

Tutorial



Angular Material

FrontEnd



Instalación

Paso 1. Instalar **Node.js versión LTS** de la página oficial <https://nodejs.org/es>, si ya se tiene una versión anterior es necesario actualizarlo.

Paso 2. Verificar la versión de Node.js ejecutando el comando **node -v** en la consola de Windows (Command Prompt).

```
C:\Users\Usuario>node -v
v16.13.2
```

Paso 3. En la consola de Windows (Command Prompt) ejecutar el siguiente comando para instalar TypeScript: **npm install typescript -g**

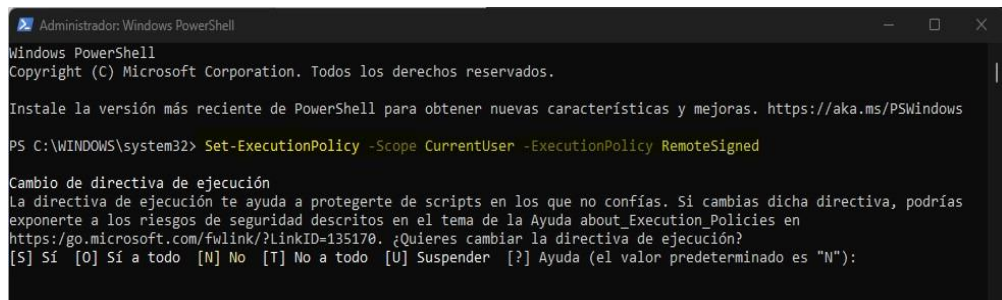
```
C:\Users\Usuario>npm install typescript -g
changed 1 package, and audited 2 packages in 3s
found 0 vulnerabilities
```

Paso 4. Instalar **Angular CLI** ejecutando el siguiente comando en la consola de Windows (Command Prompt): **npm install -g @angular/cli**

```
C:\Users\Usuario>npm install -g @angular/cli
[██████████] \ idealTree:inquirer: timing idealTree:node_1
```

Paso 5. Ejecutar como **administrador** Windows **PowerShell** y ejecutar el siguiente comando para habilitar la ejecución de script globalmente:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned



```
Administrador: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

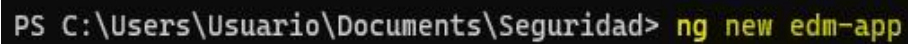
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\WINDOWS\system32> Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

Cambio de directiva de ejecución
La directiva de ejecución te ayuda a protegerte de scripts en los que no confías. Si cambias dicha directiva, podrías exponerte a los riesgos de seguridad descritos en el tema de la Ayuda about_Execution_Policies en https://go.microsoft.com/fwlink/?LinkID=135170. ¿Quieres cambiar la directiva de ejecución?
[S] Sí [O] Sí a todo [N] No [T] No a todo [U] Suspender [?] Ayuda (el valor predeterminado es "N"):
```

Crear Proyecto

Paso 6. Para crear el proyecto abrir Windows (Command Prompt) y dirigirse a la ruta donde se quiera crear el proyecto (se recomienda una carpeta en **Documents\Seguridad**) y ejecutar el comando `ng new iniciales-app` (en el campo iniciales agregar las iniciales del alumno(a)).

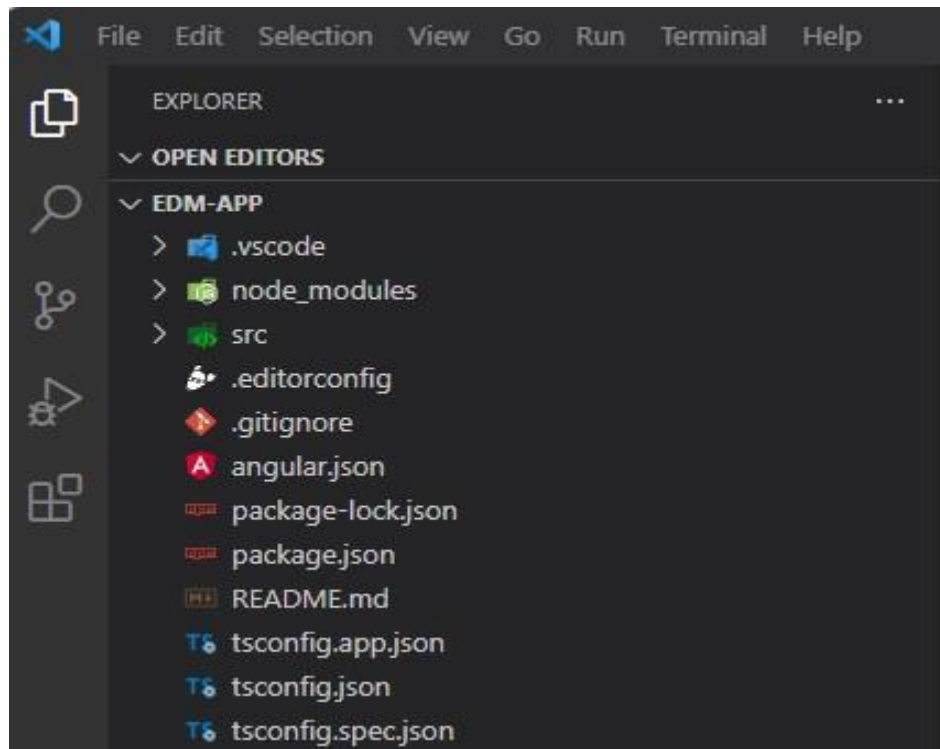


```
PS C:\Users\Usuario\Documents\Seguridad> ng new edm-app
```

