

# ProjectGO

Armando Sarrión González



# 1.- OBJETIVO DEL SISTEMA

A lo largo del segundo curso de cualquier ciclo formativo de grado superior se realiza un trabajo de fin de ciclo que tendrá un gran impacto en la nota del alumno. Estos alumnos dependen de las explicaciones de sus profesores para poder realizar un buen trabajo y muchas veces, es necesario consultar las memorias y aplicaciones de otros alumnos más antiguos para poder tener una referencia de como plantearlo. Mi aplicación funcionará como una plataforma de educación que contendrá todos los pasos a seguir para realizar un trabajo de fin de ciclo de sobresaliente.

## 2.- Alcance del sistema

La aplicación tendrá una parte cliente donde el usuario podrá interactuar con diferentes menús que le conducirán a toda la información que necesite. Además, existirá una parte servidor donde se situará una base de datos para almacenar usuarios, así como la documentación y el resto de información.

La aplicación contará con funciones para:

- 1.- Registrarse como usuario.
- 2.- Iniciar sesión.
- 3.- Consultar tu perfil de usuario para poder consultar y editar tus datos personales.
- 4.- Consultar información organizada en diferentes grupos para poder cubrir cualquier necesidad.
- 5.- Chat de comentarios para realizar consultas.
- 6.- Panel de administración para usuarios.

### 3.- Tipología de usuarios

- Usuario anónimo
  - Registrarse
  - Iniciar sesión
- Usuario registrado
  - Comentar en el chat
  - Personalización de perfil
  - Consulta de documentación
  - Consulta de información
- Usuario administrador
  - Administrar usuarios

### 4.- Restricciones

La principal restricción es asegurarse de que la funcionalidad de registro y la de inicio de sesión funcionan correctamente y que permiten diferenciar entre un usuario registrado y un usuario administrador.

Por otra parte, se deberá seguir un plan bien organizado para poder finalizar el proyecto en el tiempo establecido.

### 5.- Organización y funciones empresariales

Al tratarse de una plataforma educativa en la que solo el administrador puede administrar usuarios, será este el que desempeñará una función de dirección.

## 6.- Resultados

Página principal sin iniciar sesión. Se mostrarán las diferentes opciones de la web.



Página principal con usuario con sesión iniciada y foto de perfil.



Página de registro.

ProjectGO

Usuario

Contraseña

Enviar

Página de inicio de sesión.

ProjectGO

Usuario


Contraseña


Enviar

[Regístrate aquí](#)

Página de perfil.

ProjectGO

armando



Datos:

Usuario:  
armando

Correo:

Teléfono:


Modificar

Seleccionar archivo | Ningún archivo seleccionado

Guardar imagen

Página del administrador.

ProjectGO

armando

ID	USUARIOS
1	armando <a href="#">Eliminar</a>
2	administrador <a href="#">Eliminar</a>
3	ejemplo <a href="#">Eliminar</a>

Página de pasos a seguir. Contiene información sobre como desarrollar un proyecto, con los pasos iniciales para poder comenzar con su desarrollo.

## ProjectGO



### Índice:

- 1 - Frontend
  - 1.1 - HTML y CSS
  - 1.2 - JavaScript
- 2 - Backend
  - 2.1 - Introducción
  - 2.2 - Creación del proyecto con SpringBoot
  - 2.3 - Estructura básica del proyecto
  - 2.4 - Como crear la base de datos y añadir información inicial

### Frontend: Creación de las interfaces

#### HTML y CSS

Para crear la parte visual de nuestro proyecto utilizaremos HTML y CSS. HTML es un lenguaje de marcado de hipertexto que nos permite crear y estructurar secciones, párrafos, encabezados, enlaces y elementos para páginas web y aplicaciones. HTML funciona mediante etiquetas y cada una tiene una función diferente.

Por otra parte, CSS nos permite aplicar estilos a nuestros ficheros HTML. Utilizando CSS podremos modificar el tipo, el color o el tamaño de las letras, colocar los elementos de todas las formas que queramos, cambiar el color o colocar una imagen de fondo y mucho más. Tan solo deberemos unir nuestro fichero CSS al fichero HTML mediante un enlace en la cabecera de este último.

link href="estilo.css" rel="stylesheet" type="text/css"

#### JavaScript

Ahora que ya tenemos nuestro fichero HTML con sus estilos CSS, quizá queramos añadirle funcionalidades. Utilizando JavaScript podremos mover, crear o eliminar elementos con tan solo pulsar un botón, hacer que se ejecuten ciertos estilos solo cuando se cumpla alguna condición o incluso, obtener información de ficheros JSON que podremos utilizar en nuestra aplicación.

