



Índice

1. Título proyecto
 - 1.1. Desarrollo de interfaces
 - 1.2. Programación Web y Acceso a Datos
2. Planteamiento
3. Entorno de desarrollo
4. Desarrollo del proyecto
 - 4.1. Creación del proyecto y pasarlo a eclipse
 - 4.2. Estructura del proyecto (Back-End)
 - 4.2.1. Programación Back-End
 - 4.3. Estructura del proyecto (Front-End)
 - 4.3.1. Programación Front-End
 - 4.4. Conexión Front - Back
 - 4.5. API Flickr
 - 4.6. Login y Registrar
 - 4.7. CRUD Imágenes
 - 4.8. Generación de PDF (jsPDF)
 - 4.9. Comopdoc
 - 4.10. Usabilidad UX
 - 4.11. Accesibilidad

1. Título proyecto

1.1. Desarrollo de Interfaces

Para el apartado de Desarrollo de interfaces, el proyecto debe contar con una interfaz de usuario limpia y usable. Esta interfaz debe contar con un apartado login donde el usuario introduce “usuario” y “contraseña” para poder acceder a la aplicación. Dentro de la aplicación trabajaremos con dos partes, una dedicada a la gestión y tratamiento de las imágenes a través de un CRUD donde podremos insertar, borrar y modificar información referente a la imagen con la que estemos tratando. Aquí podréis organizarlo de la manera que más sencilla os parezca, mediante un formulario donde introduzcáis información, por ejemplo. La cuestión es que la información se trate correctamente y que para la vista del usuario sea lo más clara y entendible. Por otra parte, esta interfaz contará también con un buscador, en el que metiendo cualquier palabra clave nos mostrará las imágenes referentes a esa búsqueda, igualmente tenéis libertad en el diseño de este apartado, que aparezca la información en una tabla, en forma de cards etc... Además del buscador se añadirán al menos 3 botones de categorías pe: animales, flores y música, al pulsar uno de estos tres botones nos aparecerán todas las imágenes relacionadas con esta temática. Es importante que tengáis en cuenta en todo momento la usabilidad y la experiencia de usuario, recordad que la UX va un poco más allá y debemos pensar en la forma en la que el usuario se sentirá parte de nuestra aplicación, por ejemplo, un botón de favoritos en la imagen o de enviar a un amigo etc... Por otra parte, debéis generar un tipo informe de las imágenes en PDF con la librería jsPDF (podéis usar otra si encontráis alguna que se adapte mejor a la tecnología con la que vais a desarrollar la web), este informe contendrá información sobre las imágenes que trata la aplicación. Esta información la podéis mostrar en forma de tabla, imágenes... cuánto más completo sea mejor valoración tendrá. Para finalizar con la parte de desarrollo de interfaces, debéis generar la documentación de vuestro proyecto a través de compodoc (o la librería que lo permita en la tecnología que vayáis a implementar en la web). Es importante que comentáis cada parte del proyecto de forma que si otro compañero tuvo que continuar con el entendiera todo correctamente. Otra parte fundamental que tendréis que tener en cuenta

será el tema de la accesibilidad, debéis tener un apartado donde existan opciones que adapten la web a personas con algún tipo de discapacidad. Para ello podéis fijaros en las webs que hemos estado trabajando en la Práctica de Accesibilidad. El proyecto irá acompañado de la documentación pertinente.

1.2. Programación Web y Acceso a Datos

En el apartado de login, el usuario deberá poder entrar a la aplicación únicamente si existe dicho usuario y coincide su contraseña (log in). En caso de que no exista una cosa u otra, deberá indicarlo (validación). Si no hay ningún usuario, éste deberá poder crearse (sign in). Todos los usuarios existentes se almacenarán en una base de datos. La aplicación permitirá, una vez dentro, poder modificar datos del usuario, así como eliminarlo.

Como se indica en el apartado de Desarrollo de Interfaces, trabajaremos con dos partes:

1. Tendremos un buscador que se conectará a alguna API de imágenes existente online (nunca creada por nosotros). Mostrará los resultados y se podrán filtrar los mismos por categorías.
2. Una sección de tratamiento de imágenes donde podamos, por así decirlo, tener nuestra propia biblioteca de imágenes. En ella podremos añadir imágenes junto con su información, tags, descripción, etc. Todo esto podrá editarse posteriormente y eliminarse. Claro está, tanto imágenes como información deberá almacenarse en una base de datos.

La parte del BACK deberá trabajarse con Spring.

El ORM necesario para el mapeo de la base de datos (y con esto se indica que trabajaremos con objetos) será Hibernate.

La parte del FRONT podrá ser con Angular, React o cualquier otra tecnología que sirva para realizar dichas peticiones API y mostrarlas en el navegador.

El proyecto irá documentado siguiendo la misma metodología que en el apartado de Desarrollo de Interfaces.

2. Planteamiento

Mi planteamiento es realizar una web app, con angular y spring boot, que tenga un login (y su correspondiente registro, por si el usuario no tiene cuenta). Una vez el usuario esté dentro de la aplicación web, en un primer momento tendrá una vista de la API Flickr (junto con sus botones de categorías, de usabilidad y un buscador, aunque las imágenes se mostrarán sin tener que buscar nada en un principio). También tendrá un nav (donde se encontrarán el resto de partes de la web) y un footer. También estará nuestra página del CRUD de imágenes, donde podremos agregar, modificar, eliminar o actualizar el título y descripción de estas. Junto con esto habrá un botón para generar un PDF de los títulos y enlaces introducidos en el CRUD.



3. Entorno de desarrollo

Vamos a trabajar con Visual Studio Code (Versión: 1.64.2) con Node.js: 14.16.0.

Angular CLI: 12.2.16 con TypeScript (Versión: 4.3.5)

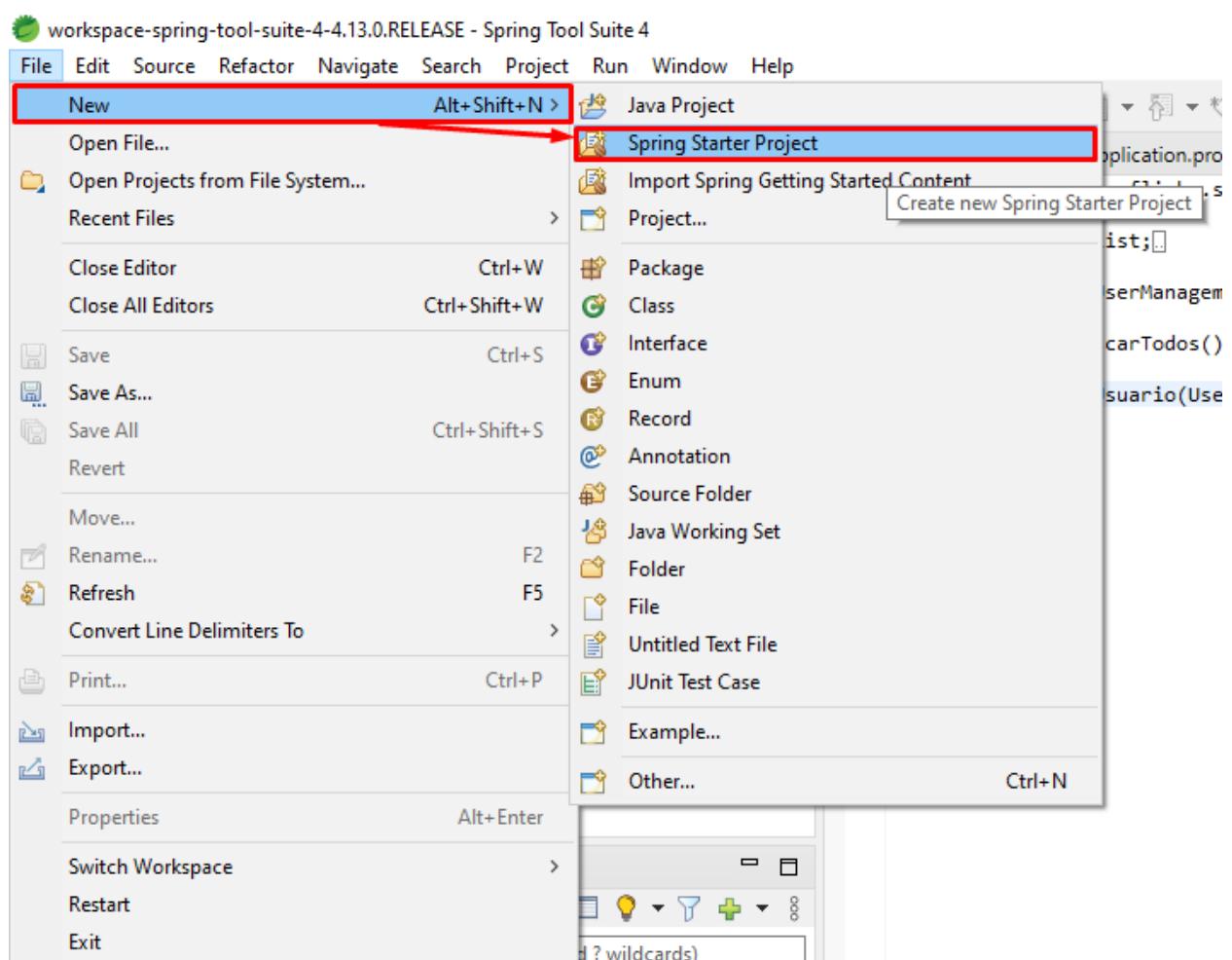
Eclipse (Versión: 2021-12 (4.22.0))

Java 1.8.0_311

4. Desarrollo del proyecto

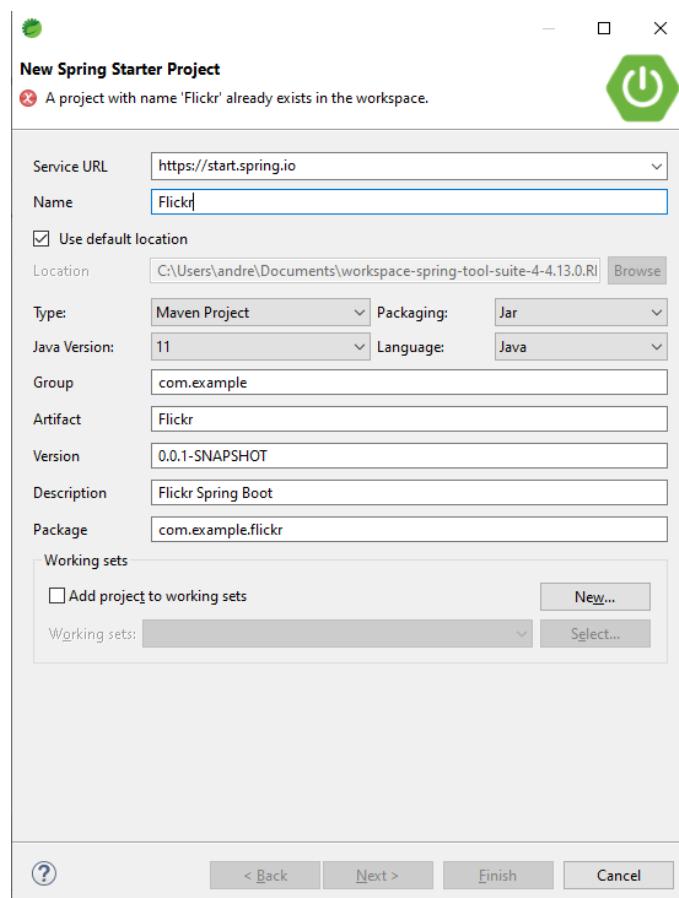
4.1. Creación del proyecto y pasarlo a eclipse

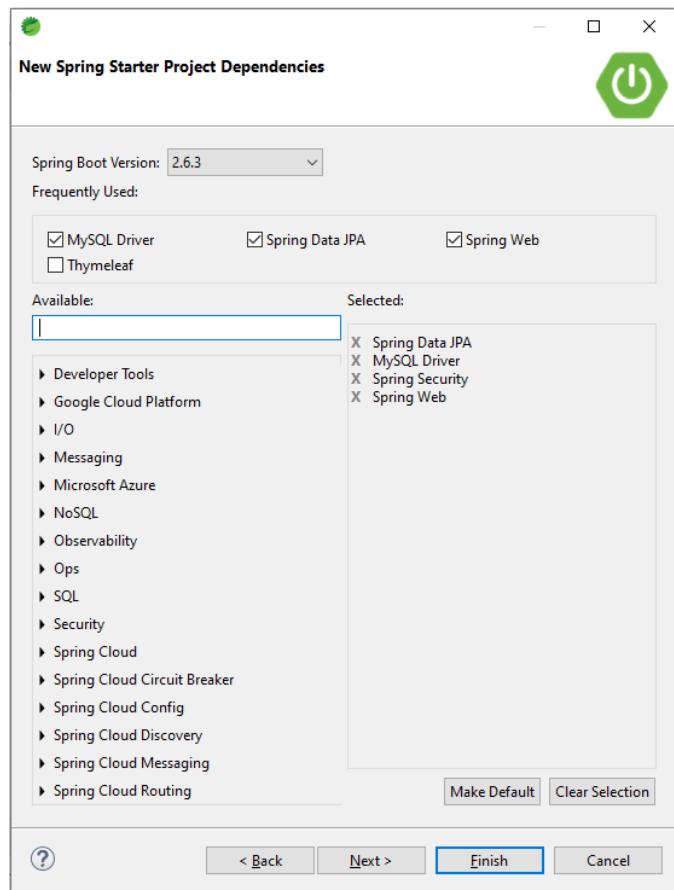
Para crear nuestro proyecto, lo primero que haremos será, abrir Spring Tool Suite 4 y crear un proyecto



Aunque podamos crear el proyecto directamente desde eclipse (con un plug-in), vamos a añadir las dependencias en este programa y lo pasaremos a Eclipse

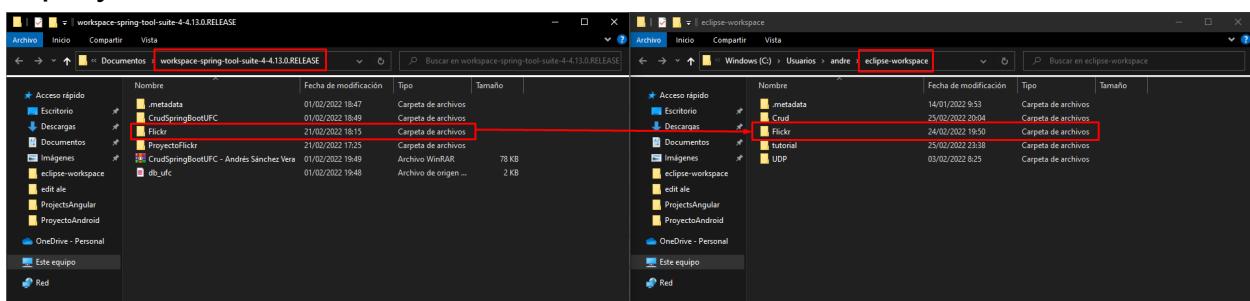
Le añadimos el nombre, la versión de java y el group, en la siguiente ventana añadiremos la dependencias





Aquí añadimos las dependencias necesarias para realizar el proyecto (si nos faltara alguna después lo podríamos añadir desde el archivo pom.xml).

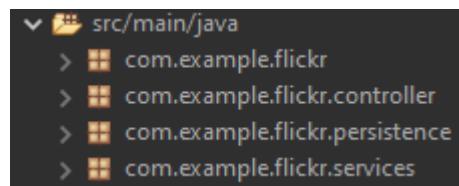
Una vez creado el Spring Starter Project, procedemos a pasarlo a Eclipse, esto se hará de manera rápida yendo a las dos ubicaciones y pasaremos el proyecto.



Una vez hecho esto abriremos Eclipse, e importamos el proyecto, le damos a File, Open Projects from File System..., Directory y seleccionamos la carpeta del proyecto.

4.2. Estructura del proyecto (Back-End)

Una vez dentro de Eclipse, vamos a crear la estructura de nuestra parte back. Esta estarán divididas en 4:



En nuestro primer paquete estará nuestro main class, lo que iniciará nuestro proyecto. El controller será nuestro controlador, encargado de dirigir la aplicación a un punto a otro. El paquete persistence donde estará los datos que trataremos, y los servicios, donde los creamos y implementaremos

4.2.1. Programación Back-End

Para explicar todo el Back - End voy subir capturas del código comentado, donde lo explico detalladamente

Primero vamos a explicar las anotaciones del proyecto

@RestController: para obtener datos o generar operaciones

@RequestMapping: Se utiliza para asignar solicitudes web a clases de controlador específicas y/o métodos de controlador.

@CrossOrigin: permite realizar conexiones con servidores ajenos

@Autowired: injectar unas dependencias con otras dentro de Spring

@GetMapping: nos permite simplificar el manejo de los diferentes métodos

@PostMapping: se utilizan para enviar información al servidor

@DeleteMapping: se utilizan para eliminar información del servidor

@PutMapping: se utilizan para actualizar información al servidor

@SpringBootApplication: Sin esto la aplicación no arrancaría, es lo que le dice al programa qué tipo de aplicación es

@Override: sirve para forzar al compilador a comprobar en tiempo de compilación que estás sobrescribiendo, para evitar errores

@Entity: Es esencialmente un nombre, o un conjunto de estados (atributos) asociados juntos en una unidad.

