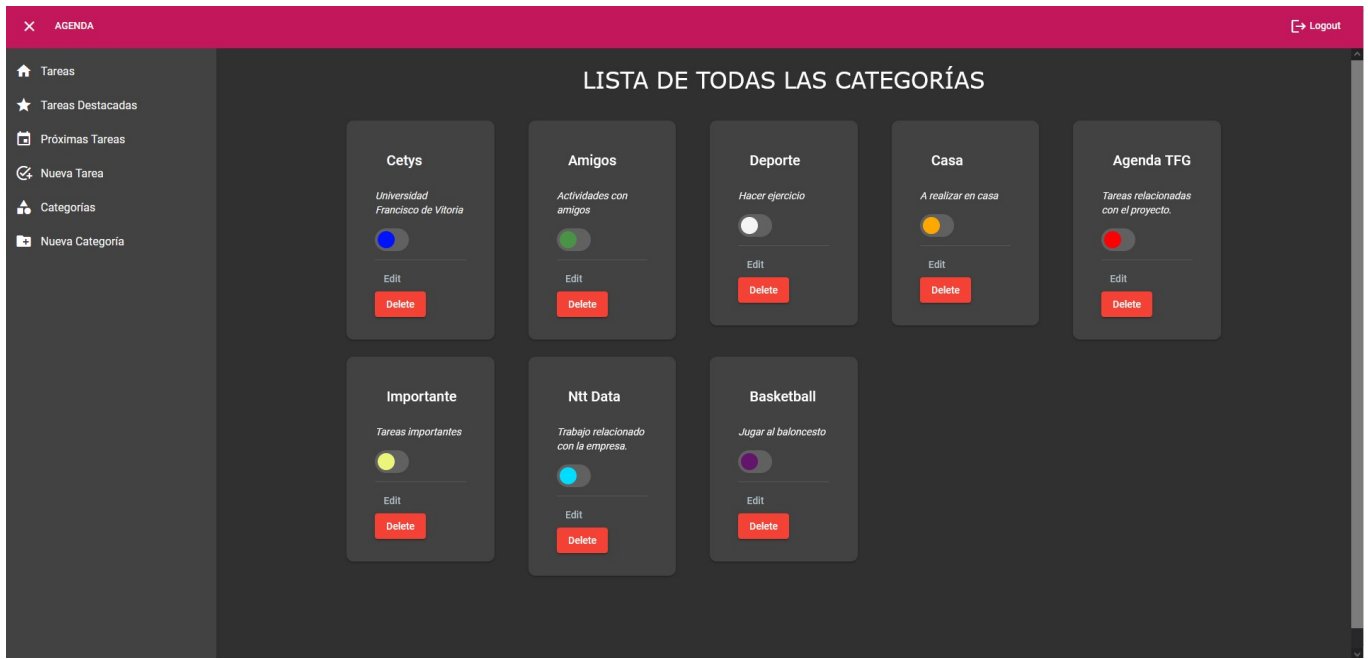
- **Todas las Categorías**

Se puede acceder también desde el menú hamburguesa en el apartado “Categorías”. En este apartado se visualizan todas las Categorías creadas por el usuario. Cada categoría se puede editar y eliminar desde los botones inferiores de cada Categoría. Únicamente se pueden eliminar aquellas Categorías que no esten relacionadas con ninguna Tarea. Se puede acceder a ver la



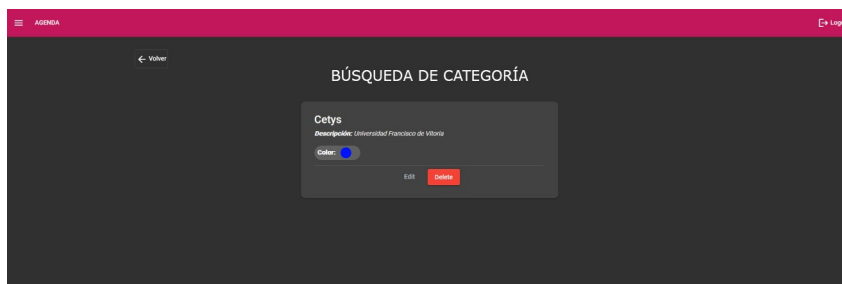
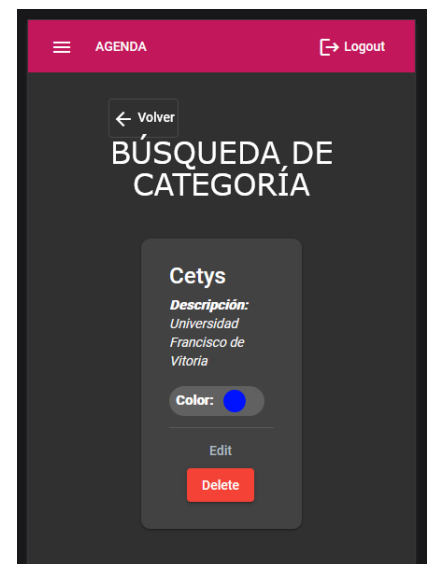


información individual de cada Categoría, haciendo click sobre el Título.