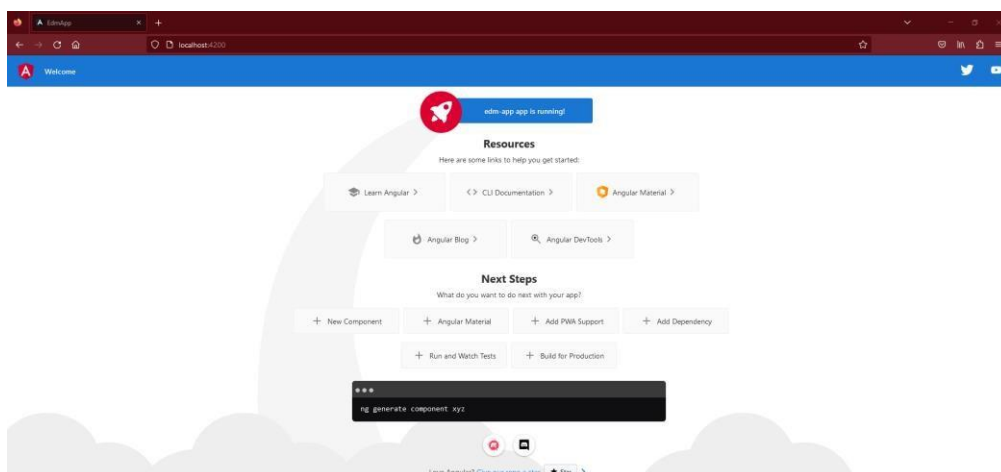
Para configurar el proyecto se deben seguir los siguientes pasos:

- Would you like to add Angular routing?: **Yes**
- Which stylesheet format would you like to use?: **SCSS**

Paso 7. Al finalizar se creará una carpeta con el nombre del proyecto, abrir **Visual Studio Code** y arrastrar la carpeta al espacio de trabajo.



Paso 8. Ir a la pestaña **Terminal** y abrir una **nueva terminal integrada** y ejecutar el comando: `ng serve`, abrir el navegador e ingresar en <http://localhost:4200>, verificar que el proyecto inicial se visualice correctamente:



3 Instalar Angular Material

Paso 9. En la consola integrada de Visual Studio Code se debe ejecutar el siguiente comando: `ng add @angular/material` y elegir el tema de tu agrado:

```
PS C:\Users\Usuario\Documents\Seguridad\edm-app> ng add @angular/material
i Using package manager: npm
✓ Found compatible package version: @angular/material@15.2.3.
✓ Package information loaded.

The package @angular/material@15.2.3 will be installed and executed.
Would you like to proceed? Yes
✓ Packages successfully installed.
? Choose a prebuilt theme name, or "custom" for a custom theme:
Indigo/Pink [ Preview: https://material.angular.io/theme=indigo-pink ]
> Deep Purple/Amber [ Preview: https://material.angular.io/theme=deeppurple-amber ]
Pink/Blue Grey [ Preview: https://material.angular.io/theme=pink-bluegrey ]
Purple/Green [ Preview: https://material.angular.io/theme=purple-green ]
Custom
```

Paso 10. Crear un módulo con la siguiente instrucción `ng g m material --flat`

Paso 11. Modificar el archivo para y crear la lista `myModules`, importa y exporta el módulo:

```
1 import { NgModule } from "@angular/core";
2
3 const myModules: any = [];
4
5 @NgModule({
6   imports: [... myModules],
7   exports: [... myModules]
8 })
9
10 export class MaterialModule { }
```

Paso 12. En el archivo `app.module.ts` agregar `MaterialModule` en la sección de los `imports`

```
9 @NgModule({
10   declarations: [
11     AppComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule,
16     BrowserModuleAnimationsModule,
17     MaterialModule
18   ],
```

Agregar y configurar componentes y módulos

Paso 1. Agregar un nuevo módulo para la pantalla de inicio:

`ng g m pages/home -m=app --route home`

```
PS C:\Users\Usuario\Documents\Seguridad\edm-app> ng g m pages/home -m=app --route home
CREATE src/app/pages/home/home-routing.module.ts (335 bytes)
CREATE src/app/pages/home/home.module.ts (343 bytes)
CREATE src/app/pages/home/home.component.html (19 bytes)
CREATE src/app/pages/home/home.component.spec.ts (585 bytes)
CREATE src/app/pages/home/home.component.ts (195 bytes)
CREATE src/app/pages/home/home.component.scss (0 bytes)
UPDATE src/app/app-routing.module.ts (341 bytes)
```

Paso 2. Crear el componente **header**:

`ng g c shared/components/header -m=app`

```
PS C:\Users\Usuario\Documents\Seguridad\edm-app> ng g c shared/components/header -m=app
CREATE src/app/shared/components/header/header.component.html (21 bytes)
CREATE src/app/shared/components/header/header.component.spec.ts (599 bytes)
CREATE src/app/shared/components/header/header.component.ts (203 bytes)
CREATE src/app/shared/components/header/header.component.scss (0 bytes)
UPDATE src/app/app.module.ts (674 bytes)
PS C:\Users\Usuario\Documents\Seguridad\edm-app>
```

Paso 3. Crear el componente **footer**:

`ng g c shared/components/footer -m=app`

```
PS C:\Users\Usuario\Documents\Seguridad\edm-app> ng g c shared/components/footer -m=app
CREATE src/app/shared/components/footer/footer.component.html (21 bytes)
CREATE src/app/shared/components/footer/footer.component.spec.ts (599 bytes)
CREATE src/app/shared/components/footer/footer.component.ts (203 bytes)
CREATE src/app/shared/components/footer/footer.component.scss (0 bytes)
UPDATE src/app/app.module.ts (774 bytes)
```

Paso 4. Crear el módulo de inicio de sesión:

`ng g m pages/auth/login -m=app --route login`

```
PS C:\Users\Usuario\Documents\Seguridad\edm-app> ng g m pages/auth/login -m=app --route login
CREATE src/app/pages/auth/login/login-routing.module.ts (339 bytes)
CREATE src/app/pages/auth/login/login.module.ts (350 bytes)
CREATE src/app/pages/auth/login/login.component.html (20 bytes)
CREATE src/app/pages/auth/login/login.component.spec.ts (592 bytes)
CREATE src/app/pages/auth/login/login.component.ts (199 bytes)
CREATE src/app/pages/auth/login/login.component.scss (0 bytes)
UPDATE src/app/app-routing.module.ts (448 bytes)
```

Configuración de la pantalla de inicio de sesión y rutas

Paso 1. Modificar el archivo **style.scss**

```
html, body {height: 100%;}

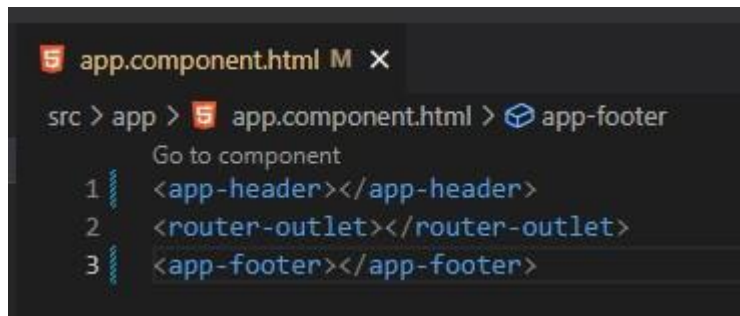
body      {
margin: 0;
  font-family: Roboto, "Helvetica Neue", sans-serif;
}

.error-snackbar {
  background-color:    darkmagenta;
color: white;
}

.success-snackbar {
  background-color: darkcyan;
  color: white;
}

.full-width {
  width: 100%;
}
```

Paso 2. Modificar el archivo **app.component.html**



```
app.component.html M X
src > app > app.component.html > app-footer
Go to component
1 <app-header></app-header>
2 <router-outlet></router-outlet>
3 <app-footer></app-footer>
```

Paso 3. Modificar el archivo **app-routing.module.ts**


```

TS app-routing.module.ts M X
src > app > TS app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3
4 const routes: Routes = [
5   {
6     path: '', redirectTo: 'login', pathMatch: 'full'
7   },
8   {
9     path: 'home',
10    loadChildren: () => import('./pages/home/home.module').then(m => m.HomeModule)
11  },
12  {
13    path: 'login',
14    loadChildren: () => import('./pages/auth/login/login.module').then(m => m.LoginModule)
15  }
16 ];
17
18 @NgModule({
19   imports: [RouterModule.forRoot(routes)],
20   exports: [RouterModule]
21 })
22 export class AppRoutingModule { }

```

Paso 4. En el archivo **pages/auth/login/login.component.ts** modificarlo de la siguiente manera, se debe verificar que contenga implementada la interfaz **OnInit** y el formulario reactivo a utilizar.

Con formularios reactivos, define el modelo de formulario directamente en la clase de componente.

[Validators Angular](#)

```

TS login.component.ts U X
src > app > pages > auth > login > TS login.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { FormBuilder, Validators } from '@angular/forms';
3
4 @Component({
5   selector: 'app-login',
6   templateUrl: './login.component.html',
7   styleUrls: ['./login.component.scss']
8 })
9 export class LoginComponent implements OnInit {
10
11   loginForm = this.fb.group({
12     username : ['', [Validators.required]],
13     password : ['', [Validators.required, Validators.minLength(3)]]
14   });
15   constructor(private fb: FormBuilder) { }
16
17   ngOnInit(): void { }
18
19 }

```

Paso 5. Crear un archivo en la ruta **shared/utils/base-form.ts** (si no existe la ruta y/o carpeta se debe crear).