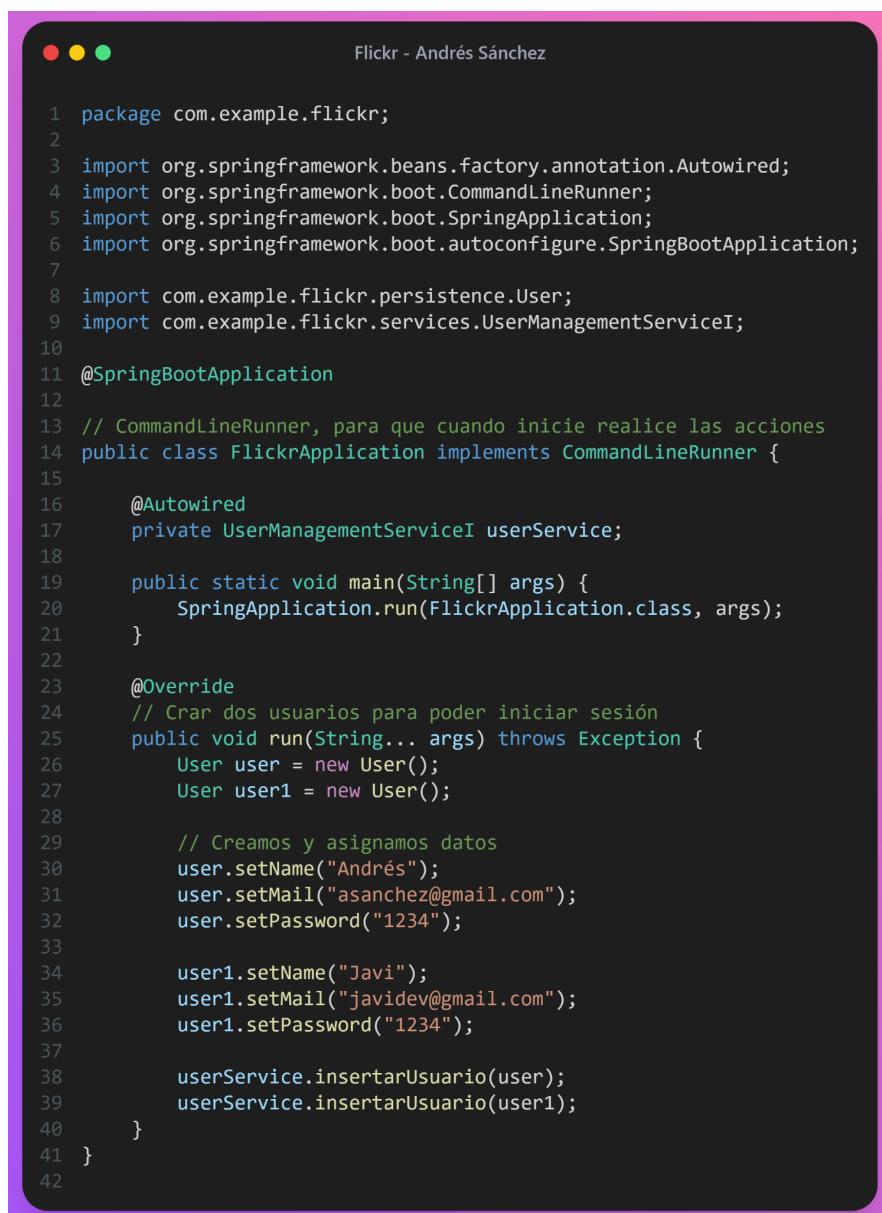
@Table: Índica sobre que tabla se va a trabajar

@Id: sirve para definir el identificador único de cada Entidad

@GeneratedValue: se usa para que se autoincremente el id

@Column: Genera una columna

@Service: Se usa para construir una clase de Servicio

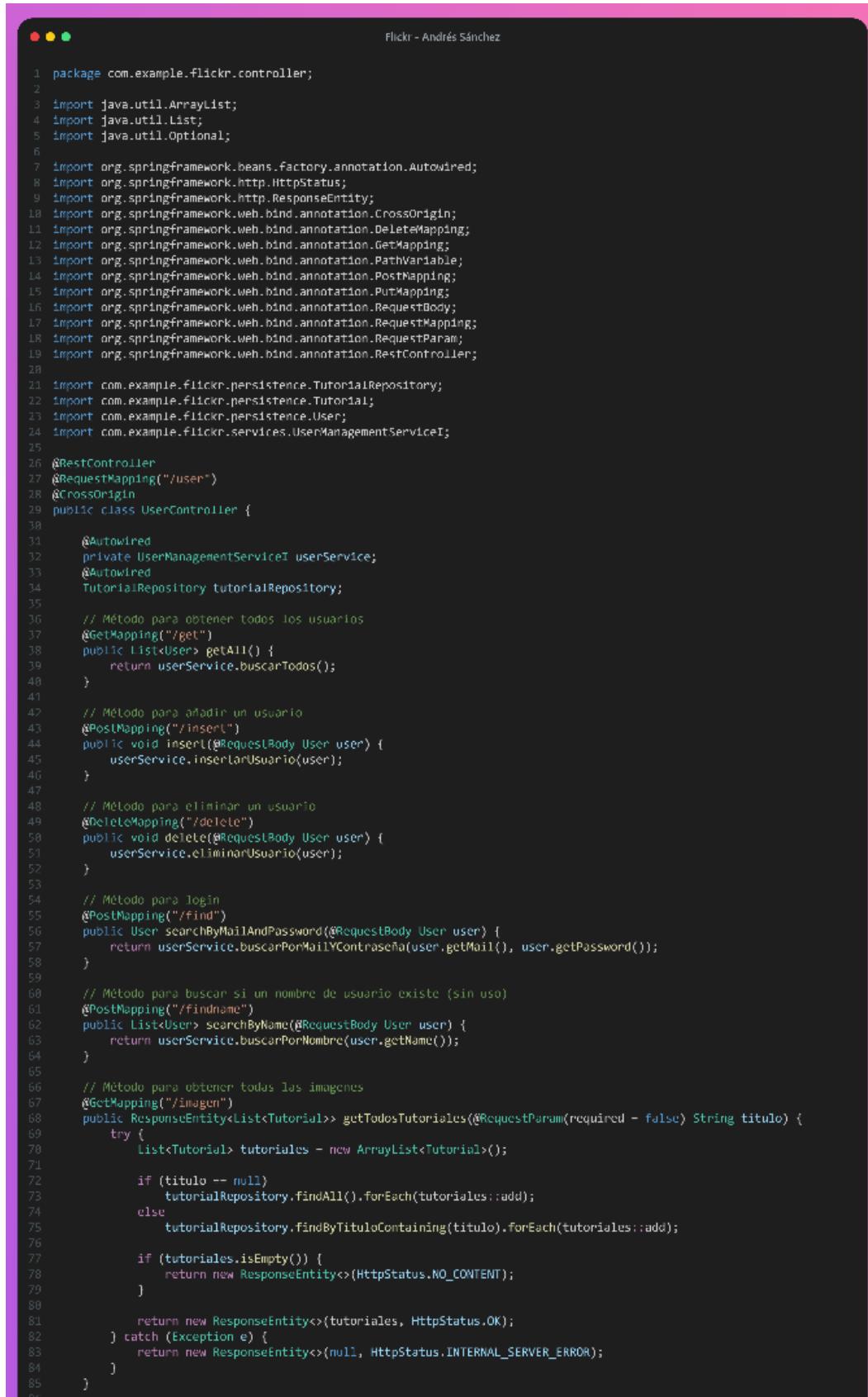


```

1 package com.example.flickr;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.boot.CommandLineRunner;
5 import org.springframework.boot.SpringApplication;
6 import org.springframework.boot.autoconfigure.SpringBootApplication;
7
8 import com.example.flickr.persistence.User;
9 import com.example.flickr.services.UserManagementServiceI;
10
11 @SpringBootApplication
12
13 // CommandLineRunner, para que cuando inicie realice las acciones
14 public class FlickrApplication implements CommandLineRunner {
15
16     @Autowired
17     private UserManagementServiceI userService;
18
19     public static void main(String[] args) {
20         SpringApplication.run(FlickrApplication.class, args);
21     }
22
23     @Override
24     // Crear dos usuarios para poder iniciar sesión
25     public void run(String... args) throws Exception {
26         User user = new User();
27         User user1 = new User();
28
29         // Creamos y asignamos datos
30         user.setName("Andrés");
31         user.setMail("asanchez@gmail.com");
32         user.setPassword("1234");
33
34         user1.setName("Javi");
35         user1.setMail("javidev@gmail.com");
36         user1.setPassword("1234");
37
38         userService.insertarUsuario(user);
39         userService.insertarUsuario(user1);
40     }
41 }
42

```

FlickrApplication.java



```

1 package com.example.flickr.controller;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Optional;
6
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.HttpStatus;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.web.bind.annotation.CrossOrigin;
11 import org.springframework.web.bind.annotation.DeleteMapping;
12 import org.springframework.web.bind.annotation.GetMapping;
13 import org.springframework.web.bind.annotation.PathVariable;
14 import org.springframework.web.bind.annotation.PostMapping;
15 import org.springframework.web.bind.annotation.PutMapping;
16 import org.springframework.web.bind.annotation.RequestBody;
17 import org.springframework.web.bind.annotation.RequestMapping;
18 import org.springframework.web.bind.annotation.RequestParam;
19 import org.springframework.web.bind.annotation.RestController;
20
21 import com.example.flickr.persistence.TutorialRepository;
22 import com.example.flickr.persistence.Tutorial;
23 import com.example.flickr.persistence.User;
24 import com.example.flickr.services.UserManagementServiceI;
25
26 @RestController
27 @RequestMapping("/user")
28 @CrossOrigin
29 public class UserController {
30
31     @Autowired
32     private UserManagementServiceI userService;
33     @Autowired
34     TutorialRepository tutorialRepository;
35
36     // Método para obtener todos los usuarios
37     @GetMapping("/get")
38     public List<User> getAll() {
39         return userService.buscarTodos();
40     }
41
42     // Método para añadir un usuario
43     @PostMapping("/insert")
44     public void insert(@RequestBody User user) {
45         userService.insertarUsuario(user);
46     }
47
48     // Método para eliminar un usuario
49     @DeleteMapping("/delete")
50     public void delete(@RequestBody User user) {
51         userService.eliminarUsuario(user);
52     }
53
54     // Método para login
55     @PostMapping("/find")
56     public User searchByMailAndPassword(@RequestBody User user) {
57         return userService.buscarPorMailYContraseña(user.getMail(), user.getPassword());
58     }
59
60     // Método para buscar si un nombre de usuario existe (sin uso)
61     @PostMapping("/findname")
62     public List<User> searchByName(@RequestBody User user) {
63         return userService.buscarPorNombre(user.getName());
64     }
65
66     // Método para obtener todas las imágenes
67     @GetMapping("/imagen")
68     public ResponseEntity<List<Tutorial>> getTodosTutorialios(@RequestParam(required = false) String titulo) {
69         try {
70             List<Tutorial> tutoriales = new ArrayList<Tutorial>();
71
72             if (titulo == null)
73                 tutorialRepository.findAll().forEach(tutoriales::add);
74             else
75                 tutorialRepository.findByTituloContaining(titulo).forEach(tutoriales::add);
76
77             if (tutoriales.isEmpty())
78                 return new ResponseEntity<>(HttpStatus.NO_CONTENT);
79         }
80
81         return new ResponseEntity<List<Tutorial>>(tutoriales, HttpStatus.OK);
82     } catch (Exception e) {
83         return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
84     }
85 }
86

```

UserController.java (1/2)

```

86     // Metodo para obtener el id de la imagen
87     @GetMapping("/imagen/{id}")
88     public ResponseEntity<Tutorial> getTutorialPorId(@PathVariable("id") long id) {
89         Optional<Tutorial> tutorialData = tutorialRepository.findById(id);
90
91         if (tutorialData.isPresent()) {
92             return new ResponseEntity<Tutorial>(tutorialData.get(), HttpStatus.OK);
93         } else {
94             return new ResponseEntity<Tutorial>(HttpStatus.NOT_FOUND);
95         }
96     }
97
98     // Metodo para crear una imagen
99     @PostMapping("/imagen")
100    public ResponseEntity<Tutorial> crearTutorial(@RequestBody Tutorial tutorial) {
101        try {
102            Tutorial _tutorial = tutorialRepository
103                .save(new Tutorial(tutorial.getTitulo(), tutorial.getDescripcion(), false));
104            return new ResponseEntity<Tutorial>(_tutorial, HttpStatus.CREATED);
105        } catch (Exception e) {
106            return new ResponseEntity<Tutorial>(null, HttpStatus.INTERNAL_SERVER_ERROR);
107        }
108    }
109
110    // Metodo para actualizar la imagen
111    @PutMapping("/imagen/{id}")
112    public ResponseEntity<Tutorial> actualizarTutorial(@PathVariable("id") long id, @RequestBody Tutorial tutorial) {
113        Optional<Tutorial> tutorialData = tutorialRepository.findById(id);
114
115        if (tutorialData.isPresent()) {
116            Tutorial _tutorial = tutorialData.get();
117            _tutorial.setTitulo(tutorial.getTitulo());
118            _tutorial.setDescripcion(tutorial.getDescripcion());
119            _tutorial.setPublicado(tutorial.esPublicado());
120            return new ResponseEntity<Tutorial>(tutorialRepository.save(_tutorial), HttpStatus.OK);
121        } else {
122            return new ResponseEntity<Tutorial>(HttpStatus.NOT_FOUND);
123        }
124    }
125
126    // Metodo para eliminar una imagen
127    @DeleteMapping("/imagen/{id}")
128    public ResponseEntity<HttpStatus> eliminarTutorial(@PathVariable("id") long id) {
129        try {
130            tutorialRepository.deleteById(id);
131            return new ResponseEntity<HttpStatus>(HttpStatus.NO_CONTENT);
132        } catch (Exception e) {
133            return new ResponseEntity<HttpStatus>(HttpStatus.INTERNAL_SERVER_ERROR);
134        }
135    }
136
137    // Metodo para eliminar todos los tutoriales
138    @DeleteMapping("/imagen")
139    public ResponseEntity<HttpStatus> eliminarTodosTutoriales() {
140        try {
141            tutorialRepository.deleteAll();
142            return new ResponseEntity<HttpStatus>(HttpStatus.NO_CONTENT);
143        } catch (Exception e) {
144            return new ResponseEntity<HttpStatus>(HttpStatus.INTERNAL_SERVER_ERROR);
145        }
146    }
147
148
149    // Metodo para publicar / despublicar (sin usar)
150    @GetMapping("/imagen/publicado")
151    public ResponseEntity<List<Tutorial>> findByPublished() {
152        try {
153            List<Tutorial> tutoriales = tutorialRepository.findByPublished(true);
154
155            if (tutoriales.isEmpty()) {
156                return new ResponseEntity<List<Tutorial>>(HttpStatus.NO_CONTENT);
157            }
158            return new ResponseEntity<List<Tutorial>>(tutoriales, HttpStatus.OK);
159        } catch (Exception e) {
160            return new ResponseEntity<List<Tutorial>>(HttpStatus.INTERNAL_SERVER_ERROR);
161        }
162    }
163
164}
165

```

UserController.java (2/2)



```

1 package com.example.flickr.persistence;
2
3 import javax.persistence.Column;
4 import javax.persistence.Entity;
5 import javax.persistence.GeneratedValue;
6 import javax.persistence.GenerationType;
7 import javax.persistence.Id;
8 import javax.persistence.Table;
9
10 // Definimos el nombre de nuestra tabla
11 @Entity
12 @Table(name = "tutoriales")
13 public class Tutorial {
14
15     // Generamos el ID y lo auto-incrementamos
16     @Id
17     @GeneratedValue(strategy = GenerationType.AUTO)
18     private long id;
19
20     // Creamos nuestras columnas
21     @Column(name = "titulo")
22     private String titulo;
23
24     @Column(name = "descripcion")
25     private String descripcion;
26
27     @Column(name = "publicado")
28     private boolean publicado;
29
30     public Tutorial() {
31
32     }
33
34     // Pasamos los atributos
35     public Tutorial(String titulo, String descripcion, boolean publicado) {
36         this.titulo = titulo;
37         this.descripcion = descripcion;
38         this.publicado = publicado;
39     }
40
41     // Creamos los gettes and setters
42     public long getId() {
43         return id;
44     }
45
46     public String getTitulo() {
47         return titulo;
48     }
49
50     public void setTitulo(String titulo) {
51         this.titulo = titulo;
52     }
53
54     public String getDescripcion() {
55         return descripcion;
56     }
57
58     public void setDescripcion(String descripcion) {
59         this.descripcion = descripcion;
60     }
61
62     public boolean esPublicado() {
63         return publicado;
64     }
65
66     public void setPublicado(boolean esPublicado) {
67         this.publicado = esPublicado;
68     }
69
70     // Y los pasamos a String
71     @Override
72     public String toString() {
73         return "Tutorial [id= " + id + ", titulo= " + titulo + ", desc= " + descripcion + ", publicado= " + publicado
74             + "]";
75     }
76 }

```

Tutorial.java

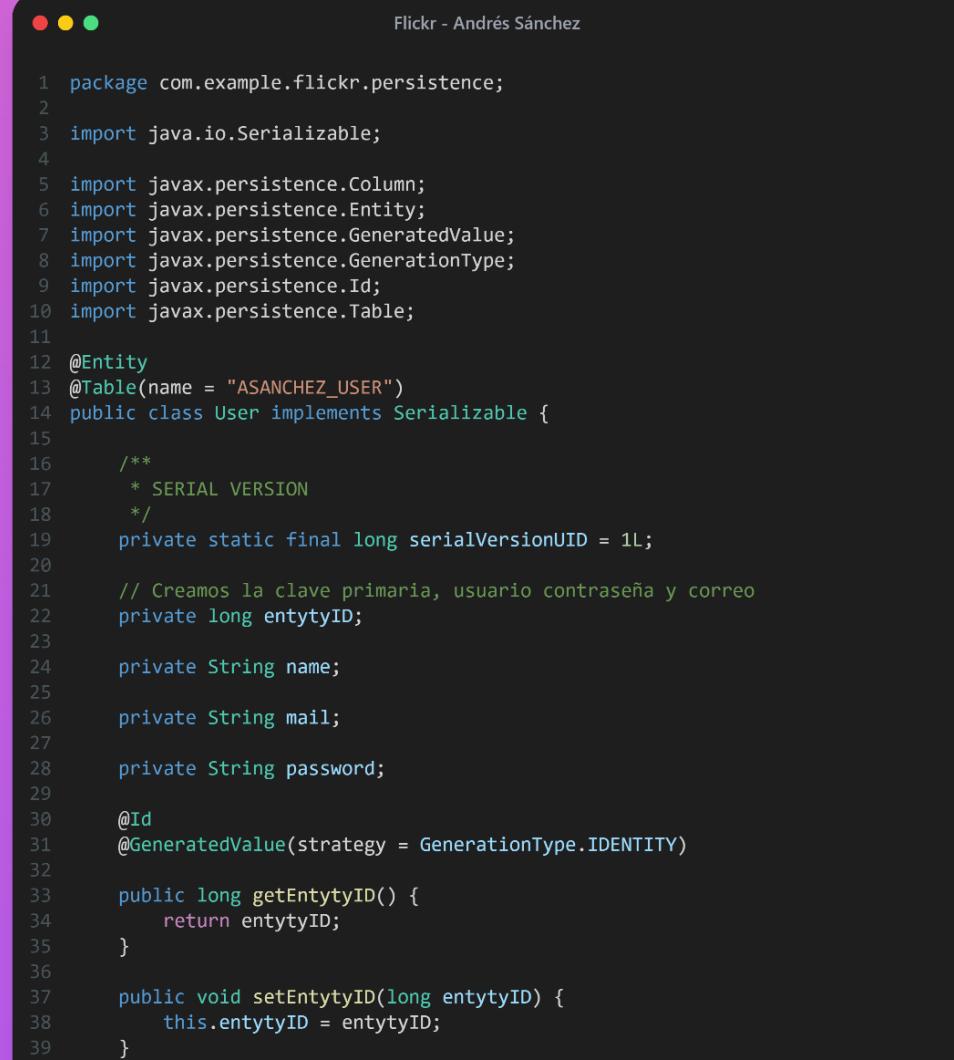


```

1 package com.example.flickr.persistence;
2
3 import java.util.List;
4
5 import org.springframework.data.jpa.repository.JpaRepository;
6
7 // Creamos el repositorio, los métodos que vamos a usar, con JpaRepository tenemos
8 // métodos como add, eliminate, actualizar, modificar
9 public interface TutorialRepository extends JpaRepository<Tutorial, Long> {
10     List<Tutorial> findByPublicado(boolean publicado);
11
12     List<Tutorial> findByTituloContaining(String titulo);
13 }

