Instalación de Ionic

Instalar Ionic CLI

Para poder instalar Ionic CLI debemos comprobar que tenemos instalado Node.js. Una vez comprobado instalamos **Ionic cli** con `npm`:

```
npm install -g @ionic/cli
```

```
D:\Documents\GitHub\Aythami_Javier_PRW>npm install -g @ionic/cli
C:\Users\Javier\AppData\Roaming\npm\ionic -> C:\Users\Javier\AppData\Roaming\npm\node_modules\@ionic\cli\bin\ionic
+ @ionic/cli@6.14.1
updated 2 packages in 9.786s
```

Si tenemos una instalación previa, deberá desinstalarse debido a un cambio en el nombre del paquete.

```
npm uninstall -g ionic
npm install -g @ionic/cli
```

Crear una aplicación

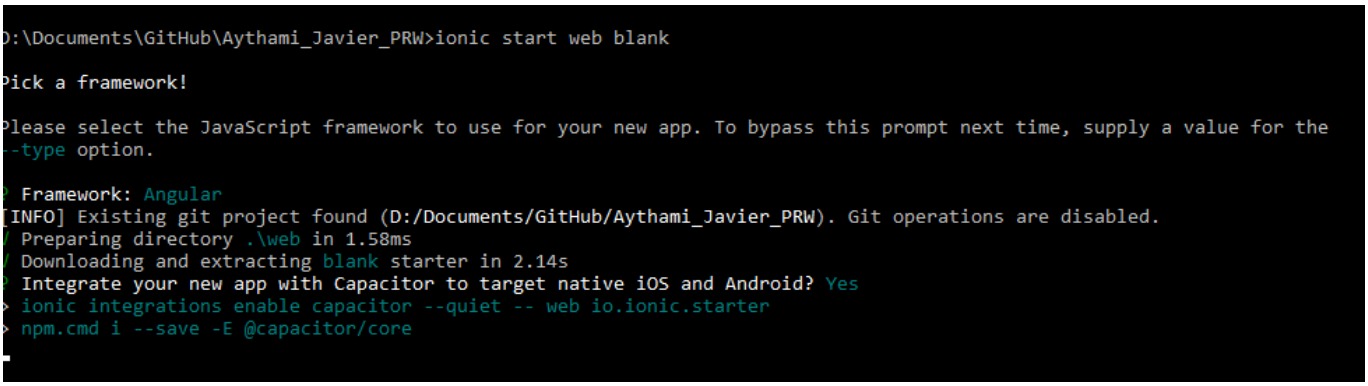
Para crear una aplicación en ionic utilizaremos la plantilla en blanco `blank`. También existen plantillas prefabricadas con un menú lateral `sidemenu` o con un iniciador de pestañas `tabs`. Para la creación utilizaremos `ionic start`.

```
ionic start web blank
```

```
D:\Documents\GitHub\Aythami_Javier_PRW>ionic start web blank

Pick a framework!

Please select the JavaScript framework to use for your new app. To bypass this prompt next time, supply a value for the
--type option.

? Framework: (Use arrow keys)
> Angular | https://angular.io
  React   | https://reactjs.org
  Vue     | https://vuejs.org
```

Una vez lanzado el comando nos mostrará una lista de frameworks para utilizar con **Ionic**. En nuestro caso utilizaremos **Angular**.

Otra pregunta en la instalación es si queremos integrar `Capacitor`, que es un puente de **Ionic** hacia lo nativo, permitiendo acceder a los recursos nativos de los dispositivos, permitiendo una comunicación sencilla.