Paso 6. Modificar el archivo **shared/utils/base-form.ts**


```
TS base-form.ts U X
src > app > shared > utils > TS base-form.ts > ...
1  import { Injectable } from "@angular/core";
2
3  @Injectable({ providedIn: 'root' })
4  export class BaseForm {
5
6      constructor() { }
7
8
9
10 }
```

Paso 7. Agregar el método **isValidField** al archivo **base-form.ts**

```
isValidField(form: AbstractControl|null) {
    var flag = false;
    if (form != null) {
        flag = form.touched || form.dirty && !form.valid;
    }
    return flag;
}
```

Nota: Verificar las siguientes importaciones en el archivo:

```
TS base-form.ts U X
src > app > shared > utils > TS base-form.ts > ...
1  import { Injectable } from "@angular/core";
2  import { AbstractControl } from "@angular/forms";
3
```

Paso 8. Agregar el método **getErrorMessage**

```
getErrorMessage(form: AbstractControl|null) {  
  let message = "";  
  if (form) {  
    const { errors } = form;  
    if (errors) {  
      const messages: any = {  
        required: 'Campo requerido',  
        email: 'Formato inválido',  
        pattern: 'Formato inválido',  
        minError: 'El rango no es correcto',  
        min: 'El rango no es correcto',  
        max: 'El rango no es correcto'  
      };  
  
      const errorKey = Object.keys(errors).find(Boolean);  
      if (errorKey) {  
        message = messages[errorKey];  
      }  
    }  
  }  
  return message;  
}
```

Paso 9. Importar la clase **base-form.ts** al archivo **login.component.ts**

```
16 constructor(private fb: FormBuilder, public baseForm: BaseForm) { }  
17
```

Paso 10. Modificar el archivo **material.module.ts**

```

src > app > TS material.module.ts > ...
1  import { NgModule } from "@angular/core";
2  import { MatButtonModule } from '@angular/material/button';
3  import { MatCardModule } from '@angular/material/card';
4  import { MatInputModule } from '@angular/material/input';
5  import { MatIconModule } from '@angular/material/icon';
6  import { MatGridListModule } from '@angular/material/grid-list';
7
8  const myModules: any = [
9      MatButtonModule,
10     MatCardModule,
11     MatInputModule,
12     MatIconModule,
13     MatGridListModule,
14 ];
15
16 @NgModule({
17     imports: [... myModules],
18     exports: [... myModules]
19 })
20
21 export class MaterialModule { }
22

```

Paso 5.10: Modificar el archivo **login.module.ts** (en la sección imports), se debe agregar el módulo de **ReactiveFormsModule** y **MaterialModule**.

```

src > app > pages > auth > login > TS login.module.ts > LoginModule
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3
4  import { LoginRoutingModule } from './login-routing.module';
5  import { LoginComponent } from './login.component';
6  import { MaterialModule } from 'src/app/material.module';
7  import { ReactiveFormsModule } from '@angular/forms';
8
9
10 @NgModule({
11     declarations: [
12         LoginComponent
13     ],
14     imports: [
15         CommonModule,
16         LoginRoutingModule,
17         ReactiveFormsModule,
18         MaterialModule
19     ]
20 })
21 export class LoginModule { }
22

```

Paso 11. Modificar el archivo **login.component.ts**

```
<mat-grid-list cols="1" rowHeight="100%" class="fondo">
```

```

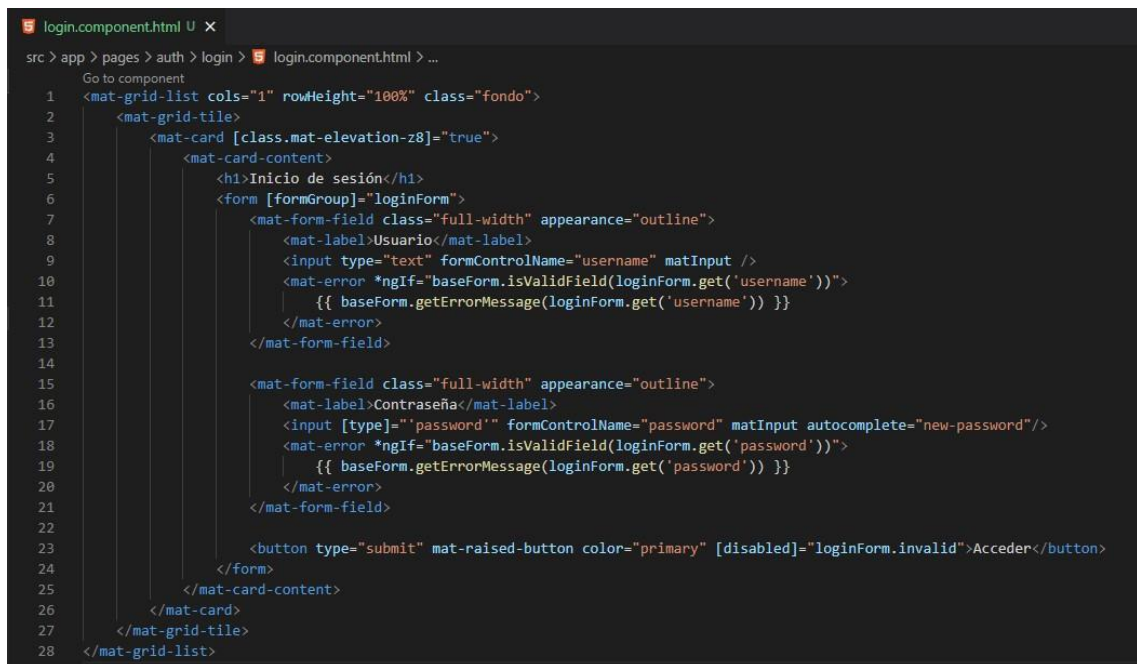
<mat-grid-tile>
  <mat-card [class.mat-elevation-z8]="true">
    <mat-card-content>
      <h1>Inicio de Sesión</h1>
      <form [formGroup]="loginForm" (ngSubmit)="onLogin()">
        <mat-form-field class="full-width" appearance="outline">
          <mat-label>Usuario</mat-label>
          <mat-error *ngIf="baseForm.isValidField(loginForm.get('username'))">
            {{ baseForm.getErrorMessage(loginForm.get('username'))}}
          </mat-error>
        </mat-form-field>

        <mat-form-field class="full-width" appearance="outline">
          <mat-label>Contraseña</mat-label>
          <mat-error *ngIf="baseForm.isValidField(loginForm.get('password'))">
            {{baseForm.getErrorMessage(loginForm.get('password'))}}
          </mat-error>
        </mat-form-field>

      </form>
    </mat-card-content>
  </mat-card>
</mat-grid-tile>

</mat-grid-list>