```

TutorialRepository.java



```

1 package com.example.flickr.persistence;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Column;
6 import javax.persistence.Entity;
7 import javax.persistence.GeneratedValue;
8 import javax.persistence.GenerationType;
9 import javax.persistence.Id;
10 import javax.persistence.Table;
11
12 @Entity
13 @Table(name = "ASANCHEZ_USER")
14 public class User implements Serializable {
15
16     /**
17      * SERIAL VERSION
18      */
19     private static final long serialVersionUID = 1L;
20
21     // Creamos la clave primaria, usuario contraseña y correo
22     private long entytyID;
23
24     private String name;
25
26     private String mail;
27
28     private String password;
29
30     @Id
31     @GeneratedValue(strategy = GenerationType.IDENTITY)
32
33     public long getEntytyID() {
34         return entytyID;
35     }
36
37     public void setEntytyID(long entytyID) {
38         this.entytyID = entytyID;
39     }
40

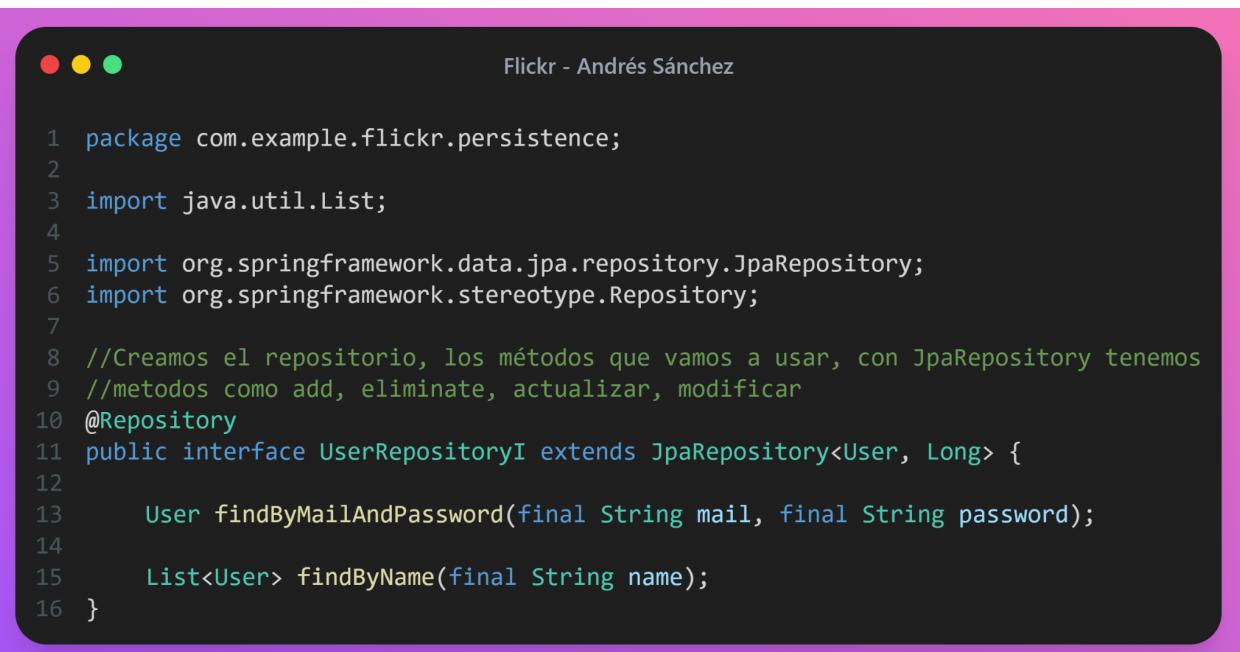
```

User.java (1/2)

```

41     // Creamos los getter and setter y asignamos la columna dentro de la tabla
42     @Column(name = "NAME", nullable = false)
43     public String getName() {
44         return name;
45     }
46
47     public void setName(String name) {
48         this.name = name;
49     }
50
51     @Column(name = "MAIL", nullable = false)
52     public String getMail() {
53         return mail;
54     }
55
56     public void setMail(String mail) {
57         this.mail = mail;
58     }
59
60     @Column(name = "PASSWORD", nullable = false)
61     public String getPassword() {
62         return password;
63     }
64
65     public void setPassword(String password) {
66         this.password = password;
67     }
68 }
69

```

User.java (2/2)


Flickr - Andrés Sánchez

```

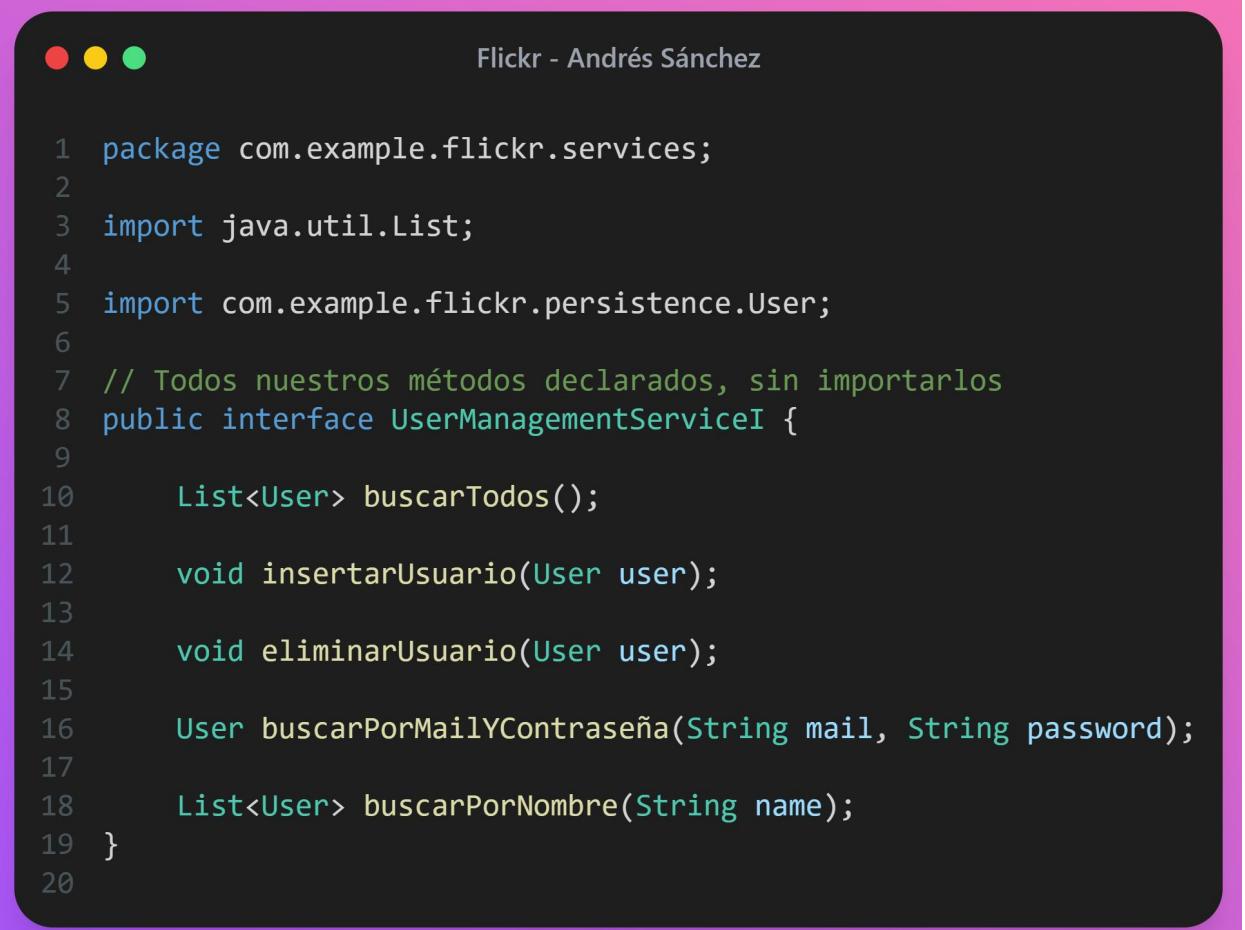
1 package com.example.flickr.persistence;
2
3 import java.util.List;
4
5 import org.springframework.data.jpa.repository.JpaRepository;
6 import org.springframework.stereotype.Repository;
7
8 //Creamos el repositorio, los métodos que vamos a usar, con JpaRepository tenemos
9 //métodos como add, eliminar, actualizar, modificar
10 @Repository
11 public interface UserRepositoryI extends JpaRepository<User, Long> {
12
13     User findByMailAndPassword(final String mail, final String password);
14
15     List<User> findByName(final String name);
16 }

```

UserRepositoryI.java

```
1 package com.example.flickr.services;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import com.example.flickr.persistence.User;
9 import com.example.flickr.persistence.UserRepositoryI;
10
11 // Todos nuestros metodos implementados
12 @Service
13 public class UserManagementServiceImpl implements UserManagementServiceI {
14
15     // Llamamos a nuestro repositorio
16     @Autowired
17     private UserRepositoryI userRepo;
18
19     // Buscar todos los usuarios
20     @Override
21     public List<User> buscarTodos() {
22         return userRepo.findAll();
23     }
24
25     // Añadir un usuario
26     @Override
27     public void insertarUsuario(User user) {
28         userRepo.save(user);
29     }
30
31     // Eliminar un usuario
32     @Override
33     public void eliminarUsuario(User user) {
34         userRepo.delete(user);
35     }
36
37     // Login
38     @Override
39     public User buscarPorMailYContraseña(String mail, String password) {
40         return userRepo.findByMailAndPassword(mail, password);
41     }
42
43     // Buscar por nombre
44     @Override
45     public List<User> buscarPorNombre(String name) {
46         return userRepo.findByName(name);
47     }
48 }
```

UserManagementServiceImpl.java



```

1 package com.example.flickr.services;
2
3 import java.util.List;
4
5 import com.example.flickr.persistence.User;
6
7 // Todos nuestros métodos declarados, sin importarlos
8 public interface UserManagementServiceI {
9
10     List<User> buscarTodos();
11
12     void insertarUsuario(User user);
13
14     void eliminarUsuario(User user);
15
16     User buscarPorMailYContraseña(String mail, String password);
17
18     List<User> buscarPorNombre(String name);
19 }
20

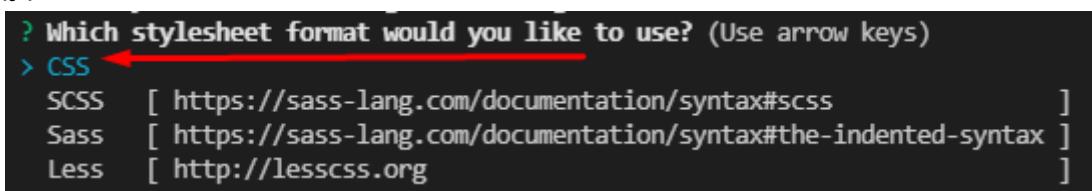
```

UserManagementServiceI.java

4.3. Estructura del proyecto (Front-End)

Para empezar nuestro front del proyecto, abriremos Visual Studio Code.

PS C:\Users\andre\Desktop\ProjectsAngular> `ng new FlickrAndres` crearemos un proyecto nuevo ? Would you like to add Angular routing? (y/N) , seleccionamos yes (y).

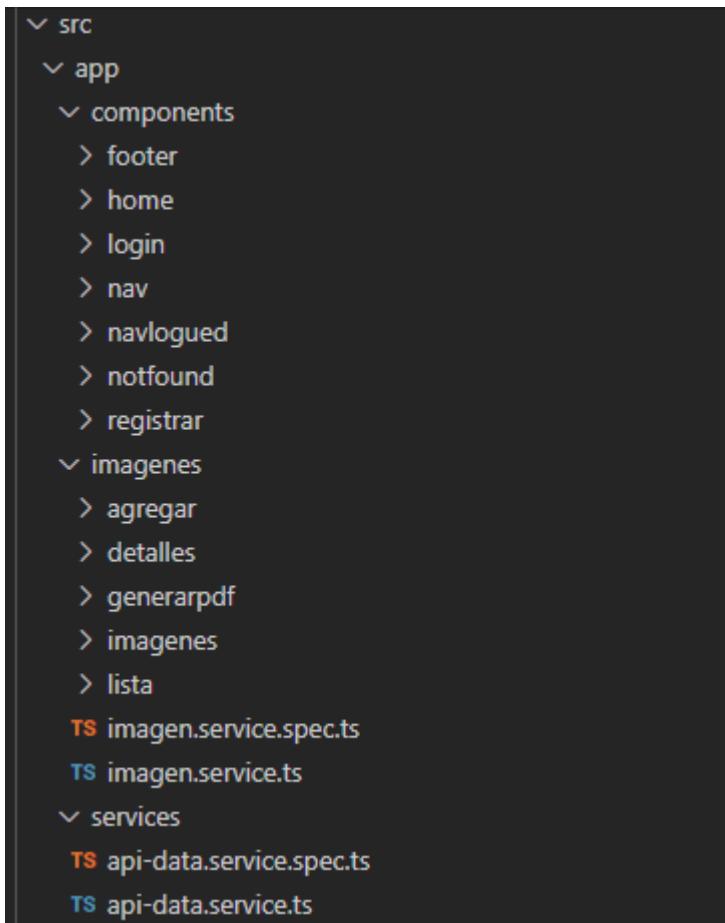


```

? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS  [ https://sass-lang.com/documentation/syntax#scss ]
  Sass   [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less   [ http://lesscss.org ]

```

Por último seleccionamos CSS como hojas de estilos



Nuestra estructura del Front será dos grandes carpetas, components, donde tendremos todo nuestro login, registrar, y elementos de la api y elementos como el nav y footer y otra carpeta, imágenes, que tendremos todo lo relacion con el CRUD de imágenes y generación del PDF junto con el servicio de back del crud (imagen.service.ts) y una carpeta de services que tendremos la API de Flickr y el login y register.

Para crear un componente solo deberemos poner en nuestro terminal: ng g c (de generate component), nombreComponente. Si la queremos meter dentro de una carpeta ng g c nombreCarpeta/nombreComponente.

Para generar un servicio deberemos escribir: ng g s (de generate service), nombreServicio, para crearlo dentro de una carpeta será exactamente igual que con los componentes.

4.3.1. Programación Front-End

Para nuestro front, hecho en HTML para la estructura, CSS para los estilos y TypeScript para la lógica.

Para el enrutamiento, parte crucial de nuestra aplicación web, deberemos ir a app-routing.module.ts, importar todos nuestros componentes,

```
import { NombreComponent } from './components/nombre/nombre.component';
```

y añadirlo a las rutas:

```
const routes: Routes = [
  { path: 'nombre', component: NombreComponent },
  { path: '', redirectTo: '/inicio', pathMatch: 'full' },
  { path: '**', component: NotfoundComponent }
];
```

Para explicar esto, las rutas son las direcciones donde nos dirigiremos para movernos en nuestra web, el path es el nombre de la ruta, es decir lo que en un futuro usaremos con /nombre. la ruta vacía, "", significa que cuando URL se encuentre vacía, nos redirigirá a la ruta que tenga como nombre inicio y la ruta con nombre **, nos redirigirá a un componente que hemos creado para solucionar el error 404.