- **Categoría Individual**

A este apartado se accede haciendo click sobre el título de la Categoría correspondiente. Se visualizan todos los campos de la Categoría, junto al nombre del campo.





## 5. Metodología y Diseño previo

Para el desarrollo de todo lo relacionado con el proyecto, se ha llevado una estricta planificación, y documentación de todos los pasos realizados. Toda esta documentación se ha llevado a acabo desde la plataforma GitHub, donde también se puede descargar y ver el código fuente de la API desde su repositorio.

La idea del diseño viene previamente desde el inicio del proyecto y también se ve influenciada y adaptada gracias a varios ejemplos de los componentes de Angular Material, los cuáles han sido de gran ayuda.

### 5.1. Agile Methodology

“ El desarrollo ágil de software envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.”

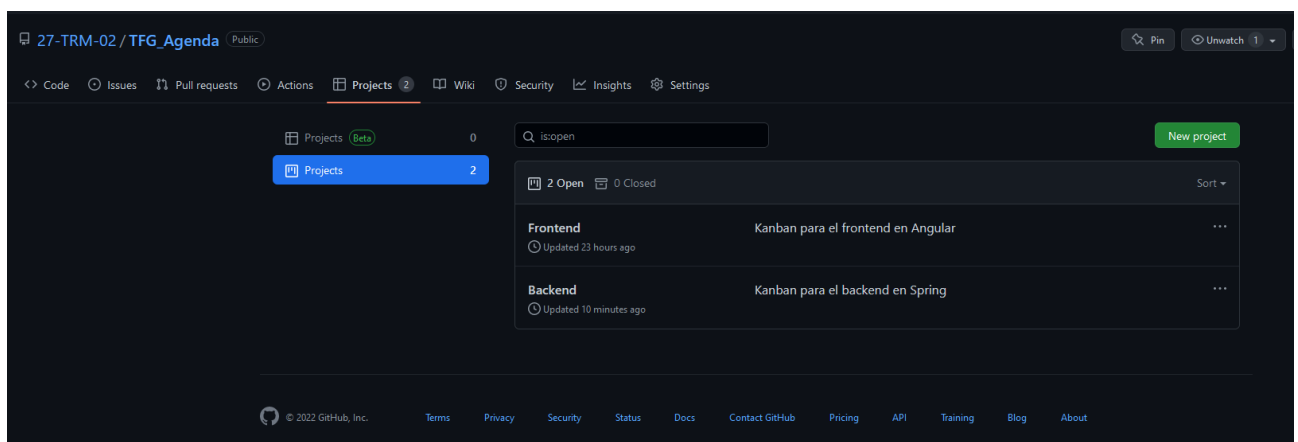
Desde el principio, con los objetivos del proyecto claros a cerca de como quería que fuese su resultado final, hicieron que este, fuese una continúa toma de decisiones al iniciarlo. Seleccionar los frameworks fue una árdua tarea, pero finalmente puedo confirmar que han sido las tecnologías idóneas para el este proyecto.



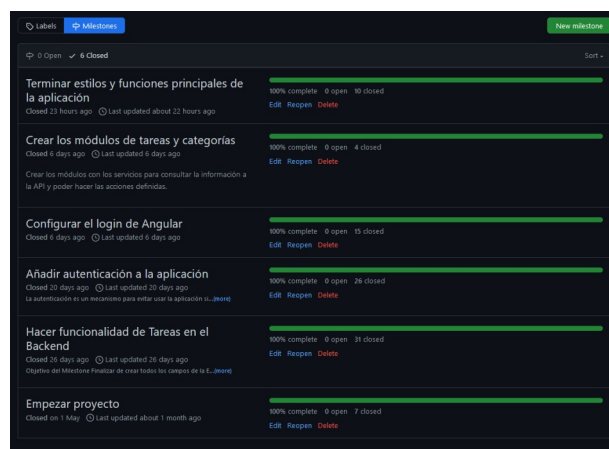
## 5.2. Organización GitHub

La organización en GitHub del proyecto se ha estructurado de la siguiente forma:

- Se han creado dos proyectos dentro del repositorio, el Backend y el Frontend.