Al igual que pasaba con los ficheros CSS, los ficheros JavaScript deberán ir unidos a nuestro HTML mediante un link.

script src="js.js"

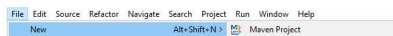
#### Backend

##### Introducción

Una vez que hemos creado nuestras interfaces y que ya tenemos claro que aspecto va a tener nuestra aplicación, lo siguiente que debemos pensar es en la estructura que deberá tener nuestra base de datos para poder almacenar toda la información que queramos tratar. Sobre esta estructura se construirá el resto del proyecto.

##### Creación proyecto con SpringBoot

Para la realización del backend utilizaremos SpringBoot, ya que nos permitirá unirlo de forma sencilla al frontend. Deberemos seguir los siguientes pasos para crear un proyecto con SpringBoot:

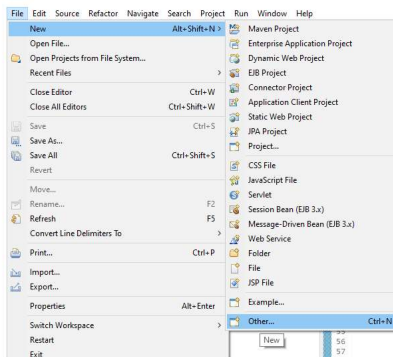


## ProjectGO



### Creación proyecto con SpringBoot

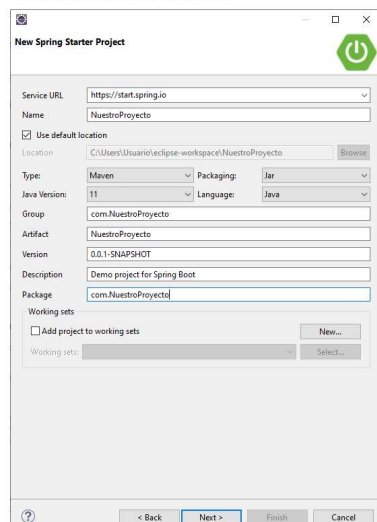
Para la realización del backend utilizaremos SpringBoot, ya que nos permitirá unirlo de forma sencilla al frontend. Deberemos seguir los siguientes pasos para crear un proyecto con SpringBoot:



Pulsamos en nuevo archivo



Escogemos el tipo de proyecto que vamos a crear



### Estructura básica del proyecto

Una vez tenemos nuestro proyecto creado, debemos tener en cuenta que todas las clases y ficheros que vayamos a utilizar deben estar organizados de una forma en concreto. Las clases Java deberán estar localizadas en sus paquetes correspondientes y los ficheros HTML, CSS y JavaScript en las carpetas que nos facilita SpringBoot. Empezaremos con las clases Java. Debemos crear una clase para cada tabla de la base de datos y para cada una de sus relaciones en caso de que sean N:M. A estas clases se les llamará modelo. Por otra parte, cada clase modelo que hayamos creado necesitará otras tres clases que nombraremos teniendo en cuenta el nombre de la misma. Tendremos pues, el modelo que se encarga de definir los campos de cada tabla de la base de datos, el repositorio del modelo que será una interfaz que extiende de CrudRepository, el servicio del modelo que será otra interfaz que contendrá las definiciones de los métodos que podremos utilizar con los objetos de los modelos y la implementación del servicio del modelo que contendrá las funcionalidades de dichos métodos.

```
ArmandoSarrión_ProjectGO [boot] [devtools]
├── src/main/java
│   ├── com.ProjectGO
│   ├── com.ProjectGO.controlador
│   ├── com.ProjectGO.modelo
│   ├── com.ProjectGO.repositorio
│   ├── com.ProjectGO.seguridad
│   ├── com.ProjectGO.servicio
│   └── com.ProjectGO.servicioImpl
└── src/main/resources
```

Paquetes donde guardaremos nuestras clases Java

En la imagen de arriba podemos ver el lugar donde guardar las clases citadas anteriormente pero, además, también podemos ver donde guardaremos los controladores, que se encargarán de mostrar los diferentes ficheros HTML cuando queramos o de pasar datos entre ellos entre otras cosas. También vemos un paquete dedicado a la clase java que va a contener todo lo relacionado con la seguridad de nuestro proyecto, ya que vamos a querer controlar que usuarios pueden acceder a la aplicación y que permisos tienen. En cuanto a la localización de los ficheros que componen nuestro frontend, deberán ser guardados en unas carpetas en concreto dentro de los recursos de nuestra aplicación. Los HTML deberán ir dentro de la carpeta "templates" y los CSS y JS dentro de la carpeta "static".

```
ArmandoSarrión_ProjectGO [boot] [devtools]
├── src/main/java
└── src/main/resources
    ├── META-INF
    ├── static
    │   ├── CSS
    │   ├── Imágenes
    │   ├── JS
    │   └── pdfs
    └── templates
        ├── admin.html
        ├── documentacion.html
        ├── ejemplos.html
        ├── guia.html
        ├── index.html
        ├── informacion.html
        ├── login.html
        ├── perfil.html
        ├── registro.html
        └── application.properties
```



informacion.html  
login.html  
perfil.html  
registro.html  
application.properties