Para ayudar a maquetar la web voy a utilizar materialize, con nuestro terminal abierto, ingresamos el siguiente comando:

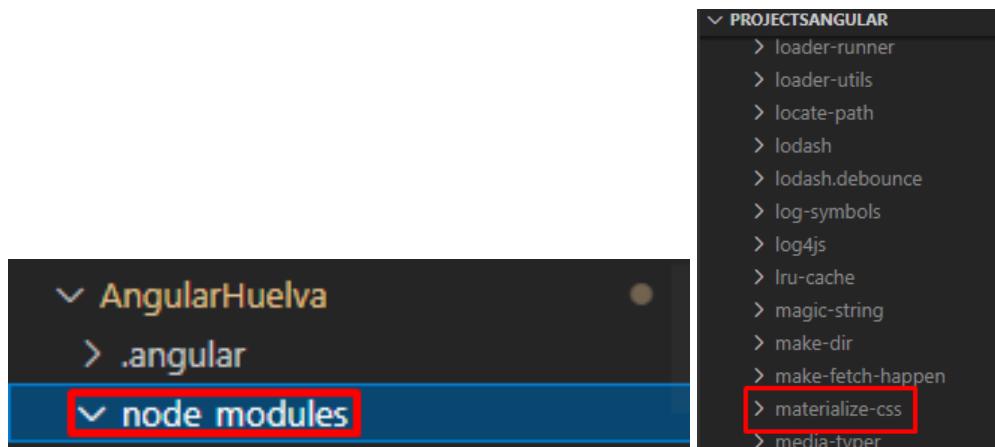
```
npm install materialize-css@next
```

```
PS C:\Users\andre\Desktop\ProjectsAngular\AngularHuelva> npm install materialize-css@next
added 1 package, removed 1 package, and audited 1027 packages in 12s

90 packages are looking for funding
  run `npm fund` for details

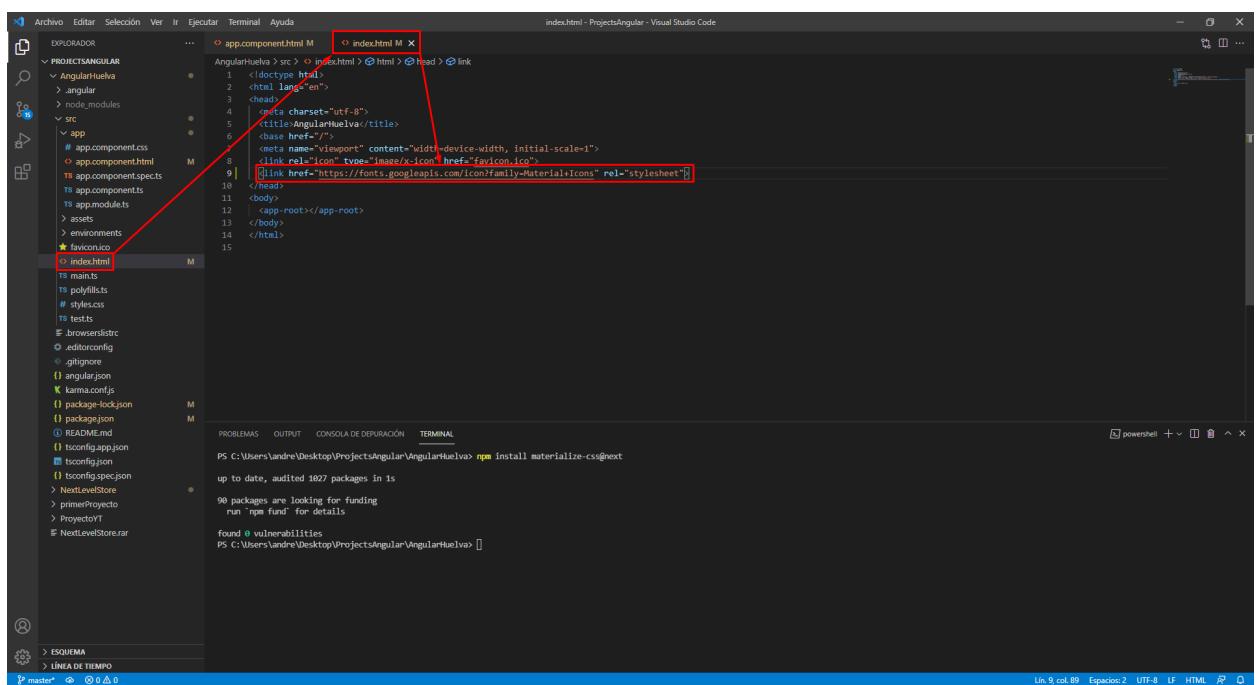
found 0 vulnerabilities
PS C:\Users\andre\Desktop\ProjectsAngular\AngularHuelva> []
```

Para comprobar si se ha instalado correctamente, nos vamos a node_modules y buscamos materialize-css



Una vez hecho esto, vamos a añadir los íconos de materialize con

```
<link
  href="https://fonts.googleapis.com/icon?family=Materia
  l+Icons" rel="stylesheet">
```



Para finalizar la instalación de materialize, deberemos incluirlo en las preferencias de los estilos en el angular.json, añadiendo lo siguiente:

Así es como esta por defecto:

```
"styles": [  
    "src/styles.css"  
],  
"scripts": []
```

Y así hay que dejarlo:

```
"styles": [  
    "./node_modules/materialize-css/dist/css/materialize.css",  
    "src/styles.css"  
],  
"scripts": [  
    "./node_modules/materialize-css/dist/js/materialize.js"  
]
```

Una vez instalado Materialize, podemos irnos a la documentación oficial, <https://materializecss.com/>

Otra cosa que debemos hacer es en el app.component.html es poner

```
<router-outlet></router-outlet>
```

Para que nos deje moverlos con nuestras rutas.



Proyecto Flickr - Andrés Sánchez

```

1 <!--Importaciones-->
2 <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
3 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-beta/css/materialize.min.css">
4 <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-beta/js/materialize.min.js"></script>
5 <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
6
7 <!--Nav-->
8 <nav>
9   <div class="nav-wrapper black">
10    <a id="text" onclick="M.toast({html: 'Debes iniciar sesion...'})" class="brand-logo">flick<span
11      id="red">r</span></a>
12    <a href="#" data-target="#mobile-demo" class="sidenav-trigger"><i class="material-icons">menu</i></a>
13    <ul class="right hide-on-med-and-down">
14      <li><a routerLink="/login">Iniciar Sesión</a></li>
15      <li><a routerLink="/signin">Crear Cuenta</a></li>
16    </ul>
17  </div>
18 </nav>
19
20 <!--Nav Mobile-->
21 <ul class="sidenav" id="mobile-demo">
22   <li><a routerLink="/signin">Iniciar Sesión</a></li>
23   <li><a routerLink="/login">Crear Cuenta</a></li>
24 </ul>

```

nav.component.html



Proyecto Flickr - Andrés Sánchez

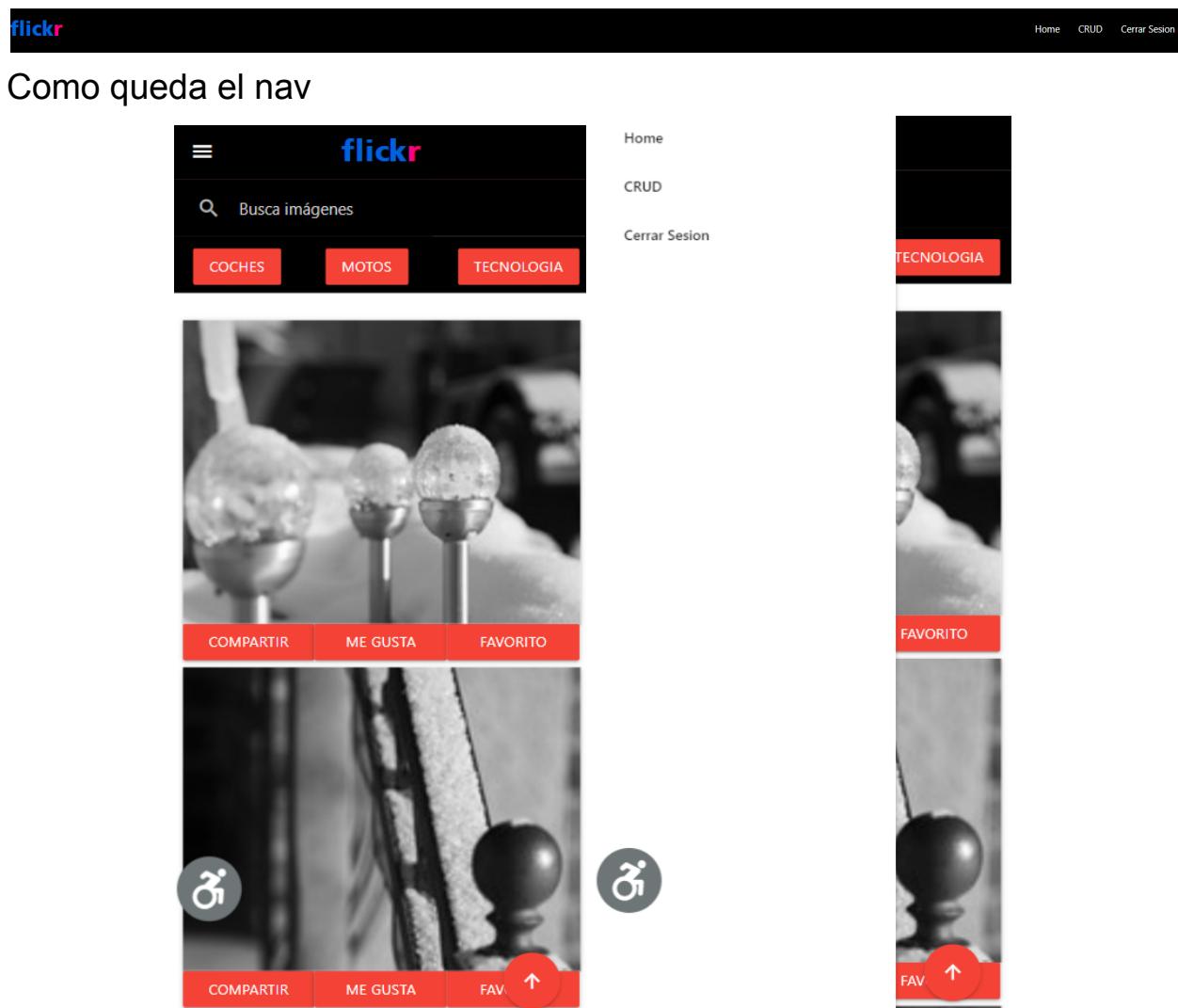
```

1 import { Component, OnInit } from '@angular/core';
2 import * as M from "materialize-css";
3
4 @Component({
5   selector: 'app-nav',
6   templateUrl: './nav.component.html',
7   styleUrls: ['./nav.component.css']
8 })
9 export class NavComponent implements OnInit {
10
11   constructor() { }
12
13   ngOnInit(): void {
14     M.AutoInit();
15   }
16
17   // Método para activar el side nav en versión móvil
18   public nav() {
19     $(document).ready(function () {
20       $('.sidenav').sidenav();
21     });
22   }
23 }

```

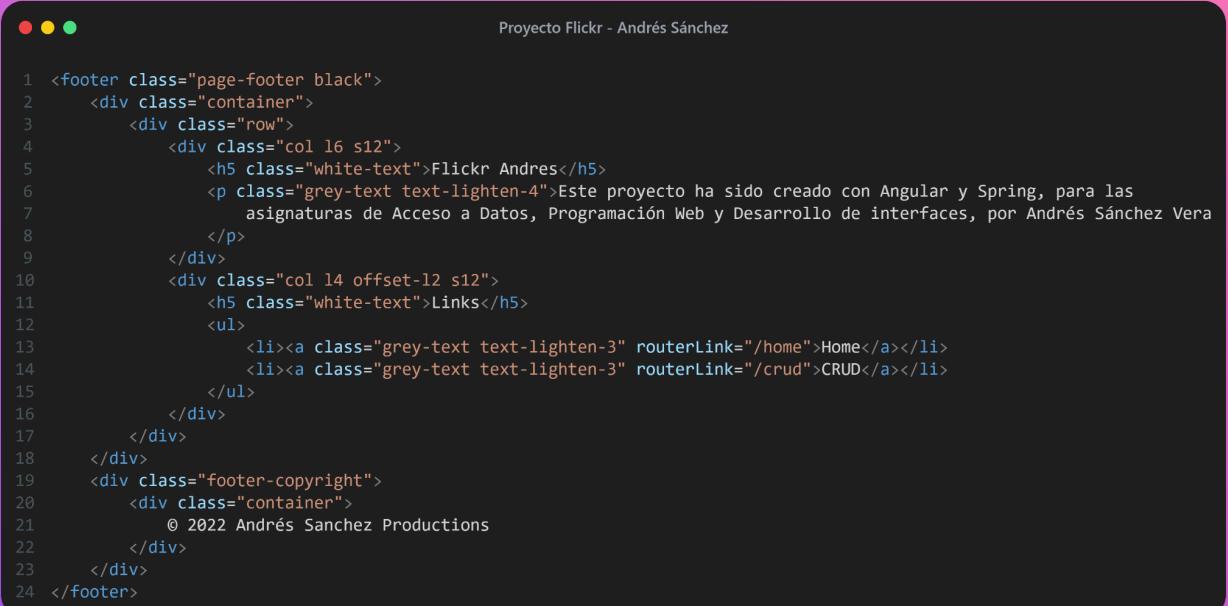
nav.component.ts

Para el nav debemos inicializar para que en la versión móvil se nos muestre el menú de hamburguesa



Y como queda el nav en versión móvil

Para el footer he elegido uno sencillo,

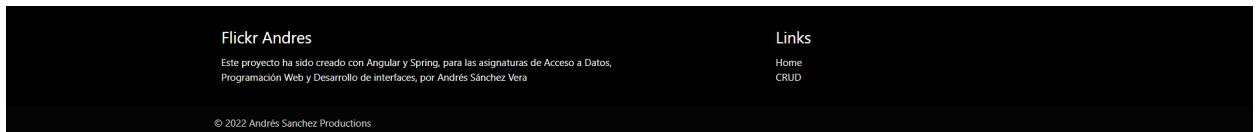


```

 1 <footer class="page-footer black">
 2   <div class="container">
 3     <div class="row">
 4       <div class="col 16 s12">
 5         <h5 class="white-text">Flickr Andres</h5>
 6         <p class="grey-text text-lighten-4">Este proyecto ha sido creado con Angular y Spring, para las
 7           asignaturas de Acceso a Datos, Programación Web y Desarrollo de interfaces, por Andrés Sánchez Vera
 8         </p>
 9       </div>
10      <div class="col 14 offset-12 s12">
11        <h5 class="white-text">Links</h5>
12        <ul>
13          <li><a class="grey-text text-lighten-3" routerLink="/home">Home</a></li>
14          <li><a class="grey-text text-lighten-3" routerLink="/crud">CRUD</a></li>
15        </ul>
16      </div>
17    </div>
18  </div>
19  <div class="footer-copyright">
20    <div class="container">
21      © 2022 Andrés Sanchez Productions
22    </div>
23  </div>
24 </footer>

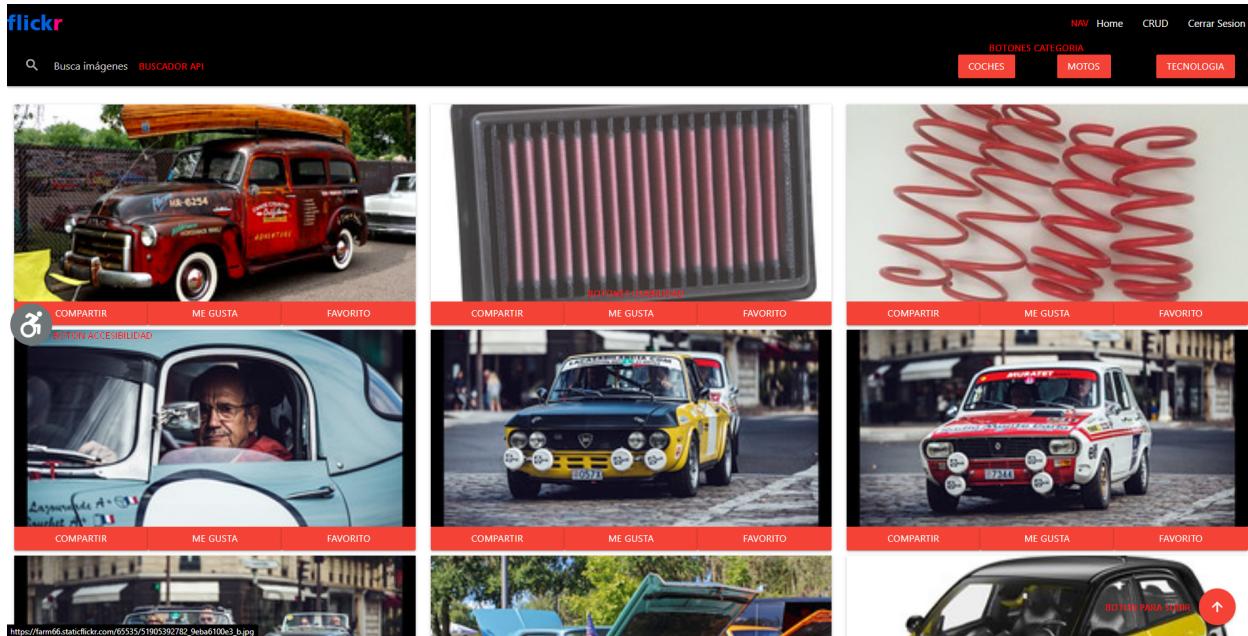
```

Que quedaría así



Con materialize tenemos una gran ventaja para crear nuestra aplicación responsive, ya que nos divide la pantalla en 12 columnas y con los tamaños large (l), medium (m) y small (s), es muy sencillo crearla.

Para no ir componente a componente explicando, ya que cada componente se compone de 4 archivos a su vez, vamos a ir viendo una vista general de cada uno, para ver todo el código HTML comentado, por favor visite [LINK GITHUB](#), nuestra parte lógica se explicar en [4.4. Conexión Front - Back](#)



En el home podemos ver la API de Flickr, que explicaremos después

Buscar por título BUSCAR

Lista de imágenes

BMW S1000RR	ELIMINAR TODO	AGREGAR	GENERAR PDF
LISTA CON IMÁGENES	BOTONES		

Flickr Andres
Este proyecto ha sido creado con Angular y Spring, para las asignaturas de Acceso a Datos, Programación Web y Desarrollo de interfaces, por Andrés Sánchez Vera

Links
Home
CRUD

© 2022 Andrés Sanchez Productions

Este es el listado del CRUD

flickr

Home CRUD Cerrar Sesión

Título
El título no puede estar vacío

Descripción (URL)
El enlace debe tener un formato correcto

ENVIAR ➤ VOLVER ←

Agregar del CRUD

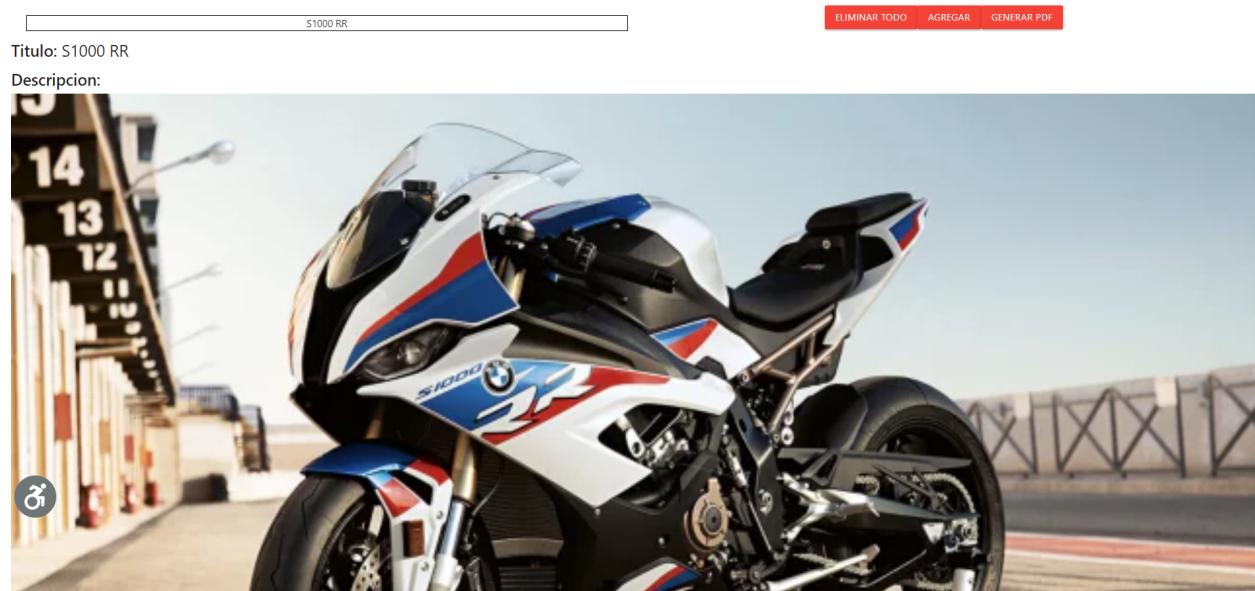
flickr

Home CRUD Cerrar Sesión

Tutorial enviado correctamente!