```



```

login.component.html U X
src > app > pages > auth > login > login.component.html > ...
Go to component
1 <mat-grid-list cols="1" rowHeight="100%" class="fondo">
2   <mat-grid-tile>
3     <mat-card [class.mat-elevation-z8]="true">
4       <mat-card-content>
5         <h1>Inicio de sesión</h1>
6         <form [formGroup]="loginForm">
7           <mat-form-field class="full-width" appearance="outline">
8             <mat-label>Usuario</mat-label>
9             <input type="text" formControlName="username" matInput />
10            <mat-error *ngIf="baseForm.isValidField(loginForm.get('username'))">
11              {{ baseForm.getErrorMessage(loginForm.get('username')) }}
12            </mat-error>
13          </mat-form-field>
14
15          <mat-form-field class="full-width" appearance="outline">
16            <mat-label>Contraseña</mat-label>
17            <input [type]="password" formControlName="password" matInput autocomplete="new-password"/>
18            <mat-error *ngIf="baseForm.isValidField(loginForm.get('password'))">
19              {{ baseForm.getErrorMessage(loginForm.get('password')) }}
20            </mat-error>
21          </mat-form-field>
22
23          <button type="submit" mat-raised-button color="primary" [disabled]="loginForm.invalid">Acceder</button>
24        </form>
25      </mat-card-content>
26    </mat-card>
27  </mat-grid-tile>
28 </mat-grid-list>

```

Paso 12. Una vez realizado lo anterior debe mostrar lo siguiente en el navegador:

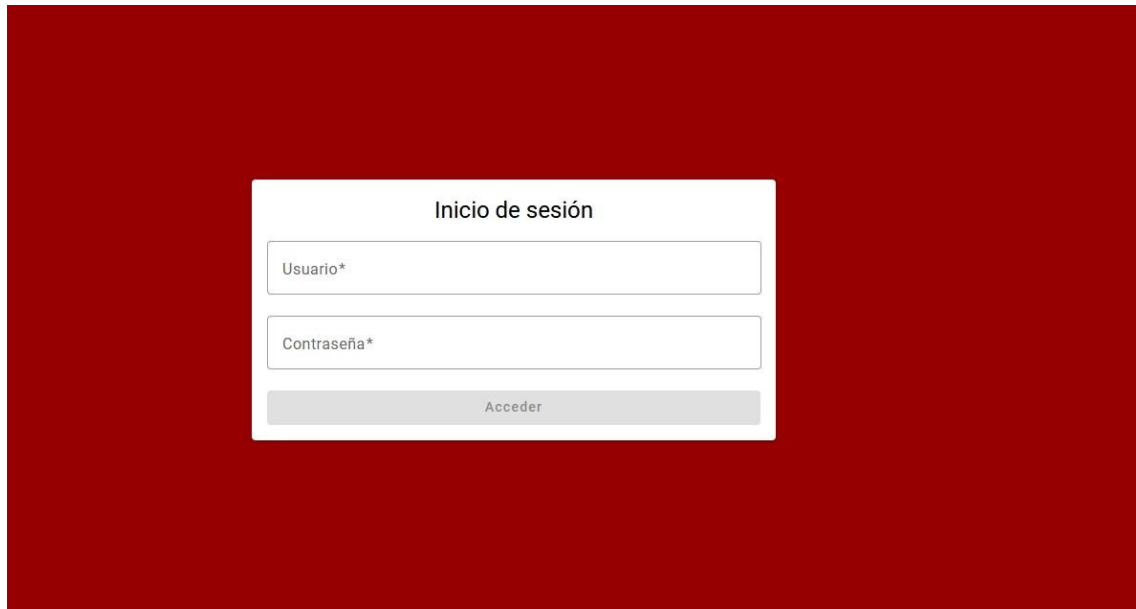
Inicio de sesión

Acceder

Paso 13. Modificar el archivo **login.component.scss** (se puede modificar el color al deseado).

```
login.component.html U X login.component.scss U X
src > app > pages > auth > login > login.component.scss > ...
1  .fondo {
2    height: 100%;
3    background: rgb(150, 0, 0);
4  }
5
6  mat-card {
7    max-width: 40%;
8    text-align: center;
9  }
10
11 button {
12   width: 100%;
13 }
```

Paso 14. Una vez realizado lo anterior debe mostrar lo siguiente en el navegador.