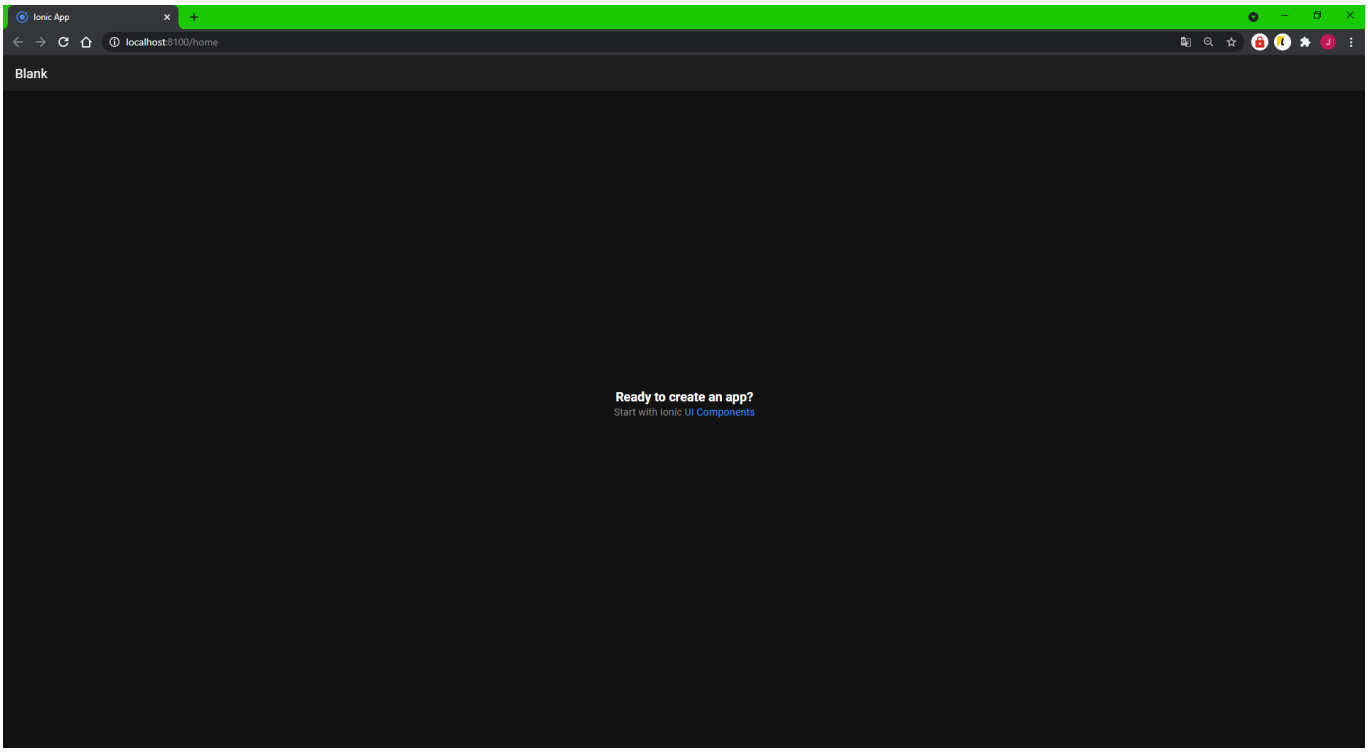


Lanzar aplicación

Para lanzar la aplicación de **Ionic** debemos situarnos en la raíz de la aplicación y ejecutar el comando `ionic serve`.

```
cd web
ionic serve
```

```
D:\Documents\GitHub\Aythami_Javier_PRW>cd web

D:\Documents\GitHub\Aythami_Javier_PRW\web>ionic serve
> ng.cmd run app:serve --host=localhost --port=8100
[ng] Compiling @angular/core : es2015 as esm2015
[ng] Compiling @angular/common : es2015 as esm2015
[ng] Compiling @angular/platform-browser : es2015 as esm2015
[ng] Compiling @angular/forms : es2015 as esm2015
[ng] Compiling @angular/platform-browser-dynamic : es2015 as esm2015
[ng] Compiling @angular/router : es2015 as esm2015
[ng] Compiling @ionic/angular : es2015 as esm2015
[ng] - Generating browser application bundles...
[ng] √ Browser application bundle generation complete.
[ng] Initial Chunk Files           | Names                         |      Size
[ng] vendor.js                     | vendor                        |   4.86 MB
[ng] polyfills.js                  | polyfills                     | 598.64 kB
[ng] styles.css, styles.js         | styles                        | 378.21 kB
[ng] main.js                       | main                          |  14.38 kB
[ng] runtime.js                    | runtime                       |   9.53 kB
[ng] | Initial Total               |   5.84 MB
[ng] Lazy Chunk Files              | Names                         |      Size
[ng] swiper-bundle-44a9b1f9-js.js  | swiper-bundle-44a9b1f9-js     | 200.02 kB
[ng] polyfills-core-js.js          | polyfills-core-js             |  92.39 kB
[ng] 11.js                         | -                             |  72.60 kB
[ng] 17.js                         | -                             |  72.29 kB
[ng] 2.js                          | -                             |  63.70 kB
[ng] 33.js                         | -                             |  56.49 kB
[ng] 1.js                          | -                             |  47.59 kB
[ng] 26.js                         | -                             |  47.15 kB
[ng] 31.js                         | -                             |  37.46 kB
[ng] 30.js                         | -                             |  32.95 kB
[ng] 21.js                         | -                             |  32.50 kB
[ng] 6.js                          | -                             |  32.27 kB
[ng] 19.js                         | -                             |  32.15 kB
[ng] 16.js                         | -                             |  31.12 kB
[ng] 0.js                          | -                             |  29.80 kB
[ng] 20.js                         | -                             |  26.68 kB
[ng] 12.js                         | -                             |  26.41 kB
[ng] 25.js                         | -                             |  26.28 kB
[ng] 32.js                         | -                             |  24.53 kB
[ng] 36.js                         | -                             |  23.93 kB
[ng] 29.js                         | -                             |  23.77 kB