AGREGAR OTRO
VOLVER A LAS LISTAS

Cuando se agrega, sale este mensaje



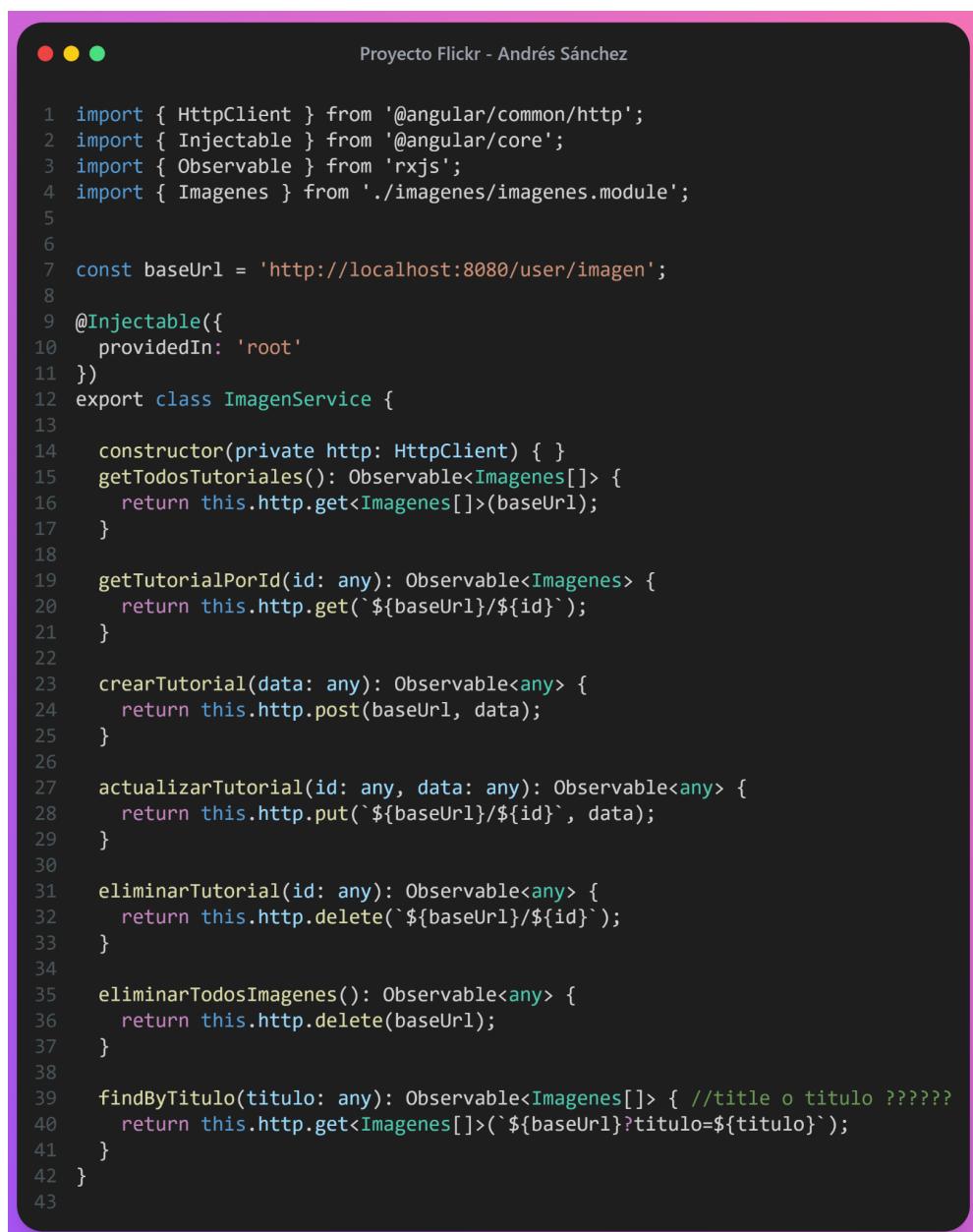
Así se ven los detalles

Detalles de la imagen

Título	S1000 RR
Descripción	https://riders.dreamag.com/wp-content/uploads/default/0001/99/RR-1218-medium.webp
<input type="button" value="ELIMINAR"/> <input type="button" value="ACTUALIZAR"/> <input type="button" value="VOLVER"/>	

Y así la edición

4.4. Conexión Front - Back

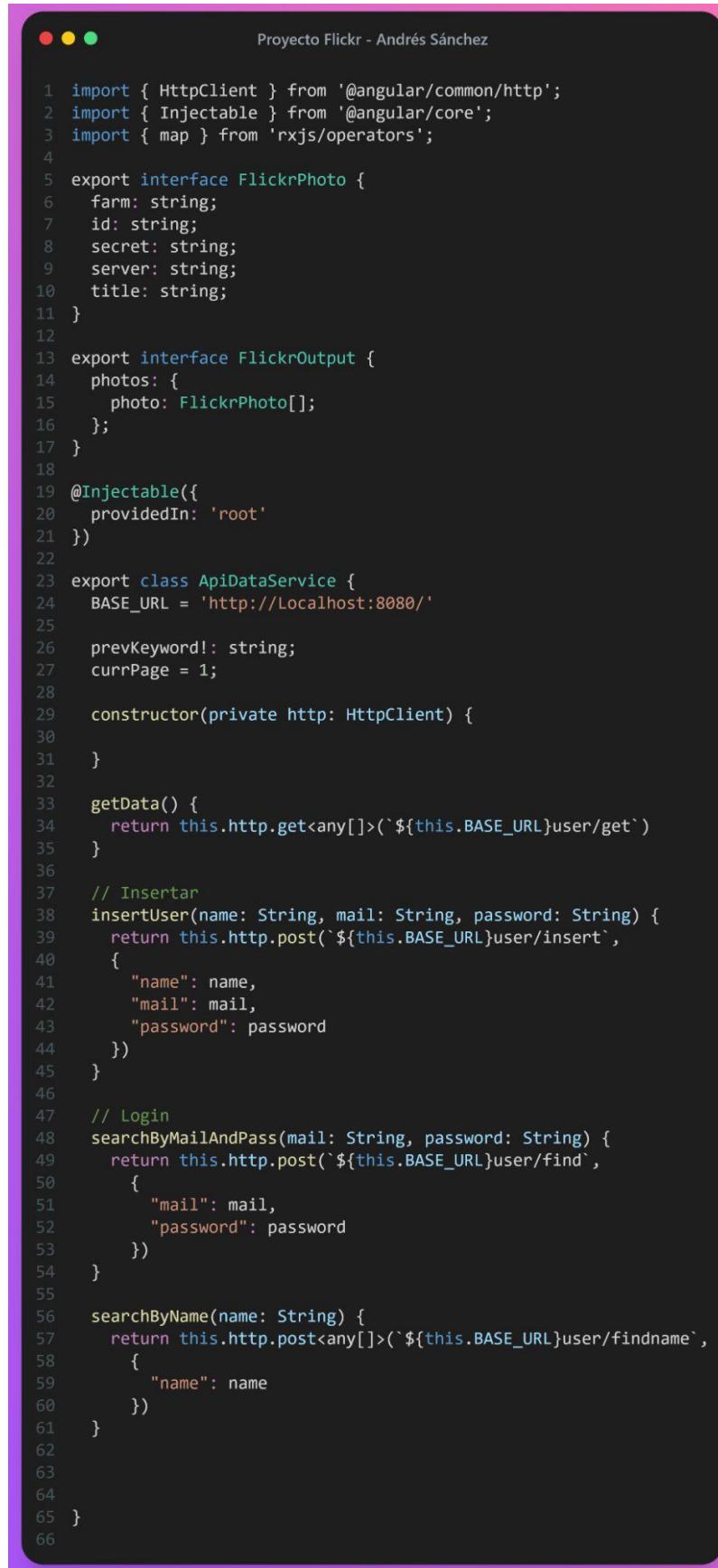


```

 1 import { HttpClient } from '@angular/common/http';
 2 import { Injectable } from '@angular/core';
 3 import { Observable } from 'rxjs';
 4 import { Imagenes } from './imagenes/imagenes.module';
 5
 6
 7 const baseUrl = 'http://localhost:8080/user/imagen';
 8
 9 @Injectable({
10   providedIn: 'root'
11 })
12 export class ImagenService {
13
14   constructor(private http: HttpClient) { }
15   getTodosTutorial(): Observable<Imagenes[]> {
16     return this.http.get<Imagenes[]>(baseUrl);
17   }
18
19   getTutorialPorId(id: any): Observable<Imagenes> {
20     return this.http.get(`${baseUrl}/${id}`);
21   }
22
23   crearTutorial(data: any): Observable<any> {
24     return this.http.post(baseUrl, data);
25   }
26
27   actualizarTutorial(id: any, data: any): Observable<any> {
28     return this.http.put(`${baseUrl}/${id}`, data);
29   }
30
31   eliminarTutorial(id: any): Observable<any> {
32     return this.http.delete(`${baseUrl}/${id}`);
33   }
34
35   eliminarTodosImagenes(): Observable<any> {
36     return this.http.delete(baseUrl);
37   }
38
39   findByTitulo(titulo: any): Observable<Imagenes[]> { //title o titulo ??????
40     return this.http.get<Imagenes[]>(`${baseUrl}?titulo=${titulo}`);
41   }
42 }
43

```

Conexión de la parte del CRUD a nuestro front



```

 1 import { HttpClient } from '@angular/common/http';
 2 import { Injectable } from '@angular/core';
 3 import { map } from 'rxjs/operators';
 4
 5 export interface FlickrPhoto {
 6   farm: string;
 7   id: string;
 8   secret: string;
 9   server: string;
10   title: string;
11 }
12
13 export interface FlickrOutput {
14   photos: {
15     photo: FlickrPhoto[];
16   };
17 }
18
19 @Injectable({
20   providedIn: 'root'
21 })
22
23 export class ApiService {
24   BASE_URL = 'http://localhost:8080/';
25
26   prevKeyword!: string;
27   currPage = 1;
28
29   constructor(private http: HttpClient) {
30   }
31
32
33   getData() {
34     return this.http.get<any[]>(`${this.BASE_URL}user/get`)
35   }
36
37   // Insertar
38   insertUser(name: String, mail: String, password: String) {
39     return this.http.post(` ${this.BASE_URL}user/insert` ,
40     {
41       "name": name,
42       "mail": mail,
43       "password": password
44     })
45   }
46
47   // Login
48   searchByMailAndPass(mail: String, password: String) {
49     return this.http.post(` ${this.BASE_URL}user/find` ,
50     {
51       "mail": mail,
52       "password": password
53     })
54   }
55
56   searchByName(name: String) {
57     return this.http.post<any[]>(` ${this.BASE_URL}user/findname` ,
58     {
59       "name": name
60     })
61   }
62
63
64
65 }
66

```

Y la conexión con nuestra parte del login

4.5. API Flickr



```

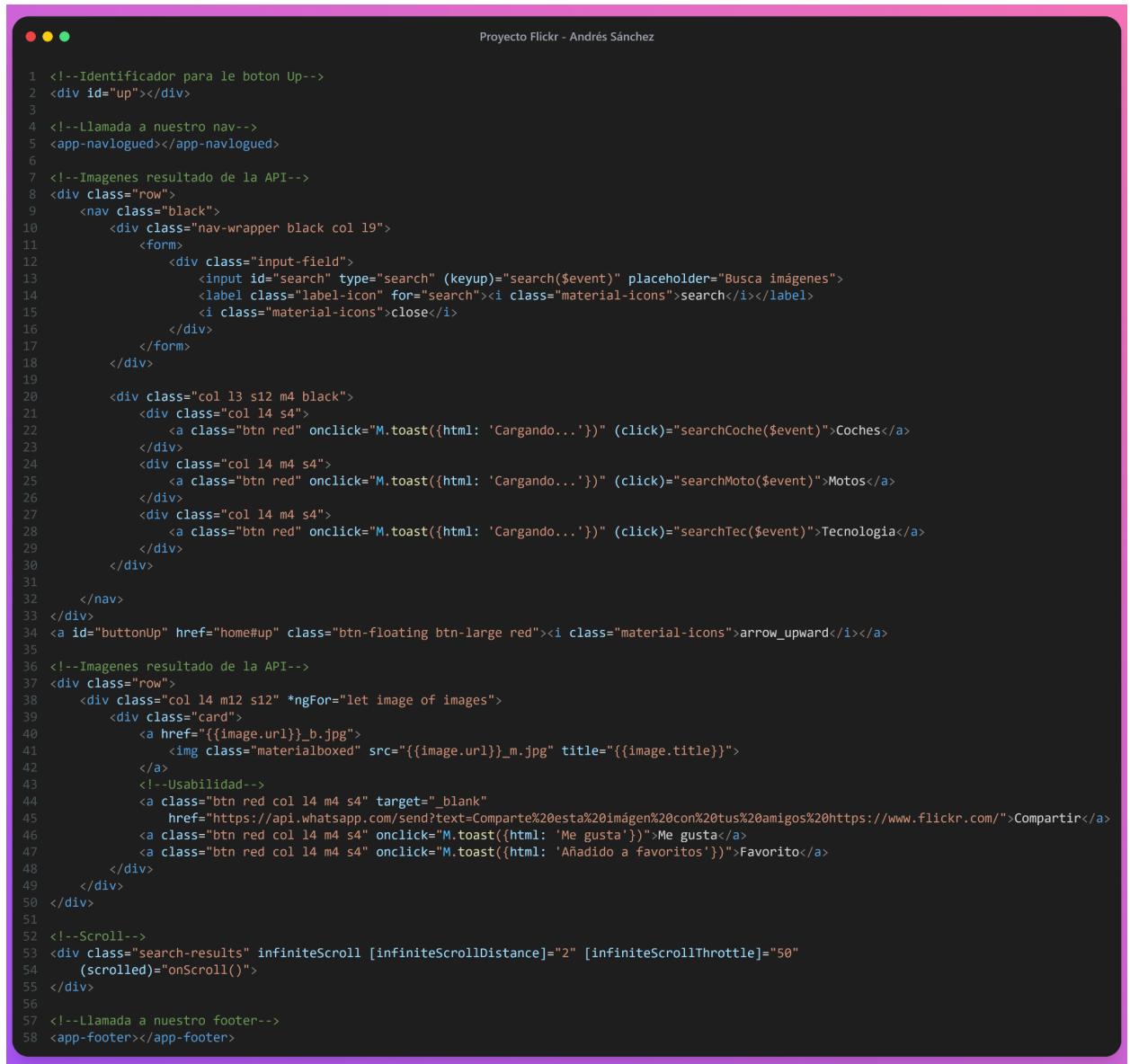
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { map } from 'rxjs/operators';
4 import { environment } from 'src/environments/environment';
5
6 export interface FlickrPhoto {
7   farm: string;
8   id: string;
9   secret: string;
10  server: string;
11  title: string;
12 }
13
14 export interface FlickrOutput {
15   photos: {
16     photo: FlickrPhoto[];
17   };
18 }
19
20 @Injectable({
21   providedIn: 'root'
22 })
23
24 export class ApiService {
25   UrlFli = 'https://www.flickr.com/services/rest/?method=flickr.photos.search&';
26
27   prevKeyword!: string;
28   currPage = 1;
29
30   constructor(private http: HttpClient) {
31   }
32 }
33
34 // Flickr metodos
35 search_keyword(keyword: string) {
36   if (this.prevKeyword === keyword) {
37     this.currPage++;
38   } else {
39     this.currPage = 1;
40   }
41   this.prevKeyword = keyword;
42
43   const params = `api_key=${environment.flickr.key}&text=${keyword}&format=json&nojsoncallback=1&per_page=12&page=${this.currPage}`;
44
45   return this.http.get( this.UrlFli + params).pipe(map((res: any) => {
46     const urlArr: any[] = [];
47     res.photos.photo.forEach((ph: FlickrPhoto) => {
48       const photoObj = {
49         url: `https://farm${ph.farm}.staticflickr.com/${ph.server}/${ph.id}_${ph.secret}`,
50         title: ph.title
51       };
52       urlArr.push(photoObj);
53     });
54     return urlArr;
55   }));
56 }
57
58 }
59 }
60 
```

api-data.service.ts

Aquí conectamos nuestra API, aunque en environments, environments.ts, debemos agregar la clave de nuestra API:

```
flickr: {
  key:"c954100e660d0a64292609de9blefac9"
}
```

Clave que generamos desde la [Página oficial Flickr API](#), registrándose y creando una aplicación.