A la hora de añadir los links a los CSS y a los JS desde HTML hay que tener en cuenta que la ruta ya comienza dentro de la carpeta "static"

### Como crear la base de datos y añadir información inicial

Con la estructura que hemos creado hasta ahora, tenemos definida la forma que queremos que tenga nuestra base de datos pero, para que esta se cree al lanzar la aplicación deberemos añadir algunas indicaciones en diferentes ficheros importantes.

```
spring.jpa.hibernate.ddl-auto=create
spring.datasource.url=jdbc:mysql://localhost:3306/proyecto?serverTimezone=Europe/Madrid
spring.datasource.username=root
spring.datasource.password=tesp0ral
|
```

El fichero application.properties deberá contener el nombre de nuestro esquema de base de datos, el usuario y la contraseña

Si además de crear la base de datos queremos que esta tenga datos iniciales deberemos crear una carpeta donde guardaremos dos ficheros, un fichero .sql que tendrá las sentencias necesarias para introducir los datos que queramos y un fichero .xml que indicará como se debe ejecutar el .sql.

```
ArmandoSanion_ProjectGO [boot] [devtools]
├── src/main/java
│   └── src/main/resources
│       └── META-INF
│           ├── data.sql
│           └── persistence.xml
├── static
├── templates
└── application.properties
```

Carpeta META-INF con los ficheros .sql y .xml

```
@SpringBootApplication
public class ArmandoSanionProjectGoApplication {

    public static void main(String[] args) {
        SpringApplication.run(ArmandoSanionProjectGoApplication.class, args);
    }

    Persistence.generateSchema("jpa", null);
}
```

Para que funcione deberemos añadir la siguiente línea en la clase programa de nuestro proyecto

Página de ejemplos prácticos. Esta página contiene imágenes que complementan a la información proporcionada en la página de Pasos a seguir.

### Índice:

- 1 - Frontend
  - 1.1 - Enlaces a estilos
  - 1.2 - Creación de un JavaScript
- 2 - Backend
  - 2.1 - Creación del proyecto con SpringBoot
  - 2.2 - Estructura básica del proyecto
  - 2.3 - Como crear la base de datos y añadir información inicial

### Frontend: Creación de las interfaces

#### Enlaces a estilos

```
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<script src="pizza.js"></script>
<link rel="stylesheet" href="/CSS/estilo1.css">
<link rel="stylesheet" href="/CSS/estilo2.css">
<title>Titulo</title>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script src="/JS/script.js"></script>
</head>
```

Así se deben enlazar los HTML con los estilos CSS y los ficheros JavaScript

#### Creación de un JavaScript

```
/* buscamos los elementos del html con los que vamos a trabajar y les asignamos un evento */
window.onload = function() {
    textArea = document.querySelector(".frase textarea");

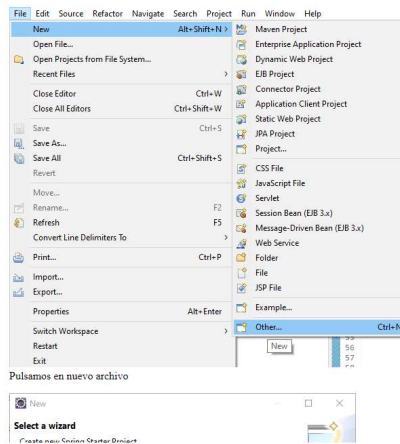
    botones = document.querySelectorAll(".botones button");

    Array.from(botones).forEach(function(boton) {
        boton.addEventListener("click", clickBoton);
    });
}
```

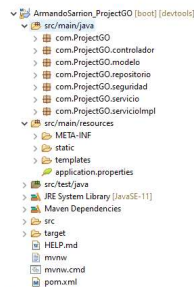
Cuando se cargue la página, se aplicarán se aplicarán las acciones indicadas en la función "onload" del JavaScript

## Backend

### Creación proyecto con SpringBoot



### Estructura básica del proyecto



Ejemplo de aplicación con todos los paquetes necesarios

### Como crear la base de datos y añadir información inicial

```
1 insert into roles(nombre) values ("ROLE_USER");
2 insert into roles(nombre) values ("ROLE_ADMIN");
3
4 insert into usuarios(usuario, password) values ("armando", "$2a$10$g8YrE3l137688IPhT.ZR0DHVWVL7mj8oo/sKkA61sPcg16hAb1");
5 insert into usuarios(usuario, password) values ("charo", "$2a$10$g8YrE3l137688IPhT.ZR0DHVWVL7mj8oo/sKkA61sPcg16hAb1");
6
7 insert into usuariosroles(idrol, idusuario) values (1, 2);
8 insert into usuariosroles(idrol, idusuario) values (2, 1);
9 insert into usuariosroles(idrol, idusuario) values (1, 1);
10
11 insert into comentarios(idusuario, contenido) values (1, "Esto es un comentario");
```