[ng] 23.js                         | -                             |  21.94 kB
[ng] 40.js                         | -                             |  21.74 kB
[ng] 22.js                         | -                             |  21.05 kB
[ng] 15.js                         | -                             |  20.55 kB
[ng] polyfills-dom.js              | polyfills-dom                 |  19.47 kB
[ng] 39.js                         | -                             |  18.33 kB
[ng] 42.js                         | -                             |  17.93 kB
[ng] 41.js                         | -                             |  16.99 kB
[ng] 7.js                          | -                             |  15.95 kB
[ng] 24.js                         | -                             |  15.84 kB
[ng] shadow-css-a3f00b33-js.js     | shadow-css-a3f00b33-js        |  15.82 kB
[ng] 4.js                          | -                             |  15.77 kB
[ng] 10.js                         | -                             |  15.70 kB
```



Dependencias

Las dependencias utilizadas en el proyecto son:

- Material

  Es una librería de guías de estilos creada por Google. Utilizada principalmente en los formularios de la aplicación. Instalación:

  ```
  ng add @angular/material
  ```

  En este proceso nos preguntará el tema que vamos a utilizar que en nuestro caso es `Deep Purple/Amber`, si queremos la tipografía `yes` y por último que si queremos las animaciones `yes`.

  ```
  Javier@LAPTOP-4HUETSTQ MINGW64 /d/Documents/GitHub/Aythami_Javier_PRW/web (feature/autenticación_de_usuarios
  )
  $ ng add @angular/material
  Skipping installation: Package already installed
  ? Choose a prebuilt theme name, or "custom" for a custom theme: Deep Purple/Amber  [ Preview: https://materi
  al.angular.io?theme=deeppurple-amber ]
  ? Set up global Angular Material typography styles? Yes
  ? Set up browser animations for Angular Material? Yes
  UPDATE package.json (2011 bytes)
  ✓ Packages installed successfully.
  UPDATE src/app/app.module.ts (745 bytes)
  UPDATE angular.json (5352 bytes)
  UPDATE src/index.html (1044 bytes)
  UPDATE node_modules/@angular/material/prebuilt-themes/deeppurple-amber.css (75847 bytes)
  ```

- Ionic Storage

  Es un módulo para el almacenamiento de valor clave simple para las aplicaciones hechas con ionic, pudiendo ser utilizada en web como en los dispositivos móviles, facilitando el proceso de traspaso a

android e ios. Instalación:

```
npm install @ionic/storage-angular
```

```
Javier@LAPTOP-4HUETSTQ MINGW64 /d/Documents/GitHub/Aythami_Javier_PRW/web (feature/login_usuarios)
$ npm install @ionic/storage-angular
npm WARN karma-jasmine-html-reporter@1.6.0 requires a peer of jasmine-core@>=3.7.1 but none is installed. You must install peer dependencies yourself.

+ @ionic/storage-angular@3.0.6
added 2 packages from 1 contributor and audited 1731 packages in 15.486s

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

found 2 vulnerabilities (1 moderate, 1 high)
  run `npm audit fix` to fix them, or `npm audit` for details
```