The screenshot shows a browser window with the title "Proyecto Flickr - Andrés Sánchez". The content of the page is the source code of the "home.component.html" file, which is a template for an Angular application. The code includes HTML structural elements like divs and forms, as well as Angular-specific directives such as *ngFor and infiniteScroll. It also contains CSS classes and some JavaScript-like logic for button behaviors and image links.

```

1 <!--Identificador para el botón Up-->
2 <div id="up"></div>
3
4 <!--Llamada a nuestro nav-->
5 <app-navlogged></app-navlogged>
6
7 <!--Imagenes resultado de la API-->
8 <div class="row">
9   <nav class="black">
10    <div class="nav-wrapper black col l9">
11      <form>
12        <div class="input-field">
13          <input id="search" type="search" (keyup)="search($event)" placeholder="Busca imágenes">
14          <label class="label-icon" for="search"><i class="material-icons">search</i></label>
15          <i class="material-icons">close</i>
16        </div>
17      </form>
18    </div>
19
20    <div class="col l3 s12 m4 black">
21      <div class="col l4 s4">
22        <a class="btn red" onclick="M.toast({html: 'Cargando...'})" (click)="searchCoche($event)">Coches</a>
23      </div>
24      <div class="col l4 m4 s4">
25        <a class="btn red" onclick="M.toast({html: 'Cargando...'})" (click)="searchMoto($event)">Motos</a>
26      </div>
27      <div class="col l4 m4 s4">
28        <a class="btn red" onclick="M.toast({html: 'Cargando...'})" (click)="searchTec($event)">Tecnología</a>
29      </div>
30    </div>
31
32  </nav>
33 </div>
34 <a id="buttonUp" href="#home" class="btn-floating btn-large red"><i class="material-icons">arrow_upward</i></a>
35
36 <!--Imagenes resultado de la API-->
37 <div class="row">
38   <div class="col l4 m12 s12" *ngFor="let image of images">
39     <div class="card">
40       <a href="{{image.url}}_b.jpg">
41         
42       </a>
43       <!--Usabilidad-->
44       <a class="btn red col l4 m4 s4" target="_blank"
45          href="https://api.whatsapp.com/send?text=Comparte%20esta%20imagen%20con%20tus%20amigos%20https://www.flickr.com/">Compartir</a>
46       <a class="btn red col l4 m4 s4" onclick="M.toast({html: 'Me gusta'})">Me gusta</a>
47       <a class="btn red col l4 m4 s4" onclick="M.toast({html: 'Añadido a favoritos'})">Favorito</a>
48     </div>
49   </div>
50 </div>
51
52 <!--Scroll-->
53 <div class="search-results" infiniteScroll [infiniteScrollDistance]="2" [infiniteScrollThrottle]="50"
54   (scrolled)="onScroll()">
55 </div>
56
57 <!--Llamada a nuestro footer-->
58 <app-footer></app-footer>

```

home.component.html

4.6. Login y Registrar

```

1 <!--Llamada a nuestro nav-->
2 <app-nav></app-nav>
3
4 <h1 id="tituloReg" class="center">Iniciar Sesión</h1>
5
6 <!--Login-->
7 <div id="marginTop" class="row">
8   <div class="offset-l1 col l5 s12">
9     
10  </div>
11  <form class="col l5 offset-l1 s12">
12    <div class="row">
13      <div class="input-field col s12 l8">
14        <input id="email" type="email" #email2 required class="validate">
15        <label for="email">Email</label>
16      </div>
17    </div>
18    <div class="row">
19      <div class="input-field col s12 l8">
20        <input id="password" type="password" #password2 class="validate">
21        <label for="password">Contraseña</label>
22      </div>
23    </div>
24    <a id="button" (click)="searchByNameAndPassword(email2.value, password2.value)" class="btn modal-trigger red"><i
25      class="material-icons right">send</i>Iniciar Sesión</a> <br><br>
26    <span>¿Aun no tienes cuenta? <a id="enlaceLogin" routerLink="/signin"> Create una ahora</a></span>
27  </form>
28 </div>

```

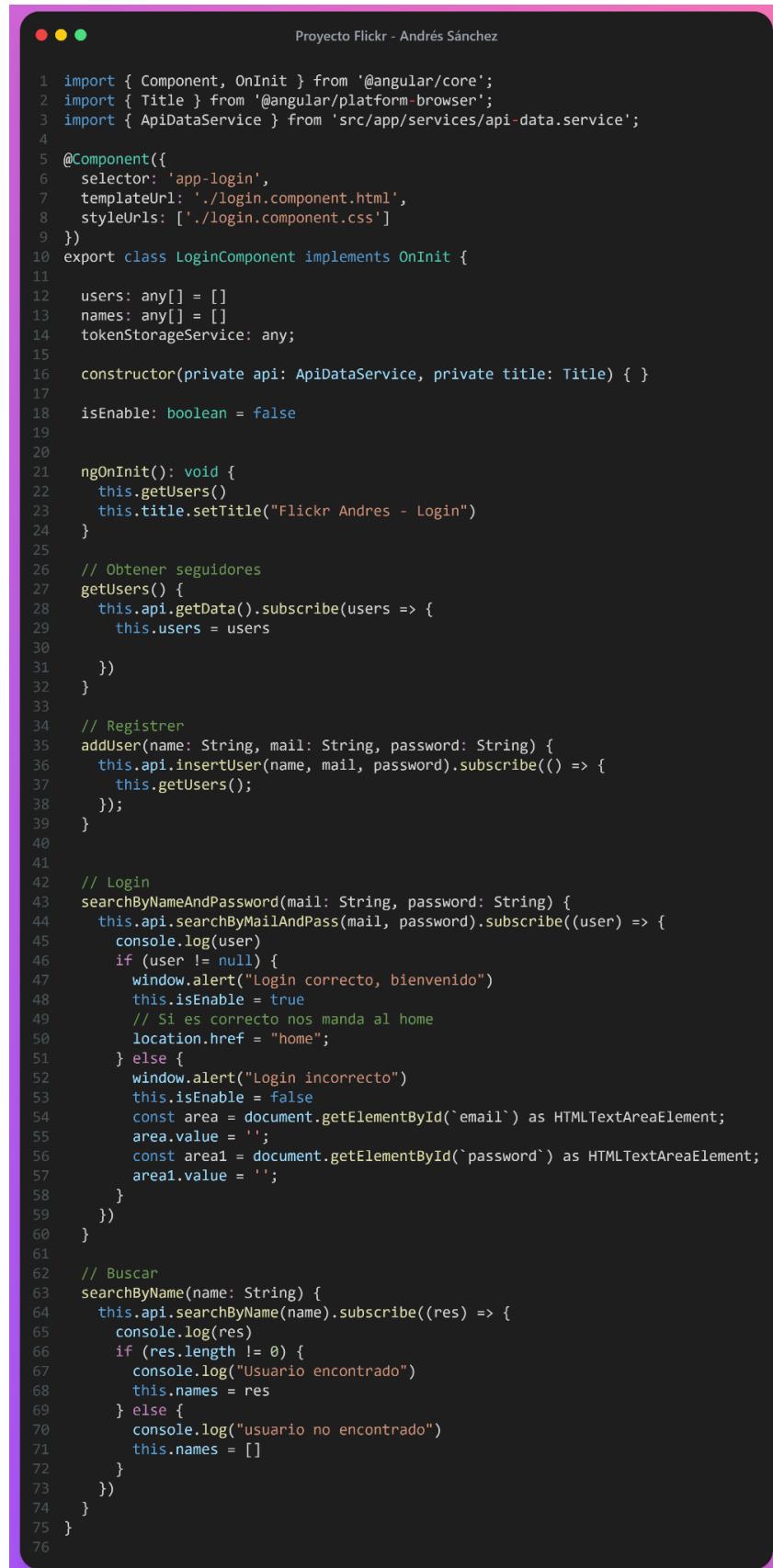
login.component.html

```

1 <!--Llamada a nuestro nav-->
2 <app-nav></app-nav>
3
4 <h1 id="tituloReg" class="center">Iniciar Sesión</h1>
5
6 <!--Login-->
7 <div id="marginTop" class="row">
8   <div class="offset-l1 col l5 s12">
9     
10  </div>
11  <form class="col l5 offset-l1 s12">
12    <div class="row">
13      <div class="input-field col s12 l8">
14        <input id="email" type="email" #email2 required class="validate">
15        <label for="email">Email</label>
16      </div>
17    </div>
18    <div class="row">
19      <div class="input-field col s12 l8">
20        <input id="password" type="password" #password2 class="validate">
21        <label for="password">Contraseña</label>
22      </div>
23    </div>
24    <a id="button" (click)="searchByNameAndPassword(email2.value, password2.value)" class="btn modal-trigger red"><i
25      class="material-icons right">send</i>Iniciar Sesión</a> <br><br>
26    <span>¿Aun no tienes cuenta? <a id="enlaceLogin" routerLink="/signin"> Create una ahora</a></span>
27  </form>
28 </div>

```

register.component.html



```

1 import { Component, OnInit } from '@angular/core';
2 import { Title } from '@angular/platform-browser';
3 import { ApiService } from 'src/app/services/api-data.service';
4
5 @Component({
6   selector: 'app-login',
7   templateUrl: './login.component.html',
8   styleUrls: ['./login.component.css']
9 })
10 export class LoginComponent implements OnInit {
11
12   users: any[] = []
13   names: any[] = []
14   tokenStorageService: any;
15
16   constructor(private api: ApiService, private title: Title) { }
17
18   isEnabled: boolean = false
19
20
21   ngOnInit(): void {
22     this.getUsers()
23     this.title.setTitle("Flickr Andres - Login")
24   }
25
26   // Obtener seguidores
27   getUsers() {
28     this.api.getData().subscribe(users => {
29       this.users = users
30
31     })
32   }
33
34   // Registrar
35   addUser(name: String, mail: String, password: String) {
36     this.api.insertUser(name, mail, password).subscribe(() => {
37       this.getUsers();
38     });
39   }
40
41
42   // Login
43   searchByNameAndPassword(mail: String, password: String) {
44     this.api.searchByMailAndPass(mail, password).subscribe((user) => {
45       console.log(user)
46       if (user != null) {
47         window.alert("Login correcto, bienvenido")
48         this.isEnabled = true
49         // Si es correcto nos manda al home
50         location.href = "home";
51       } else {
52         window.alert("Login incorrecto")
53         this.isEnabled = false
54         const area = document.getElementById(`email`) as HTMLTextAreaElement;
55         area.value = '';
56         const area1 = document.getElementById(`password`) as HTMLTextAreaElement;
57         area1.value = '';
58       }
59     })
60   }
61
62   // Buscar
63   searchByName(name: String) {
64     this.api.searchByName(name).subscribe((res) => {
65       console.log(res)
66       if (res.length != 0) {
67         console.log("Usuario encontrado")
68         this.names = res
69       } else {
70         console.log("usuario no encontrado")
71         this.names = []
72       }
73     })
74   }
75 }
76 
```

login.component.ts

```

1 import { Component, OnInit } from '@angular/core';
2 import { Title } from '@angular/platform-browser';
3 import { ApiService } from 'src/app/services/api-data.service';
4
5 @Component({
6   selector: 'app-registrar',
7   templateUrl: './registrar.component.html',
8   styleUrls: ['./registrar.component.css']
9 })
10 export class RegistrarComponent implements OnInit {
11
12   users: any[] = []
13   names: any[] = []
14
15   constructor(private api: ApiService, private title: Title) { }
16
17   isEnabledbutton: boolean = false
18   isEnabled: boolean = false
19   isEnabled: boolean = false
20   isEnableddd: boolean = false
21
22
23   ngOnInit(): void {
24     this.title.setTitle("Flickr Andres - Registrar")
25   }
26
27   // Añadir usuario
28   addUser(name: String, mail: String, password: String) {
29     if (name != null && mail != null && password != null) {
30       window.alert("Registro correcto")
31       this.api.insertUser(name, mail, password).subscribe();
32       const area = document.getElementById(`name`) as HTMLTextAreaElement;
33       area.value = '';
34       const area1 = document.getElementById(`email`) as HTMLTextAreaElement;
35       area1.value = '';
36       const area2 = document.getElementById(`password`) as HTMLTextAreaElement;
37       area2.value = '';
38     } else {
39       window.alert("Registro incorrecto porfavor vuelva a intentarlo")
40     }
41   }
42
43   // Método para habilitar y deshabilitar el botón
44   comprobar(name: String, mail: String, password: String) {
45     if (name.length > 5 && mail.includes("@") && password.length > 7) {
46       this.isEnabledbutton = true
47     } else {
48       this.isEnabledbutton = false
49     }
50   }
51
52   // Método para comprobar que el nombre sea mayor de 5
53   comprobarname(name: String) {
54     if (name.length > 5) {
55       this.isEnabled = true
56     } else {
57       this.isEnabled = false
58     }
59   }
60
61   // Método para comprobar que el mail tenga un formato correcto
62   comprobarmail(mail: String) {
63     if (mail.includes("@")) {
64       this.isEnabled = true
65     } else {
66       this.isEnabled = false
67     }
68   }
69
70   // Método para comprobar que la contraseña sea mayor que 7
71   comprobarpass(password: String) {
72     if (password.length > 7) {
73       this.isEnableddd = true
74     } else {
75       this.isEnableddd = false
76     }
77   }
78 }

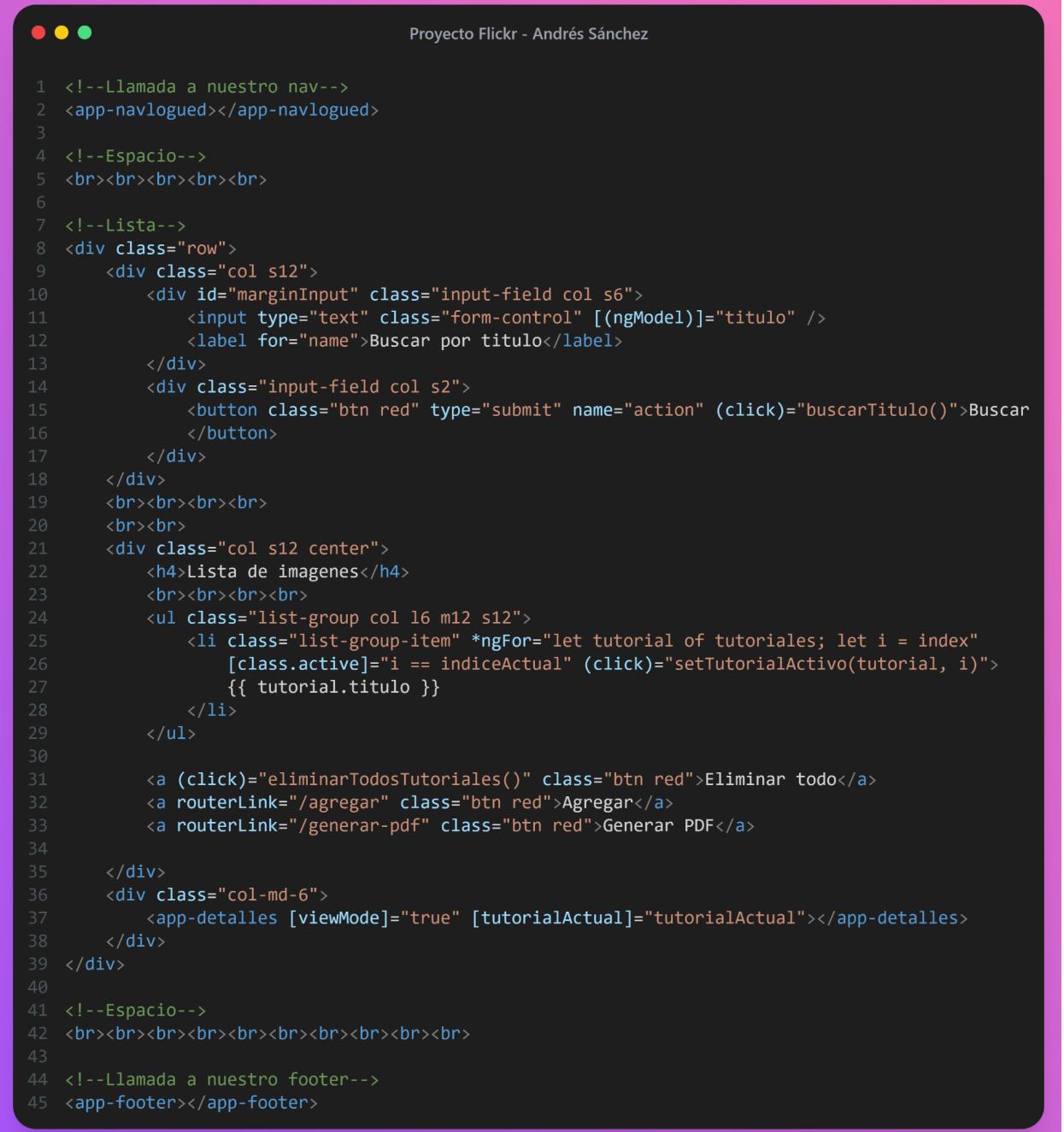
```

register.component.ts

4.7. CRUD Imágenes

agregar.component.html

detalles.component.html



```

1 <!--Llamada a nuestro nav-->
2 <app-navlogged></app-navlogged>
3
4 <!--Espacio-->
5 <br><br><br><br><br>
6
7 <!--Lista-->
8 <div class="row">
9     <div class="col s12">
10        <div id="marginInput" class="input-field col s6">
11            <input type="text" class="form-control" [(ngModel)]="titulo" />
12            <label for="name">Buscar por titulo</label>
13        </div>
14        <div class="input-field col s2">
15            <button class="btn red" type="submit" name="action" (click)="buscarTitulo()">Buscar
16            </button>
17        </div>
18    </div>
19    <br><br><br><br>
20    <br><br>
21    <div class="col s12 center">
22        <h4>Lista de imagenes</h4>
23        <br><br><br><br>
24        <ul class="list-group col 16 m12 s12">
25            <li class="list-group-item" *ngFor="let tutorial of tutoriales; let i = index"
26                [class.active]="i == indiceActual" (click)="setTutorialActivo(tutorial, i)">
27                {{ tutorial.titulo }}
28            </li>
29        </ul>
30
31        <a (click)="eliminarTodosTutorial()" class="btn red">Eliminar todo</a>
32        <a routerLink="/agregar" class="btn red">Agregar</a>
33        <a routerLink="/generar-pdf" class="btn red">Generar PDF</a>
34
35    </div>
36    <div class="col-md-6">
37        <app-detalles [viewMode]="true" [tutorialActual]="tutorialActual"></app-detalles>
38    </div>
39 </div>
40
41 <!--Espacio-->
42 <br><br><br><br><br><br><br><br><br><br>
43
44 <!--Llamada a nuestro footer-->
45 <app-footer></app-footer>

```

4.8. Generación de PDF (jsPDF)

Para usar la librería jsPDF, deberemos instalarla, con el comando `npm install jspdf --save` en nuestra terminal, para usarlo en el archivo TypeScript, deberemos hacer las importaciones:

```

import jsPDF from 'jspdf';
import html2canvas from 'html2canvas';