Ejemplo de fichero data.sql

```
1<?xml version='1.0' encoding='UTF-8' standalone='no' ?>
2<!-- This file is part of the Spring Data project
3     For more information see http://www.springframework.org/spring-data
```

```
7   version="2.0">
8
9
10
11
12   <persistence-unit name="jpa">
13
14     <properties>
15
16       <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
17       <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/projecto?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC" />
18       <property name="javax.persistence.jdbc.user" value="root" />
19
20       <property name="javax.persistence.jdbc.password" value="temporal" />
21
22       <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL57Dialect" />
23
24       <property name="hibernate.show_sql" value="true" />
25
26       <property name="javax.persistence.schema-generation.database.action"
27         value="drop-and-create"/>
28
29       <property name="javax.persistence.sql-load-script-source"
30         value="META-INF/data.sql" />
31
32     </properties>
33   </persistence-unit>
34 </persistence>
```

Ejemplo de fichero persistence.xml

```
@SpringBootApplication
public class ArmandoSarrionProjectGoApplication {
    public static void main(String[] args) {
        SpringApplication.run(ArmandoSarrionProjectGoApplication.class, args);
    }
}
```

Para que funcione deberemos añadir la siguiente línea en la clase programa de nuestro proyecto

Página de documentación. Contiene enlaces a PDFs importantes y una sección de chat donde dejar comentarios.

[Programación del proyecto](#)  
[Métodos de iteración en JavaScript](#)  
[Introducción a AJAX](#)  
[Chuleta HTML](#)  
[Chuleta CSS](#)  
[Introducción API Fetch](#)

armando:  
Comenta lo que quieras!

Envía tu mensaje!

Enviar

Página de información. Contiene un breve resumen de la funcionalidad de la web y un enlace al documento de la memoria.

ProjectGO



## ¿Qué es ProjectGO?

ProjectGO es una aplicación web pensada para funcionar como una plataforma educativa que te enseñará a diseñar y programar un buen proyecto de fin de ciclo.

Aquí encontrarás ejemplos reales, documentación útil y consejos, tanto de memorias como de aplicaciones.

[Memoria de este proyecto](#)

Aplicación web creada por Armando Sarrión González.

## 7.- Instalación, registro de dominio y alojamiento web

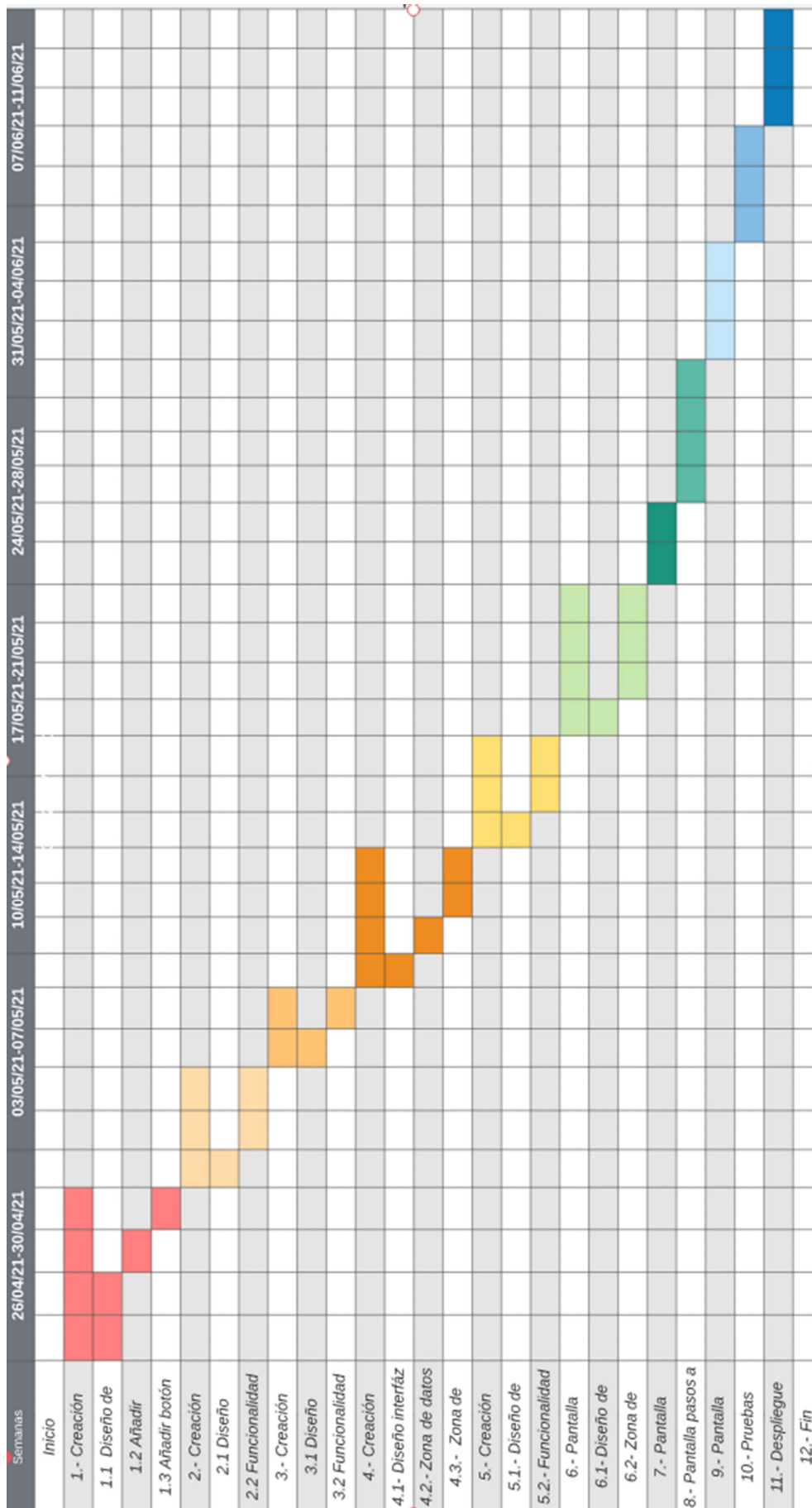
Se generará un archivo JAR y se desplegará utilizando Jelastic.