Inicio de sesión

Usuario*

Contraseña*

Acceder

Desafío

- Agregar un header, footer y home personalizado.

Navegación

1. Para realizar la navegación se debe configurar el archivo `login.component.ts`, se agrega la importación de Router y se crea la variable de forma privada en el constructor

```
import { FormBuilder, Validators } from '@angular/forms';
3 import { Router } from '@angular/router';

17 constructor(private fb: FormBuilder, public baseForm: BaseForm, private router: Router) { }
```

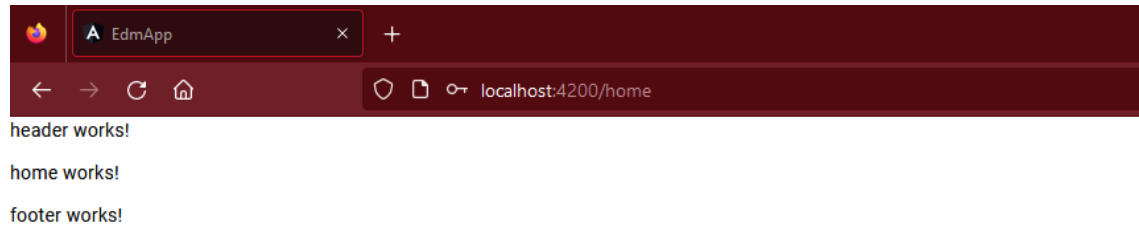
2. Una vez que los datos son correctos se debe de hacer clic en el botón **Ingresar**, pero para ello se debe de programar el método `onLogin()` para navegar a la pantalla **home**

```
onLogin() {
  this.router.navigate(['home']);
}
```

3. Modificar el archivo `login.component.html`, y en la etiqueta del formulario se debe agregar en evento `submit`, como se muestra en la imagen.

```
<form [formGroup]="loginForm" (ngSubmit)="onLogin()">
```

Paso 6.4: Al ejecutar el proyecto en el navegador y dar clic en el formulario deberá mostrar lo siguiente, donde navega a pantalla home



Seguridad

Implementando REcaptcha

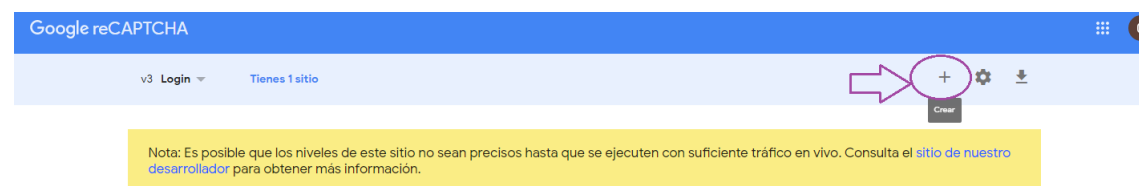
Para mayor referencia véase [Tutorial](#)

Visitar la página [Google reCAPTCHA](#)

Da clic en **v3 AdminConsole**



Crear un nuevo sitio



Ingresa datos y da clic en guardar

Guía de Angular

Etiqueta ⓘ

Login

5/50

Tipo de reCAPTCHA: v3

Claves de reCAPTCHA ▾

Dominios ⓘ

✕ localhost

+ Agregar un dominio, p. ej., ejemplo.com

Propietarios

✕ gbarron@utng.edu.mx

+👤 Ingresar direcciones de correo electrónico

☒ Verificar el origen de las soluciones de reCAPTCHA

Si esta opción está inhabilitada, debes [revisar el nombre del host](#) en tu servidor al verificar una solución.


La clave de sitio es la importante

Configur

Tipo de reCAPTCHA: v3

Claves de reCAPTCHA ^


Inserta esta clave de sitio en el código HTML que utiliza tu sitio. [Ver la integración del lado del cliente](#)

 **COPIAR CLAVE DE SITIO**

6LsfFgmAAAAAwjZ-UpSmdl74O0vcyCeJUnbr7

Utiliza esta clave secreta para la comunicación entre tu sitio y reCAPTCHA.