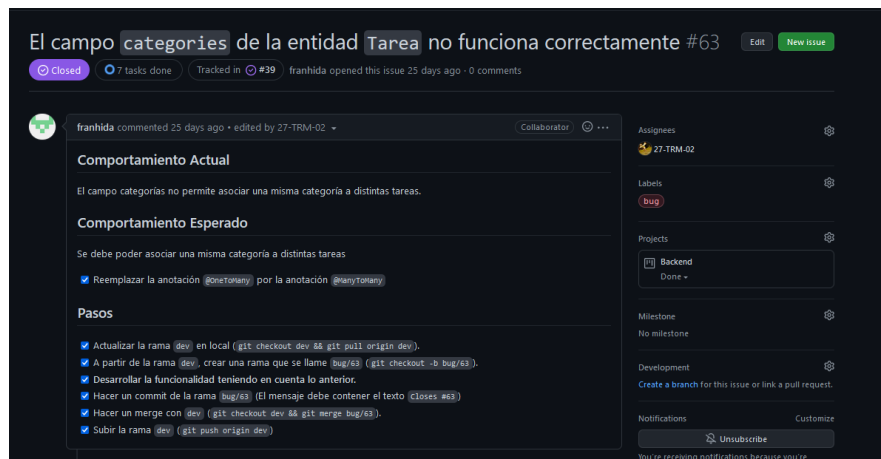


- Se han generado “Milestones” que tenían una duración de 2 semanas aproximadamente. Dentro de los “Milestones” se declaraban “epics” en los que a su vez, se declaraban en ellos “features”. Cada “Issue” lleva su correspondiente categoría de tipo de tarea.





- A lo largo de cada “Milestone”, si encontraba algún defecto, abría una incidencia de tipo “bug” en su correspondiente contexto.



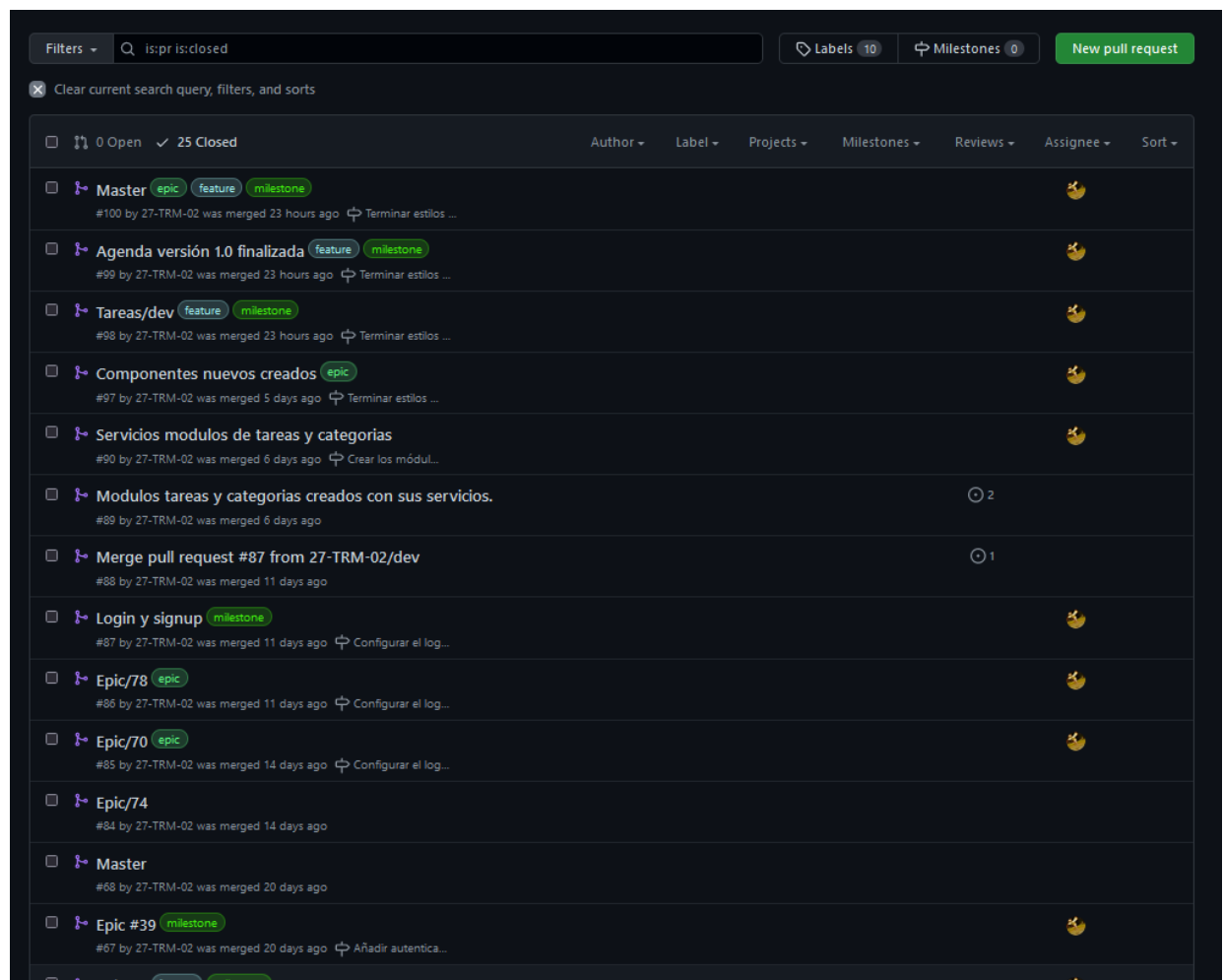
- Al terminar cada “feature” realizaba un ‘git pull’ en la rama de su “epic” donde se albergaba.
- Al terminar cada “epic” se realizaba un ‘pull request’ contra la rama ‘dev’
- Al terminar cada ‘milestone’ se realizaba un ‘pull request’ contra la rama ‘master’.