## 8.- Analítica web, posicionamiento y promoción

Debido a que esta aplicación no está pensada para su venta, no se realizará ningún tipo de operación de posicionamiento o de promoción.

## 9.- Planificación temporal y estimación del tiempo empleado en el desarrollo del proyecto.

El proyecto se desarrollará entre los meses de marzo y junio. Se realizará un análisis previo de los requisitos necesarios para su creación y se dividirán las distintas etapas del desarrollo según su cometido. De esta forma se desarrollarán el frontend, el backend y los casos de prueba en bloques independientes con temporalizaciones diferentes.



## 10.- Formación y ayuda

La aplicación está pensada de forma que sea lo más intuitiva y fácil de manejar posible. Con esto se pretende que el usuario no necesite ninguna clase de estudio previo antes de utilizar la aplicación. En caso de que algo no sea lo suficientemente claro y se necesite ayuda, el administrador del sitio estará disponible para la resolución de dudas.

## 11.- Plazo de entrega

La estimación del tiempo total de desarrollo es de 2 meses, desde la finalización de las clases en abril hasta principios de junio.

## 12.- Garantía y soporte

La aplicación no será vendida a terceros, pero el administrador ofrecerá un mantenimiento y un soporte con resolución de dudas y errores.

## 13.- Licencia y propiedad intelectual

Los derechos de la propiedad intelectual pertenecen al desarrollador del sitio web, así como todas las publicaciones realizadas en el sitio web una vez que el usuario las sube.

## 14.- Formas de pago

No se establecen métodos de pago debido a que la aplicación será completamente gratuita para todo usuario que desee utilizarla.

## 15.- Tecnologías aplicadas

Para el desarrollo del proyecto se han utilizado las siguientes tecnologías:

En el diseño de las diferentes vistas (frontend), se ha utilizado HTML, CSS y JavaScript.

Enlaces en HTML que redirigen a ficheros PDF en ventanas diferentes:

```
<a href="/pdfs/programacionProyecto.pdf" target="_blank">Programación del proyecto</a>  
<a href="/pdfs/metodosIteracionJS.pdf" target="_blank">Métodos de iteración en JavaScript</a>
```

Para desplegar el menú que contiene las opciones de iniciar sesión, ver perfil y, en el caso del administrador, gestionar los usuarios, se ha utilizado una función JQuery en un fichero JavaScript.

```
window.onload = function() {  
  
    $("#container-usuario").click(function() {  
        $(".nav-dropdown").slideToggle("slow");  
    });  
  
}
```

Para el cambio de la imagen de perfil también se ha utilizado un JQuery que cambia la ruta del recurso de la imagen.

```
window.onload = function(){  
  
    $("input[type=file]").change(function(){  
  
        if (this.files && this.files[0]) {  
  
            var reader = new FileReader();  
  
            reader.readAsDataURL(this.files[0]);  
  
            reader.onload = function (e) {  
                $("#imagen").attr("src", e.target.result);  
            }  
        }  
    });  
  
};
```

En lo referente al backend, se ha utilizado SpringBoot para la creación de las clases, Spring Security para administrar el registro y acceso a la aplicación y JPA para la inserción de datos en la base de datos.