En la instalación surgió una vulnerabilidad grave. Para corregirlo utilice un comando que recorre las dependencias e intenta corregir el error

```
npm audit fix --force
```

```
Javier@LAPTOP-4HUETSTQ MINGW64 /d/Documents/GitHub/Aythami_Javier_PRW/web (feature/login_usuarios)
$ npm audit fix --force
npm WARN using --force I sure hope you know what you are doing.
npm WARN karma-jasmine-html-reporter@1.6.0 requires a peer of jasmine-core@>=3.7.1 but none is installed. You must install peer dependencies yourself.

+ karma@6.3.2
added 5 packages from 7 contributors, removed 30 packages and updated 11 packages in 15.117s

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

fixed 1 of 2 vulnerabilities in 1731 scanned packages
  1 vulnerability required manual review and could not be updated
  1 package update for 1 vulnerability involved breaking changes
  (installed due to `--force` option)
```

- DropZone

  Dropzone es una dependencia que nos permite el poder arrastrar los archivos y soltarlos dentro de la app, facilitando al usuario la selección de los archivos. Instalación:

```
npm install --save ngx-dropzone
```

```
Javier@LAPTOP-4HUETSTQ MINGW64 /d/Documents/GitHub/Aythami_Javier_PRW/web (feature/dashboard)
$ npm install --save ngx-dropzone
npm WARN karma-jasmine-html-reporter@1.6.0 requires a peer of jasmine-core@>=3.7.1 but none is installed. You must install peer dependencies yourself.

+ ngx-dropzone@3.0.0
added 1 package from 1 contributor, removed 1 package and audited 1707 packages in 36.135s

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

found 1 moderate severity vulnerability
  run `npm audit fix` to fix them, or `npm audit` for details
```

Organización del proyecto

Core

En el directorio **app/core** estarán los `servicios`, las `directivas` y los `guard` que utilizara la aplicación.

- Servicios

  Un servicio trabaja de manera síncrona con datos y llamadas. Para crear un servicio utilizaremos el siguiente comando:

  ```
  ng generate service [Nombre del Servicio]

  // Reducido
  ng g s [Nombre del Servicio]
  ```

  Actualmente disponemos de 4 servicios:

  - auth

    ```
    Javier@LAPTOP-4HUETSTQ MINGW64 /d/Documents/GitHub/Aythami_Javier_PRW/web (feature/login_usuarios)
    $ ng g s core/service/auth/auth
    CREATE src/app/core/service/auth/auth.service.spec.ts (347 bytes)
    CREATE src/app/core/service/auth/auth.service.ts (133 bytes)
    ```

    El `authService` se encarga de la autenticación de usuarios, mandando peticiones a la API

    - Iniciar sesión

      ```
      login(email: string, password: string): Observable<any> {
        return this.http.post(URL + 'login', {
          email,
          password
        }, httpOptions);
      }
      ```

    - Registro

      ```
      register(name: string, email: string, password: string, password_confirm: string): Observable<any> {
        return this.http.post(URL + 'createUser', {
          name,
          email,
          password,
          password_confirm
        }, httpOptions);
      }
      ```

    - Olvido de contraseña

      ```
      forgot(email: string): Observable<any> {
        return this.http.post(URL + 'forgot', {
          email
        }, httpOptions);
      }
      ```

    - Resetear contraseña

```
resetPassword(token: string, password: string, password_confirm: string): Observable<any> {
  return this.http.post(URL + 'restpassword', {
    token,
    password,
    password_confirm
  }, httpOptions)
}
```

- Cerrar sesión

```
logout(token: string): Observable<any> {
  return this.http.post(URL + 'logout', {
    token
  }, httpOptions);
}
```

- storage

```
Javier@LAPTOP-4HUETSTQ MINGW64 /d/Documents/GitHub/Aythami_Javier_PRW/web (feature/login_usuarios)
$ ng g s core/service/storage/storage
CREATE src/app/core/service/storage/storage.service.spec.ts (362 bytes)
CREATE src/app/core/service/storage/storage.service.ts (136 bytes)
```

El `storageService` se encarga de almacenar y gestionar los datos del usuario guardados en **ionic storage**

- Crear Almacenamiento

```
async create(): Promise<void> {
  await this.storage.create();
}
```

- Guardar Token

```
saveToken(token: string): void {
  this._storage?.set(KEY_TOKEN, token);
}
```

- Guardar Usuario

```
saveUser(user: any): void {
  this._storage?.set(KEY_USER, JSON.stringify(user));
}
```

- Recoger Token

```
async getToken(): Promise<string> {
  return await this._storage?.get(KEY_TOKEN);
}
```