```

```

1  public openPDF(): void {
2      // Seleccionamos la informacion que va a coger, mediante el ID
3      let DATA: any = document.getElementById('htmlData');
4      html2canvas(DATA).then((canvas) => {
5          // Seleccionamos la altura y anchura
6          let fileWidth = 208;
7          let fileHeight = (canvas.height * fileWidth) / canvas.width;
8          const FILEURI = canvas.toDataURL('image/png');
9          let PDF = new jsPDF('p', 'mm', 'a4');
10         let position = 0;
11         // Añadimos una imagen
12         PDF.addImage(FILEURI, 'PNG', 0, position, fileWidth, fileHeight);
13         // Asignamos el tamaño de fuente
14         PDF.setFontSize(20)
15         // Asignamos un texto
16         PDF.text('Este proyecto ha sido realizado con\nSpring y Angular, por Andrés\nSánchez Vera', 50, 130)
17         var img = new Image()
18         img.src = 'assets/flickrbysanchezblack.png'
19         PDF.addImage(img, 'png', 10, 200, 180, 90)
20         // Asignamos el metodo para guardar, y el nombre del documento
21         PDF.save('informe-imagenes-andres-sanchez.pdf');
22     });
23 }
24 }
```

generarpdf.component.ts

```

1  <!--Llamada a nuestro nav-->
2  <app-navlogued></app-navlogued>
3
4  <!--Espacio en blanco-->
5  <br><br><br><br>
6
7  <!--Tabla que se va a imprimir con el id htmlData-->
8  <div id="htmlData" class="row">
9      <div class="offset-12 col 19 s12">
10         <tr class="table-primary">
11             <th>Titulo</th>
12             <th>Enlace</th>
13             <th><a (click)="openPDF()" class="btn red">Generar PDF</a></th>
14             <th><a routerLink="/lista" class="btn red">Volver a listas</a> </th>
15         </tr>
16     </div>
17     <div class="offset-12 col 19 s12">
18         <tr *ngFor="let tutorial of tutoriales">
19             <td>{{tutorial.titulo}}</td>
20             <td>{{tutorial.descripcion}}</td>
21         </tr>
22     </div>
23 </div>
```

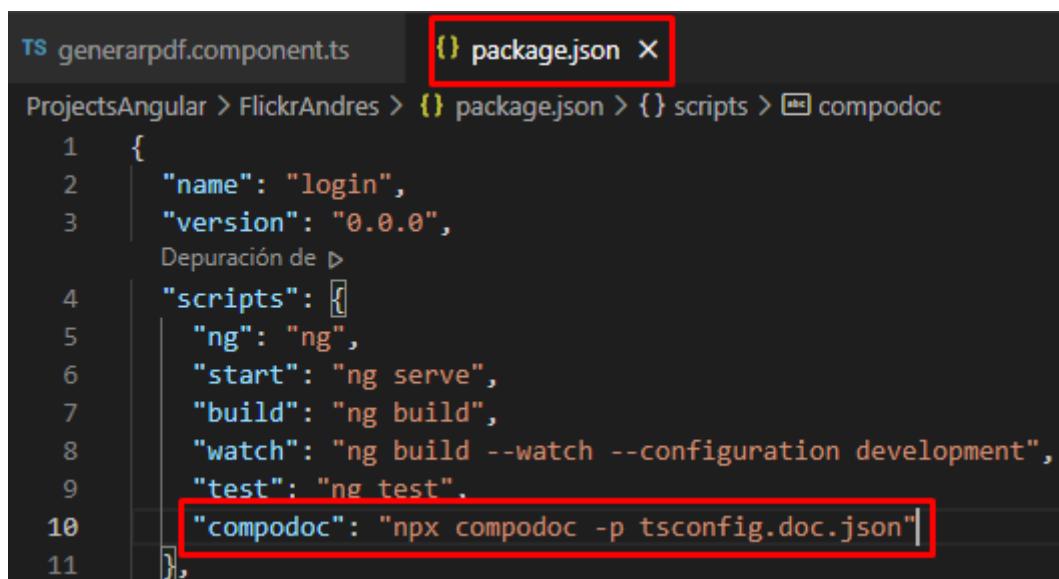
generarpdf.component.html

4.9. Compodoc

Para generar el Compodoc, primero debemos instalar la librería, con npm install -g @compodoc/compodoc

```
PS C:\Users\andre\Desktop\ProjectsAngular\flickrandres> npm install -g @compodoc/compodoc
>>
[ ] | idealTree:@babel/preset-env: sill fetch manifest xmldoc@^1.1.2
```

Dentro del package.json, ponemos la línea compodoc, para que nos genere el documento



```
TS generarpdf.component.ts          {} package.json X
ProjectsAngular > FlickrAndres > {} package.json > {} scripts > compodoc
1  {
2    "name": "login",
3    "version": "0.0.0",
4    "scripts": [
5      "ng": "ng",
6      "start": "ng serve",
7      "build": "ng build",
8      "watch": "ng build --watch --configuration development",
9      "test": "ng test",
10     "compodoc": "npx compodoc -p tsconfig.doc.json"
11   ],
12 }
```

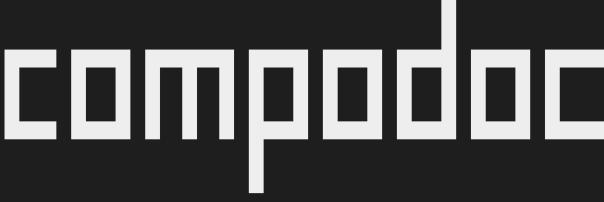
Crearemos a la altura del proyecto, un documento: tsconfig.doc.json, con la ubicación donde se guardará la documentación y la key

```
{
  "include": ["src/**/*.ts"],
  "exclude": ["src/test.ts", "src/**/*.spec.ts", "src/app/file-to-exclude.ts"]
}
```

Una vez hecho esto, lo ejecutaremos, con npm run compodoc,

```
found 0 vulnerabilities
PS C:\Users\andre\Desktop\ProjectsAngular\flickrandres> npm run compodoc

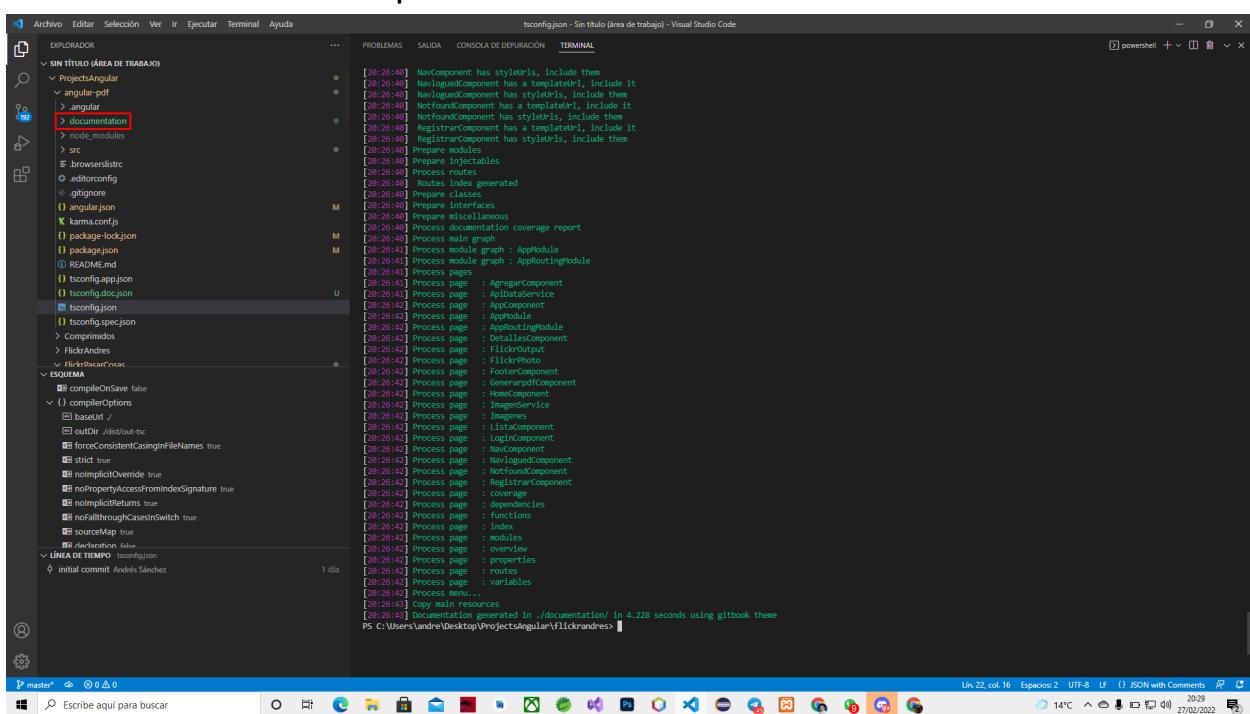
> login@0.0.0 compodoc
> npx compodoc -p tsconfig.doc.json

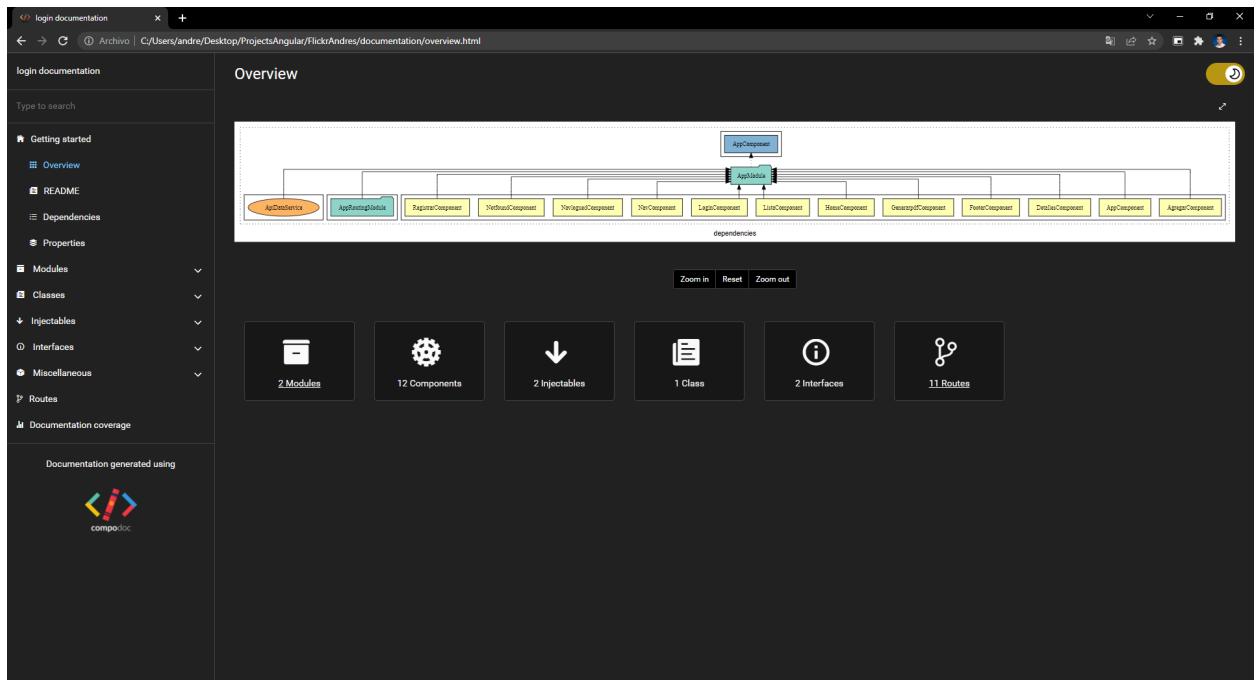

1.1.19

TypeScript version used by CompoDoc : 4.5.5
TypeScript version of current project : 4.3.5
Node.js version : v16.13.1
Operating system : Windows 10

[20:26:39] No configuration file found, switching to CLI flags.
[20:26:39] Using tsconfig file : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\tsconfig.doc.json
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\main.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\polyfills.ts
[20:26:40] Excluding : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\test.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\app-routing.module.ts
[20:26:40] Ignoring : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\app.component.spec.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\app.module.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\app.component.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\environments\environment.prod.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\environments\environment.ts
[20:26:40] Ignoring : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\imagenes\imagen.service.spec.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\imagenes\imagen.service.ts
[20:26:40] Ignoring : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\services\api-data.service.spec.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\services\api-data.service.ts
[20:26:40] Ignoring : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\nav\nav.component.spec.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\nav\nav.component.ts
[20:26:40] Ignoring : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\home\home.component.spec.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\home\home.component.ts
[20:26:40] Ignoring : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\login\login.component.spec.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\login\login.component.ts
[20:26:40] Ignoring : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\footer\footer.component.spec.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\footer\footer.component.ts
[20:26:40] Ignoring : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\navlogued\navlogued.component.spec.ts
[20:26:40] Including : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\navlogued\navlogued.component.ts
[20:26:40] Ignoring : C:\Users\andre\Desktop\ProjectsAngular\flickrandres\src\app\components\registrar\registrar.component.spec.ts
```

Ya tenemos nuestra carpeta con la documentación

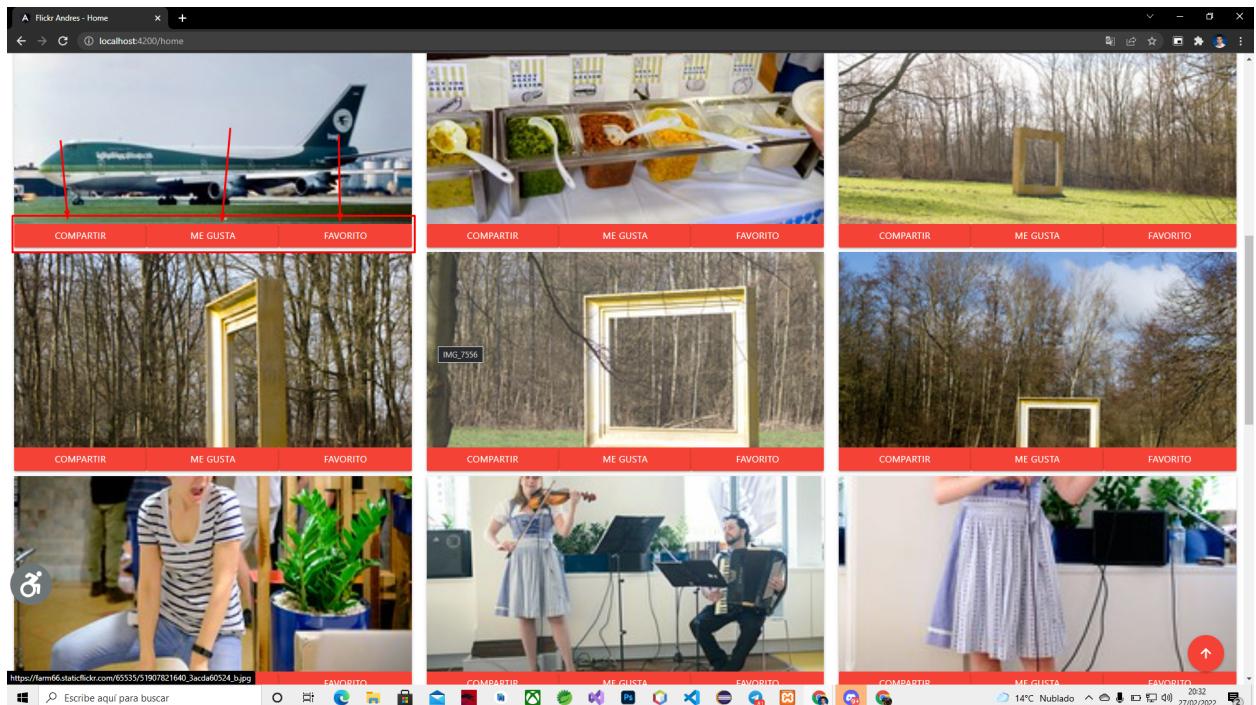




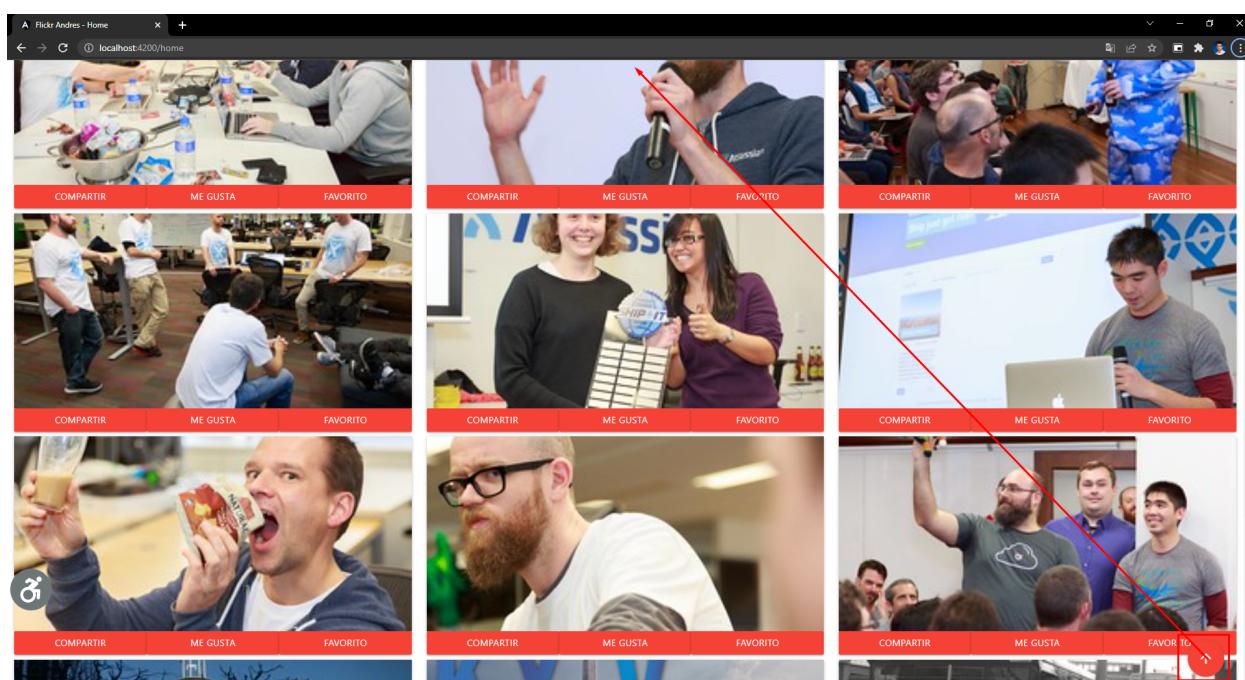
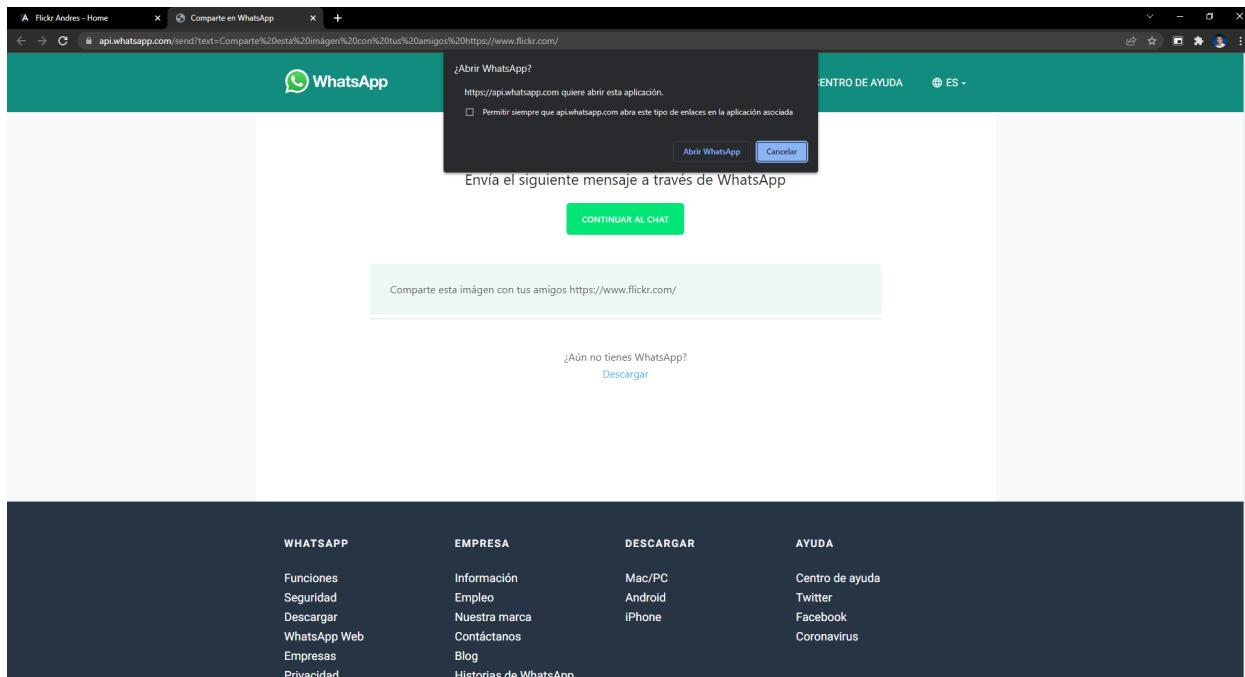
Toda esta documentación estará en el proyecto!