El controlador encargado de eliminar un usuario cuando el administrador así lo desea es el siguiente:

```
@GetMapping("/eliminarusuario")
@Transactional
public String eliminarUsuario(@RequestParam int idusuario, Model modelo) {
    su.deleteByUsuario(su.findById(idusuario).get());
    su.deleteById(idusuario);
    return "redirect:/index";
}
```

Controlador encargado de cambiar la imagen de perfil del usuario:

```
@RequestMapping("/cambiarImagen")
public String actualizarImagen(@RequestParam("imagen") MultipartFile image, Authentication authentication, Model model) {

    UsuarioVO usuario = su.findByUsuario(authentication.getName());

    if (!image.isEmpty()) {
        try {
            //lo maximo que deja mysql por defecto son 4MB, el maximo que dejamos en el servidor
            //tanto como de subida de archivos como de request son 3MB por si acaso
            usuario.setImagen(Base64.getEncoder().encodeToString(image.getBytes()));

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    else {
        usuario.setImagen(null);
    }

    su.save(usuario);

    return "redirect:/perfil";
}
```



Controlador que lanza la página Home y que se encarga de identificar el rol del usuario que ha iniciado sesión:

```
@GetMapping("/index")
public String index(Authentication authentication, Model model) {

    if (authentication != null && authentication.isAuthenticated()) {

        UsuarioVO usuario = su.findByUsuario(authentication.getName());

        model.addAttribute("usuario", usuario);

        List<UsuarioRolVO> usuariosroles = sur.findByUsuario(usuario);

        Boolean isAdmin = usuariosroles.stream().anyMatch(x->x.getRol().getIdrol() == 2);

        model.addAttribute("nombreUsuario", authentication.getName());

        model.addAttribute("autenticado", 1);

        if(isAdmin) {

            model.addAttribute("admin", "true");

            return "index";
        }
        else {
            return "index";
        }
    }
    else {
        model.addAttribute("nombreUsuario", " ");
        model.addAttribute("autenticado", 0);
        return "index";
    }
}
```