- Recoger Usiario

```
async getUser(): Promise<any> {
  return await JSON.parse(await this._storage?.get(KEY_USER));
}
```

- Comprobar si existe un Token

```
async isAuth(): Promise<boolean> {
  if (await this.getToken() === null) {
    return false;
  } else {
    return true;
  }
}
```

- Limpiar Almacenamiento

```
async clearStorage(): Promise<void> {
  await this.storage.clear();
}
```

- file

```
Javier@LAPTOP-4HUETSTQ MINGW64 /d/Documents/GitHub/Aythami_Javier_PRW/web (feature/dashboard)
$ ng g s core/service/file/file
CREATE src/app/core/service/file/file.service.spec.ts (347 bytes)
CREATE src/app/core/service/file/file.service.ts (133 bytes)
```

El `fileService` se encarga del almacenamiento y control de archivos, mandando peticiones a la API. En la cabecera de estas peticiones se envía el token del usuario, sino el servidor nos devolverá un error al no identificar al usuario.

```
async header(contentType: boolean = false): Promise<{headers: HttpHeaders;}> {
  if (contentType) {
    return {headers: new HttpHeaders({ 'Content-Type': 'application/json', 'Authorization': 'Bearer ' + await this.storage.getToken()})};
  } else {
    return {headers: new HttpHeaders({'Authorization': 'Bearer ' + await this.storage.getToken()})};
  }
}
```

- Recoger todos los archivos del usuario

```
async index(): Promise<Observable<any>> {
  return this.http.get(URL + 'index', await this.header(true));
}
```

- Subir un archivo

```
async store(fileForm: FormData): Promise<Observable<any>> {
  return this.http.post(URL + 'store', fileForm, await this.header());
}
```

- Editar un archivo

```
async update(fileForm: FormData): Promise<Observable<any>> {
  return this.http.post(URL + 'update', fileForm, await this.header());
}
```

- Recoger los datos de un archivo

```
async show(id: number): Promise<Observable<any>> {
  return this.http.get(URL + id, await this.header(true));
}
```

- Descargar el archivo

```
async recuperarArchivo(id: number): Promise<Observable<any>> {
  return this.http.get(URL + 'recuperarArchivo/' + id, await this.header(true));
}
```

- Mostrar todos los archivos de una categoria

```
async showCategoria(id: number): Promise<Observable<any>> {
  return this.http.get(URL + 'showCategoria/' + id, await this.header(true));
}
```

- Eliminar un archivo

```
async delete(id: number): Promise<Observable<any>> {
  return this.http.post(URL + 'delete/' + id, null, await this.header(true));
}
```

- cetegoria

El `categoriaService` se encarga del almacenamiento y control de las categorías, mandando peticiones a la API. En la cabecera de estas peticiones se envía el token del usuario, sino el servidor nos devolverá un error al no identificar al usuario.

```
async header(): Promise<{headers: HttpHeaders;}> {
  return {headers: new HttpHeaders({'Content-Type': 'application/json', 'Authorization': 'Bearer ' + await this.storage.getToken()})};
}
```

- Recoger todas las categorías del usuario

```
async index(): Promise<Observable<any>> {
  return this.http.get(URL + 'index', await this.header());
}
```

- Crear categoría

```
async store(name: string, idCategoria: string): Promise<Observable<any>> {
  return this.http.post(URL + 'store', {
    name,
    idCategoria
  }, await this.header());
}
```

- Editar categoría

```
async update(id: number, name: string, idCategoria: string): Promise<Observable<any>> {
  return this.http.post(URL + 'update', {
    id,
    name,
    idCategoria
  }, await this.header());
}
```

- Recoger los datos de una categoría

```
async show(id: number): Promise<Observable<any>> {
  return this.http.post(URL + id, null, await this.header());
}
```

- Eliminar categoría

```
async destroy(id: number): Promise<Observable<any>> {
  return this.http.post(URL + 'delete/' + id, null, await this.header());
}
```

- Directiva

  Una directiva es una serie de elementos que aplicaremos a nuestro código HTML como si de un atributo se tratara con el fin de añadir una nueva funcionalidad

  ```
  ng generate directive [Nombre de la Directiva]
  ```