- Se ha finalizado la versión 1.0. del proyecto, con un total de 75 “Issues” cerradas, y un total de 6 “Milestones” realizados.
- Se han realizado 25 ‘pull requests’





- Aspecto final del repositorio de la API Agenda en su versión 1.0. En la parte inferior derecha se encuentran el porcentaje de uso de los distintos lenguajes de programación empleados en el proyecto.

The screenshot shows the GitHub interface for the repository '27-TRM-02 / TFG\_Agenda'. The repository is public and has 8 branches and 0 tags. The main content area displays a list of files and folders with their commit history:

File/Folder	Commit Message	Commit Date
.devcontainer	Configure frontend in docker	2 days ago
.vscode	Modelo Categoria creado con campo id	last month
Documentacion	backup commit	3 days ago
backend	Add logout component	2 days ago
frontend	Add logout component	2 days ago
.gitignore	Configure docker in environment for backend	last month
docker-compose.yml	Configure frontend in docker	2 days ago

Below the file list, there is a prompt to 'Add a README' to help people understand the project.

On the right side, the 'About' section shows 1 star, 1 watching, and 0 forks. The 'Releases' section indicates no releases published. The 'Packages' section indicates no packages published. The 'Languages' section shows a bar chart of the code language usage:

Language	Percentage
TypeScript	37.9%
Java	35.9%
HTML	19.4%
SCSS	4.4%
JavaScript	1.1%
Dockerfile	1.1%
Shell	0.2%



## 6. Referencias

Para haber tenido los conocimientos necesarios para este proyecto, me han sido fundamentales las herramientas y lenguajes aprendidos en clase. Pero al haber realizado el proyecto con frameworks nuevos para mí, he tenido que adquirir dos cursos individuales en la plataforma Udemy de Spring boot y de Angular.

También han sido fundamentales los consejos de mi amigo Pablo, pero sobre todo, buscar mucha información acerca de las tecnologías en internet.

### **Cursos:**

- <https://www.udemy.com/home>
- <https://www.udemy.com/course/spring-framework-desarrollo-web-spring-mvc>
- <https://www.udemy.com/course/angular-fernando-herrera/>

### **Imágenes:**

- <https://www.google.com/search?q=futuro+proyecto>
- <https://cleventy.com/tutorial-spring-boot/>
- [https://es.wikipedia.org/wiki/Angular\\_%28framework%29](https://es.wikipedia.org/wiki/Angular_%28framework%29)
- <https://www.toptal.com/java/rest-security-with-jwt-spring-security-and-java>
- <https://medium.com/@yonem9/angular-qu%C3%A9-son-los-m%C3%B3dulos-y-c%C3%B3mo-se-refactoriza-una-aplicaci%C3%B3n-9457550e8e9>
- <https://jhymer.dev/docker-compose-entorno-desarrollo/>
- <https://docs.docker.com/>
- <https://fonts.google.com/icons>



## **Información:**

- [https://es.wikipedia.org/wiki/Desarrollo\\_%C3%A1gil\\_de\\_software](https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software)
- <https://www.udemy.com/course/spring-framework-desarrollo-web-spring-mvc>
- <https://www.udemy.com/course/angular-fernando-herrera/>
- <https://es.stackoverflow.com/>
- <https://zoaibkhan.com/blog/create-a-responsive-card-grid-in-angular-using-flex-layout-part-1/>  
<https://indepth.dev/posts/1208/angular-flex-layout-flexbox-and-grid-layout-for-angular-component>
- <https://material.angular.io/components/categories>
- <https://www.baeldung.com/spring-boot-angular-web>