Clase de Spring Security que se encarga de la autenticación y la autorización.

```
@Configuration
@EnableWebSecurity
public class MiSeguridad extends WebSecurityConfigurerAdapter{

    @Autowired
    private ServicioUsuarioImpl sui;

    @Bean
    BCryptPasswordEncoder encriptador(){
        return new BCryptPasswordEncoder();
    }

    public String encripta(String password) {
        return encriptador().encode(password);
    }

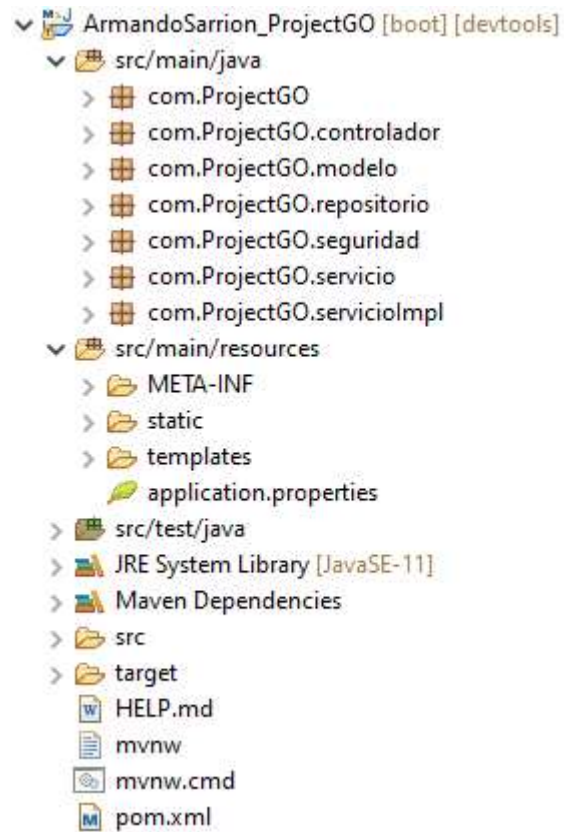
    // Autorización
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/", "/CSS/**", "/Imágenes/**", "/JS/**", "/login", "/registro", "/logout", "/index", "/submit").permitAll()
            .anyRequest().authenticated()
            .and().formLogin().loginPage("/login").defaultSuccessUrl("/")
            .and().logout().logoutSuccessUrl("/index");

        //si no pones esto al hacer submit de un form da error forbidden
        http.csrf().disable();
    }

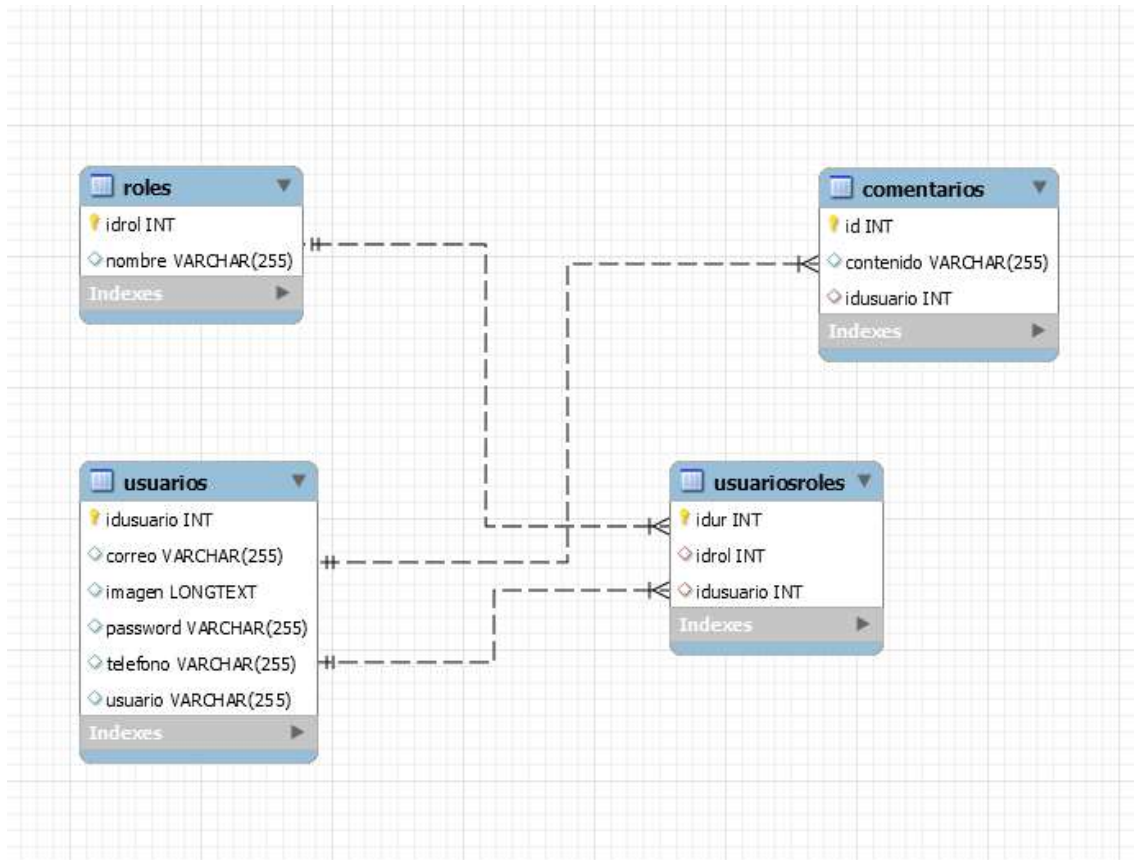
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(sui).passwordEncoder(encriptador());
    }
}
```

## 16.- Modelo de datos

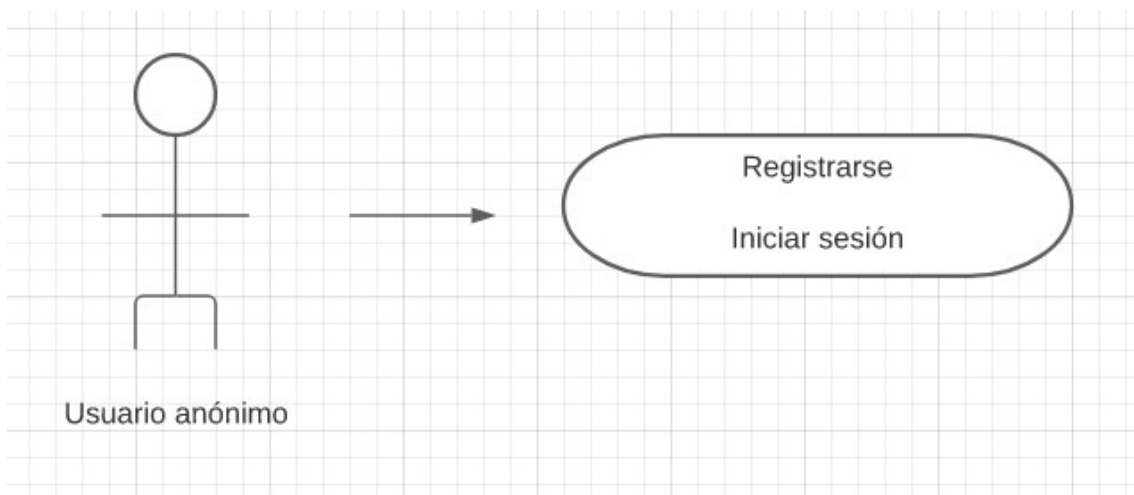
Se ha seguido un modelo Modelo-Vista-Controlador para la organización de los diferentes elementos que componen la totalidad del proyecto.