  - password-equal-validator

    Se encarga de comprobar en tiempo real que las dos contraseñas que se introducen en los formularios de `resetPassword` y `register` sean iguales. En el caso que no sean iguales se creará un `ValidatorError`

    ```
    export class PasswordEqualValidatorDirective implements Validator {
      @Input() appPasswordEqualValidator: string;

      validate(control: AbstractControl): ValidationErrors | null {
          const controlToCompare = control.parent.get(this.appPasswordEqualValidator);
          if (controlToCompare) {
              const subcription: Subscription = controlToCompare.valueChanges.subscribe(() =>{
                  control.updateValueAndValidity();
                  subcription.unsubscribe();
              });
          }

          return controlToCompare && controlToCompare.value !== control.value ? { 'passwordNotEqual': true } : null;
      }
    ```

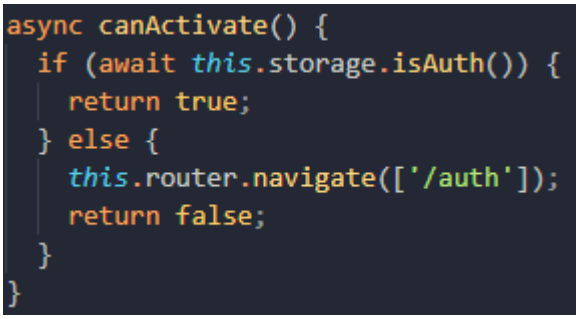
- Guard

  Un guard es un middlewares que se ejecutan antes de cargar una ruta y determinan si se puede cargar dicha ruta o no.

  ```
  ng generate guard [Nombre del Guard]
  ```

- auth

  Con la función canActive comprobamos si el usuario está logueado. De lo contrario no podrá acceder



```
async canActivate() {
  if (await this.storage.isAuth()) {
    return true;
  } else {
    this.router.navigate(['/auth']);
    return false;
  }
}
```

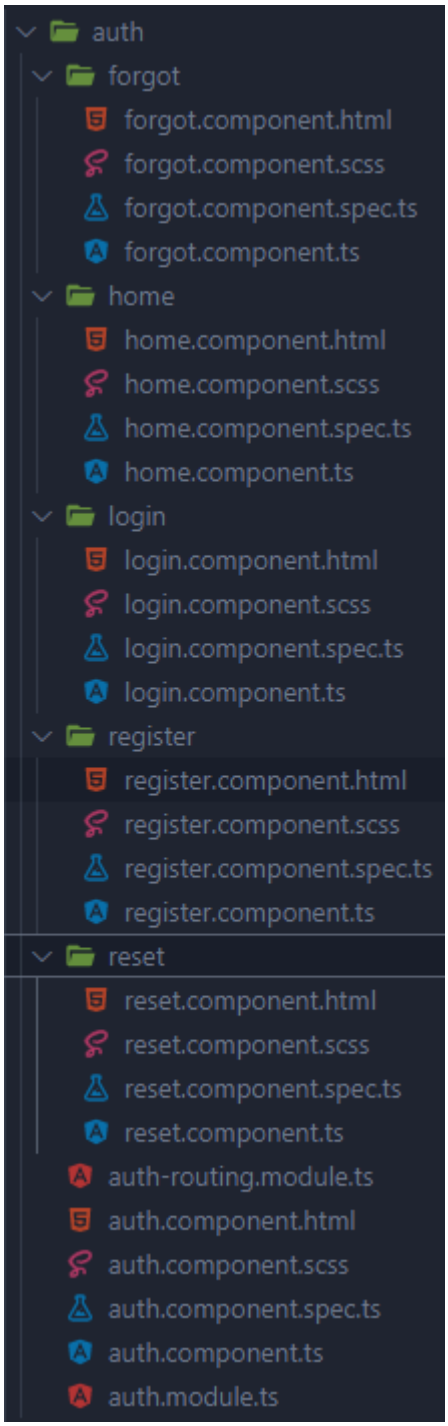
Views

En el directorio **app/views** estarán todas las vistas de la aplicación. Dentro de este directorio encontramos la app donde se inicializa la aplicación.

```
// Crear Modulo
ng generate module [Nombre del Modulo]
ng g m [Nombre del Modulo]

// Crear Routing
ng generate module [Nombre del Modulo]-routing --flat --module=[Nombre del Modulo]
ng g m [Nombre del Modulo]-routing --flat --module=[Nombre del Modulo]

// Crear Componente
ng generate component [Nombre del Componente]
ng g c [Nombre del Componente]
```