[Ver la integración del lado del servidor](#)

 **COPIAR CLAVE SECRETA**

6LsfFgmAAAAAwjZ-UpSmdl74O0vcyCeJUnbr7

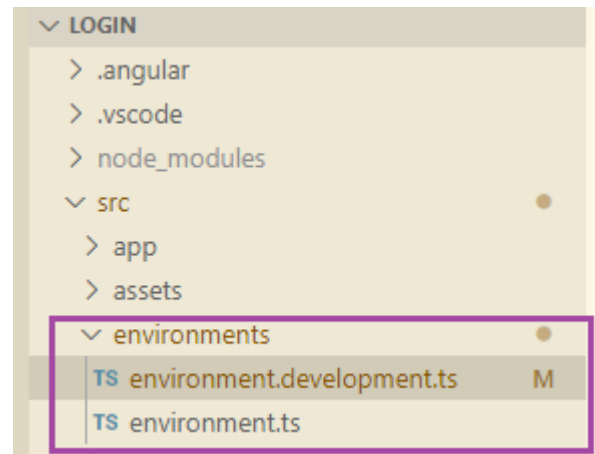
Configuración de entornos Build en angular

Véase referencia [Environments](#)

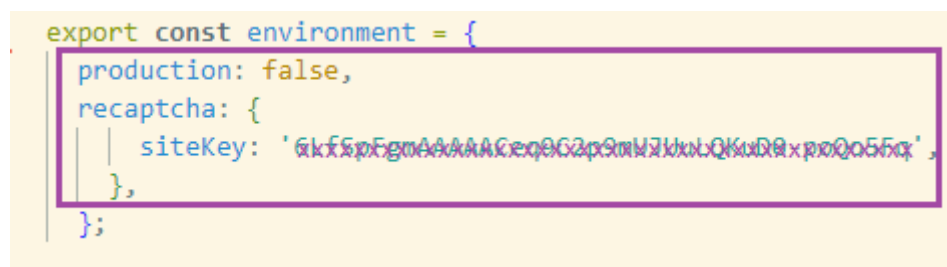
Ingresar el comando en Angular CLI: [ng generate environments](#)

Instalar la librería recaptcha: [npm install ng-recaptcha](#)

Observa la carpeta creada en tu proyecto



Modificar el archivo de configuración environments.ts de acuerdo a tu clave de sitio



Configurar app.module.ts para recaptcha



Abrir el archivo login.component.ts

Importar el servicio recaptcha

```
import { ReCaptchaV3Service } from 'ng-recaptcha';
```

Agregar lo siguiente a la clase:

- Propiedades `tokenVisible`, `reCAPCHAToken`
- Inyectar el servicio `recaptchaV3Service` en constructor
- En método `onLogin` programar el método, observa qué se comentó la ruta que lleva a ruta Home

```
export class LoginComponent implements OnInit {  
  loginForm: any;  
  tokenVisible: boolean = true;  
  reCAPCHAToken: string = "";  
  
  constructor(private fb: FormBuilder, public validaForm: ValidaFormService, public route: Router,  
    private recaptchaV3Service: ReCaptchaV3Service) {  
  }  
  
  ngOnInit(): void {  
    this.loginForm = this.fb.group({  
      username: ['', [Validators.required]],  
      password: ['', [Validators.required, Validators.minLength(3)]],  
    });  
  }  
  
  onLogin() {  
    this.recaptchaV3Service.execute('importantAction').subscribe((token: string) => {  
      this.tokenVisible = true;  
      this.reCAPCHAToken = `Token [${token}] generated`;  
    });  
    //this.route.navigate(['home']);  
  }  
}
```

Modificar archivo HTML agregando un DIV abajo de botón

```
<button type="submit" mat-raised-button color="primary"  
  [disabled]="!loginForm.valid">Ingresar</button>  
<div *ngIf="tokenVisible"> reCAPTCHA Token <br />  
  <p style="color:  
    red">{{reCAPCHAToken}}</p>  
</div>
```

Ingresar el usuario y password y observa el token

Inicio de Sesión

Usuario*

qwe

Contraseña*


.....