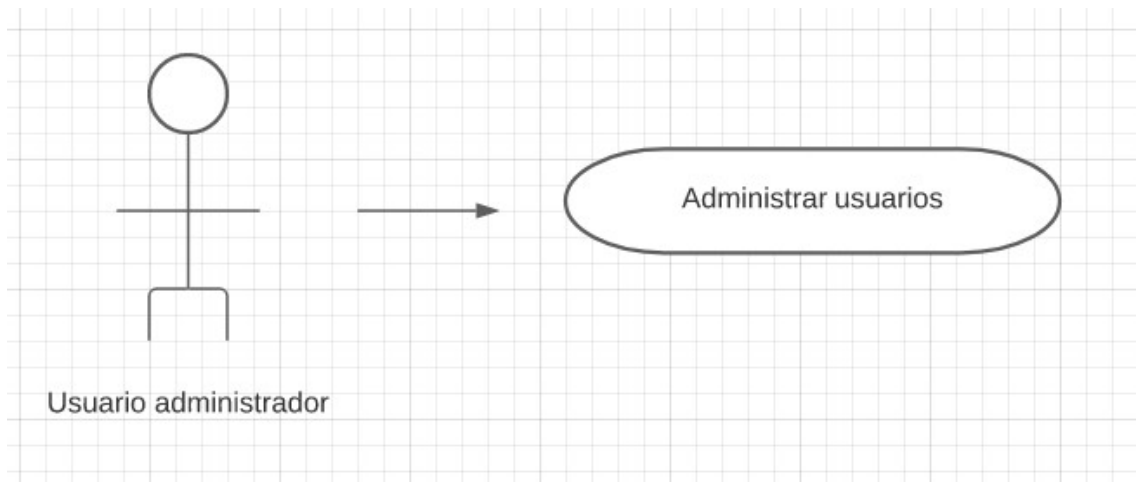
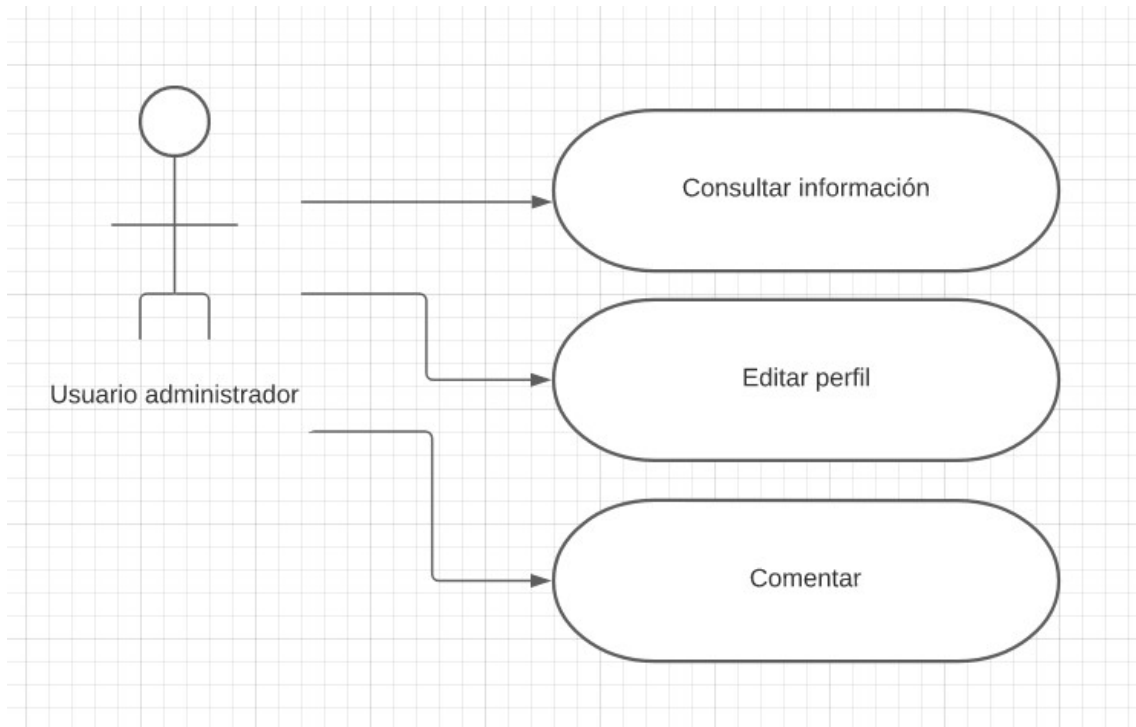


## 17.- Diagrama de clases



## 18.- Casos de uso





## 19.- Diagramas de secuencia