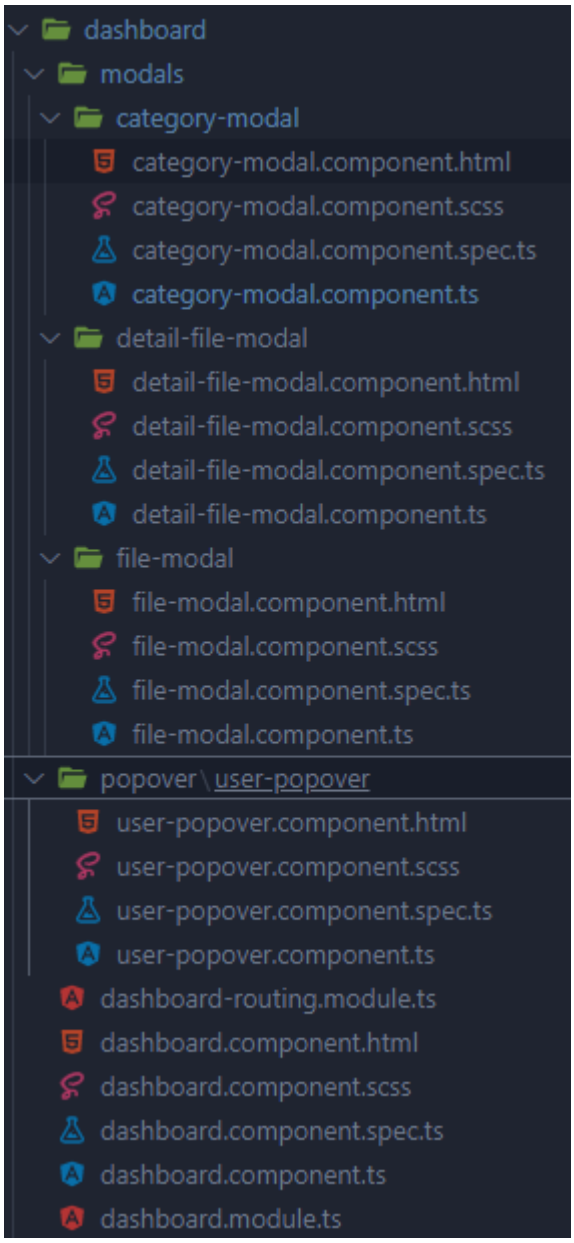
- Auth

  En el directorio **app/views/auth** encontramos las vistas encargadas de la creación y control de los usuarios, donde podemos crear una cuenta, iniciar sesión, recuperar cuenta y resetear la contraseña



- Dashboard

  En el directorio **app/views/dashboard** encontramos todas las vistas de la ejecucion principal de la aplicación. Para poder acceder a estas vistas debemos iniciar sesión, sino el guard no redirige al login



Problemas

Implementación de la plantilla Metronic Demo1

Tanto mi compañero como yo nos habíamos puesto de acuerdo y decidimos que íbamos a implementar la plantilla Metronic Demo1 de Angular que nos parecía muy adecuada con el diseño que habíamos presentado en el anteproyecto.

Cuando empece a implementar, me comenzó a aparecer muchos errores en distintos sitios y a su misma vez quería también añadir algunas funciones que a la hora de intentar añadirlas en el proyecto me daba muchos conflictos con las dependencias de la propia plantilla.

Por todo estos errores y problemas que me han surgido y que hemos intentado arreglar, he perdido 3 semanas, viendo que se aproximaba la fecha de entrega y que no habíamos terminado de arreglar los fallos que se producían, hemos decidido empezar a utilizar **Ionic** para poder seguir con el desarrollo del proyecto.

Promesas

Ha sido un poco confuso la utilización de las promesas. Durante el curso no llegamos a verla, su utilización dentro del proyecto ha sido complicada y hemos tenido que utilizar bastante tiempo en contemplar su funcionamiento.

Descarga de los archivos

Otro problema que nos surgió fue a la hora de descargar los archivos, ya que nos saltaba un error de cabecera, error que era un poco confuso pues nos informaba que nos faltaba el `token` pero sabíamos que esto no podía ser así ya que el envío del `token` había sido utilizado anteriormente en el resto de servicios.