4.10. Usabilidad UX

Para crear usabilidad en nuestra aplicación web, he creado 3 botones donde se encuentra nuestra API de Flickr



2 de ellos tienen un Toast, Añadido a favoritos y Me gusta, mientras el botón de compartir, te comparte un texto y un enlace mediante WhatsApp



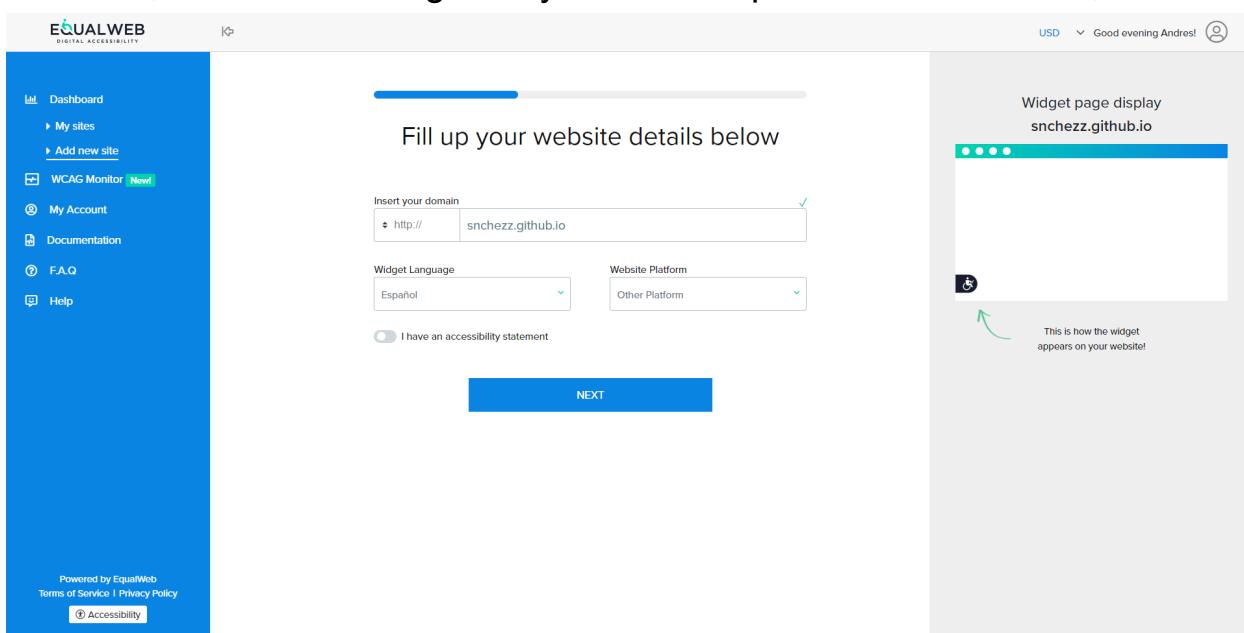
También tenemos un botón para subir arriba de la web, al ser un Scroll infinito, deberemos colocar este botón para que el usuario no deba usar el scroll continuamente si quiere subir.

También tenemos un efecto en los botones, para que agrande al estar encima de ellos, y en toda la parte del CRUD, se puede volver atrás mediante botones y está con comprobaciones.

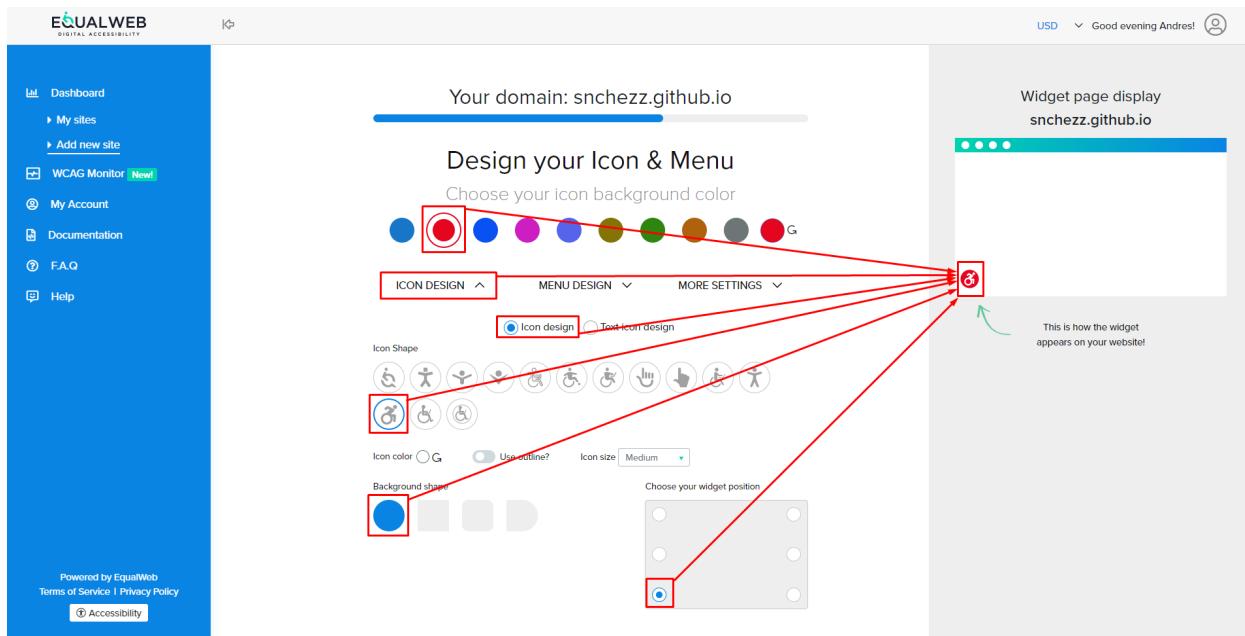
4.11. Accesibilidad

Para agregar todas las funcionalidades de Accesibilidad a nuestra web, iremos a [EqualWeb](#), mismo desarrollador que ha implementado la accesibilidad de webs como Oysho, McDonald's, Pizza Hut...

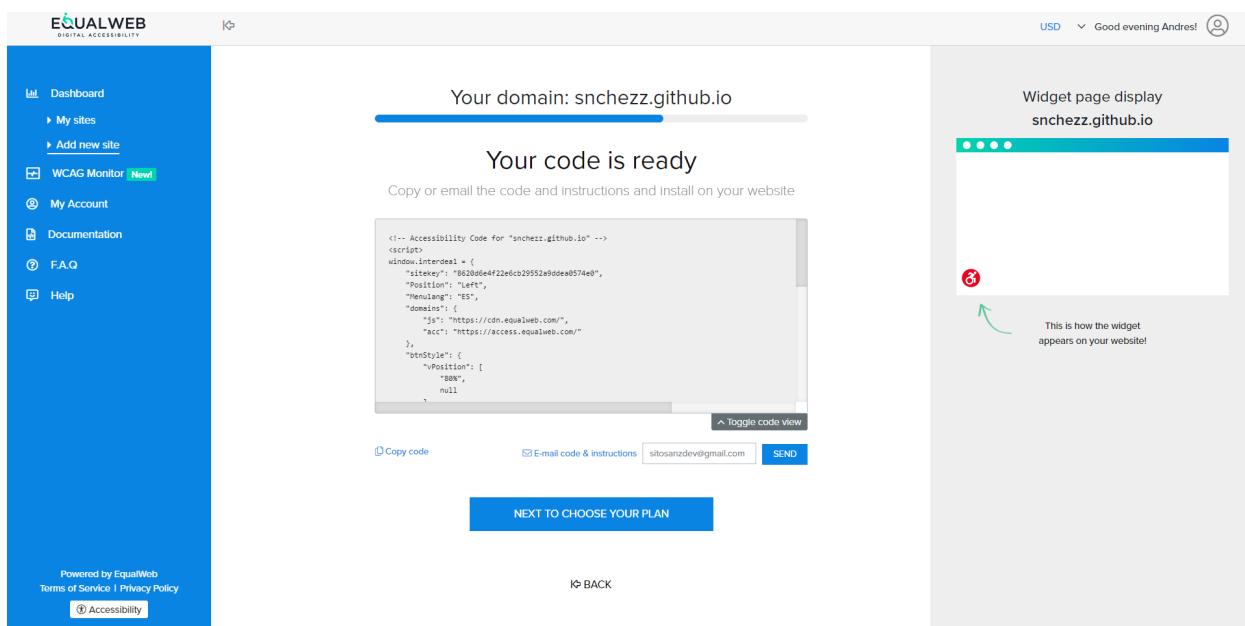
Para ello, nos vamos a registrar y nos saldrá para añadir una web,



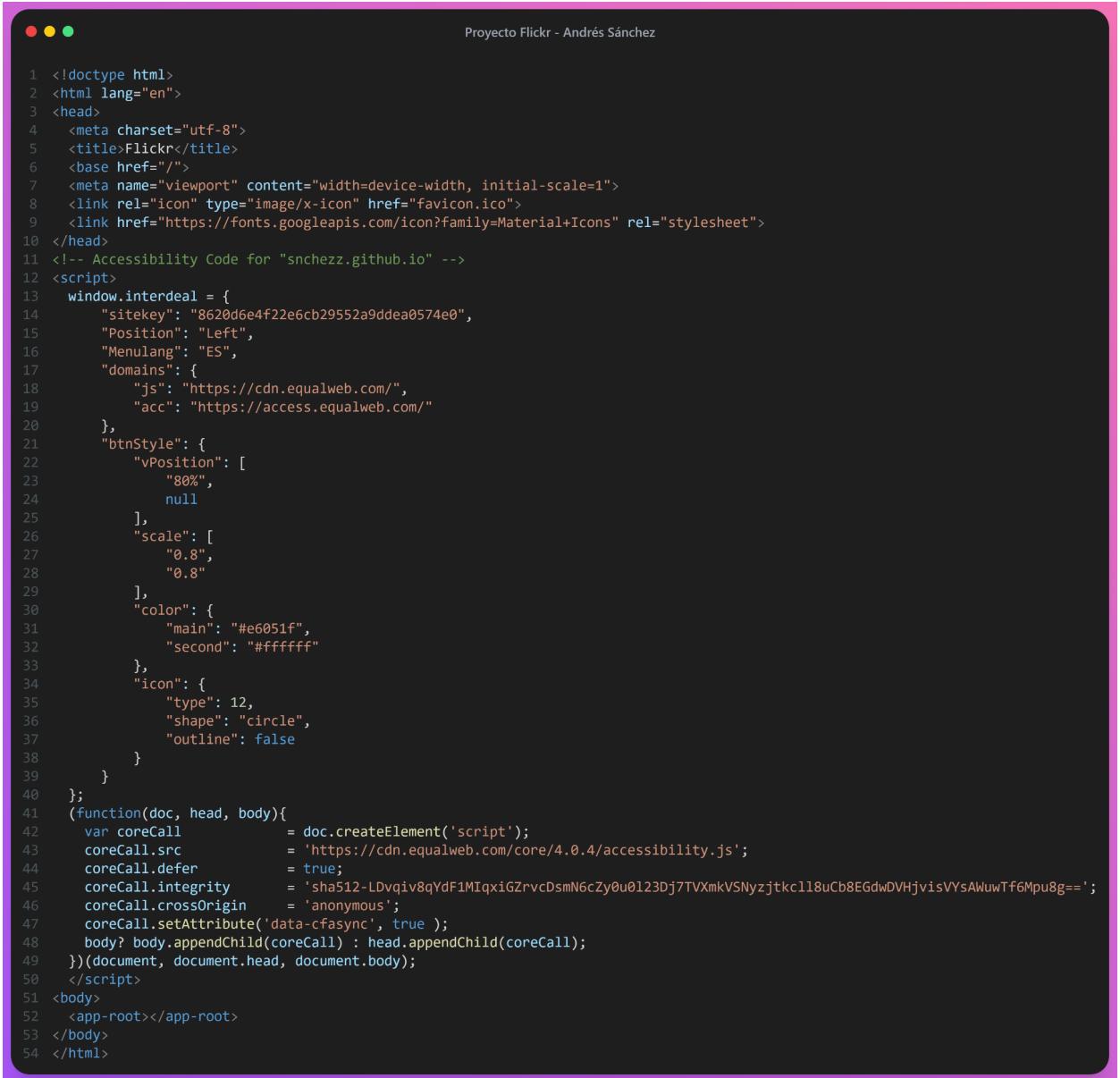
Ponemos el enlace (que no será utilizado como tal, pero nos sirve para que nos proporcione la información necesaria)



Seleccionamos como queremos que sea el ícono, color, forma, posición y pulsamos next.



Para agregar esta funcionalidad, deberemos añadir el código en nuestro index.html



```

1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>Flickr</title>
6   <base href="/">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <link rel="icon" type="image/x-icon" href="favicon.ico">
9   <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
10 </head>
11 <!-- Accessibility Code for "snchez.github.io" -->
12 <script>
13   window.interdeal = {
14     "sitekey": "8620d6e4f22e6cb29552a9dde0574e0",
15     "Position": "Left",
16     "Menulang": "ES",
17     "domains": {
18       "js": "https://cdn.equalweb.com/",
19       "acc": "https://access.equalweb.com/"
20     },
21     "btnStyle": {
22       "vPosition": [
23         "80%",
24         null
25       ],
26       "scale": [
27         "0.8",
28         "0.8"
29     ],
30       "color": {
31         "main": "#e6051f",
32         "second": "#ffffff"
33     },
34       "icon": {
35         "type": 12,
36         "shape": "circle",
37         "outline": false
38     }
39   };
40   (function(doc, head, body){
41     var coreCall      = doc.createElement('script');
42     coreCall.src      = 'https://cdn.equalweb.com/core/4.0.4/accessibility.js';
43     coreCall.defer    = true;
44     coreCall.integrity = 'sha512-LDvqiv8qYdF1MIqxiGZrvCDsmN6cZy0u0123Dj7TVXmkVSNyzjtkcll8uCb8EGdwDVHjvisVYsAWuwTf6MpU8g==';
45     coreCall.crossOrigin = 'anonymous';
46     coreCall.setAttribute('data-cfasync', true );
47     body? body.appendChild(coreCall) : head.appendChild(coreCall);
48   })(document, document.head, document.body);
49   </script>
50 </body>
51 <app-root></app-root>
52 </body>
53 </html>

```

Ya en nuestra web nos saldrá el ícono:

A screenshot of the Flickr login page. At the top, there's a black header bar with the Flickr logo on the left and 'Iniciar Sesión' and 'Crear Cuenta' buttons on the right. Below the header, the word 'flickr' is written in its signature blue and pink font. To the right of the logo is a large, stylized 'BY' logo followed by a handwritten signature of 'Sanchez'. A red arrow points from the bottom left towards the male gender icon (♂) located at the bottom left of the screen.

Y todas sus opciones

A screenshot of the Flickr login page with the 'Accesibilidad' (Accessibility) menu open. The menu is a sidebar on the left side of the screen, containing various icons and options for assistive technology. At the bottom of this sidebar, there's a teal bar with the text 'Aprendé más sobre EqualWeb'. The main content area shows the same Flickr login form as the previous screenshot, with the male gender icon visible at the bottom left.

Y así se vería

The image shows a side-by-side comparison. On the left is a dark-themed accessibility toolbar from EqualWeb, featuring icons for text size, contrast, and keyboard navigation, along with options for light mode, text magnification, and color customization. On the right is the Flickr 'Iniciar Sesión' (Log In) page, which has a bright yellow background. The Flickr logo ('flickr') is partially visible at the top. The log-in form includes fields for 'Email' and 'Contraseña' (Password), a 'INICIAR SESIÓN' (Log In) button with a play icon, and a link for new users to 'Create una ahora' (Create one now). A circular logo with the name 'Sanchez' is also present.