Ingresar

reCAPTCHA Token

Token

[03Al.8dmw9J6z6ufh0UpnQznUSYAYNIHW0cgoa76tjys0WM3l7uCLklFuFn0n5XlWfhnl1-1Kq15vhacQ8DBj3Tdp0NK_22VqfoYVzeualRCuRuAJHhQCiz7uBa4fsv6v_3aU8ioecS-wdhjTlkkT0AmHqj-ZGPzp3Ryr0MshIKF5W0mGnWKYF_UZmEvjnQoke3hJg3eIFLayBcT9BQ2vGkXaps7LYJ9nGOSSvFANPI7105JrD0JUsWK5ywLniVBZMKNK7dTU9WlVn3ZGGCngaD3SSNj5H-BIEmCb0xdVN87_XQ3wKD2UQdj37J0LVAI73YvNl2vDVIzf_htr6paE7psCTEOIKDIWsgAdSnJUihp0ilwyyx2niRyVUg6q7ayfGYevQJNEmxEB9fETWbs86ecEBK16tY7hy8bJkq0rtImLvS6lKjsNcoKNy8FjcxVID8VD1Z7YZvRuRuVgQgbAbZmYmk3eJ0jxhVTJIS9Vjbj8a11fpp8_RHnN4K0yVvewqtZ2PIL-O8tOFD3ox7bmhlLcvNoR6P0ooQq18CQqEF1XdNfG0hEe32CtZ0dv7yNENfWcdCDk35UcP_IK8MzHa9czRLv2xbqLfuzFKlu2EgIIX9ckBlhIMFf_19P2ACQJZCowFw1y4UGK1YJTCQdWH4gBBJL2IP2IduHCKKsx87bKfai86Lqfpg88F-x7MRXnoaid4k3YEBhVOBKgVLAfUp_Fgra6tzxhjnqCHTR-SlQe-GBxcnFqj06JmgJ1wTomJGOuiytXaKWruLQtriblbi6GI-HA0ex5rJzXYuVp0bFKwpYn9W1YY8wsCvJkWZamhQ3dlbd-qQwYKl3geTTUvluqYL7PAvvcmkTjUx88-2ke90Xckn4USFVur8WkXGRf-w6adV2NywyTl7EzGcrB-DmyRO_w45ZzCnhAe6GDzsDQU6QcRp-tLJzsf1qYwWH2GR0ptf0hVJ21FD_Z22o0uZbcpsxRbrmb50IDTz_jfRwHexPrxfmE-_gB_0eaMZEoXrr8nKLQ6G-YWBMfm2clY7n8TY6SBGa6rIfk58wxwqBd5SLUW82cdXzvYlpObDDviW5qa1cN0opJFvnlHC-3WkIVCsJGYeXByTLaWOGhnDcoZgm0l6La0-bPp5p1BzamiH_eRyRZYHYbMILWSnJefeWmWf_6FV6G_CAERlrvtCVOodRQfruzTBPm9nR7U0D4Qf1qMCB8ZQcke9LCMDFlfjxadlXvRNnA4VgmS_bO3P5sE6E3QjSblimgE40INAWJHhP00fipYBxJaMivpT3x0Onx75NHR0nUFP2wt9PVRFRto2k0Q_GHsIoV5sMDUGfJ2CREKh3qkEOBJPYdA2-generated

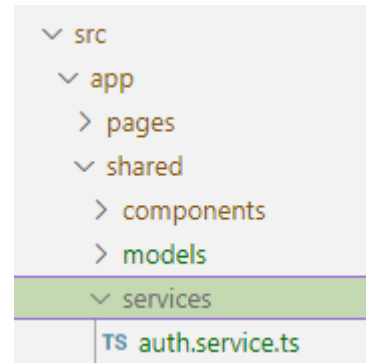


Crear servicio de Autenticación

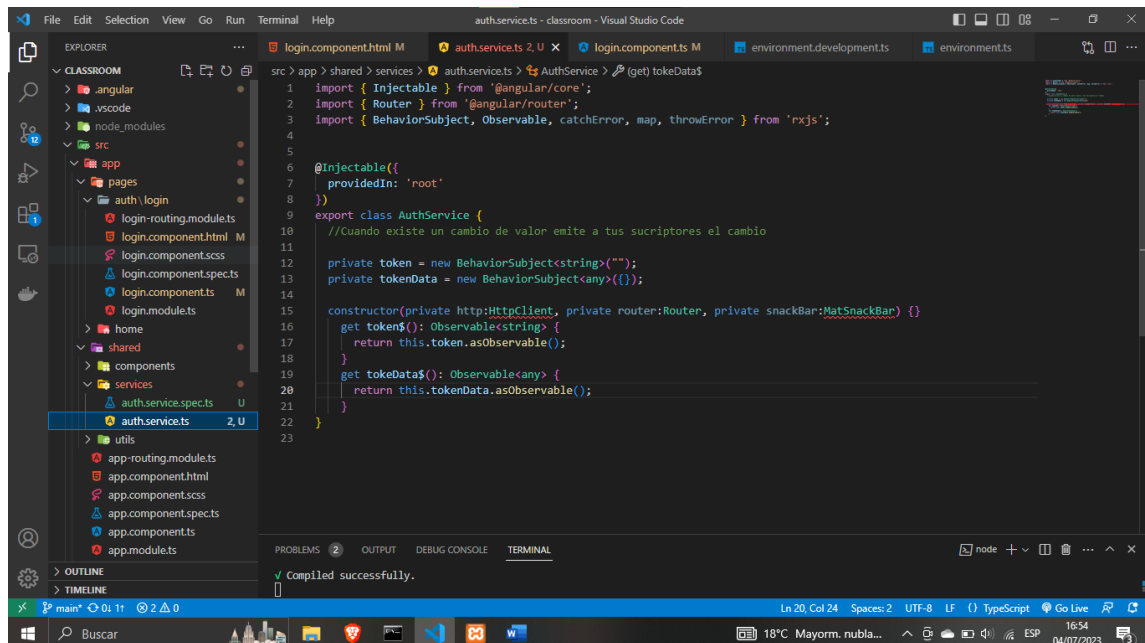
Primeramente ,agregar módulo [MatSnackBarModule](#) al archivo **material.module.ts** para usar el servicio de mensajes; modificar el archivo **auth.service.ts**

Abrir terminal y ejecutar el siguiente comando para crear el servicio de autenticación: **ng g s shared/services/auth**

De hecho la ruta la definen de acuerdo a su experiencia



Guía de Angular



Agregamos la siguiente instrucción para ocupar elementos a utilizar para el manejo del Token y datos de usuario

```
import { BehaviorSubject, Observable, catchError, map, throwError }  
from 'rxjs';
```

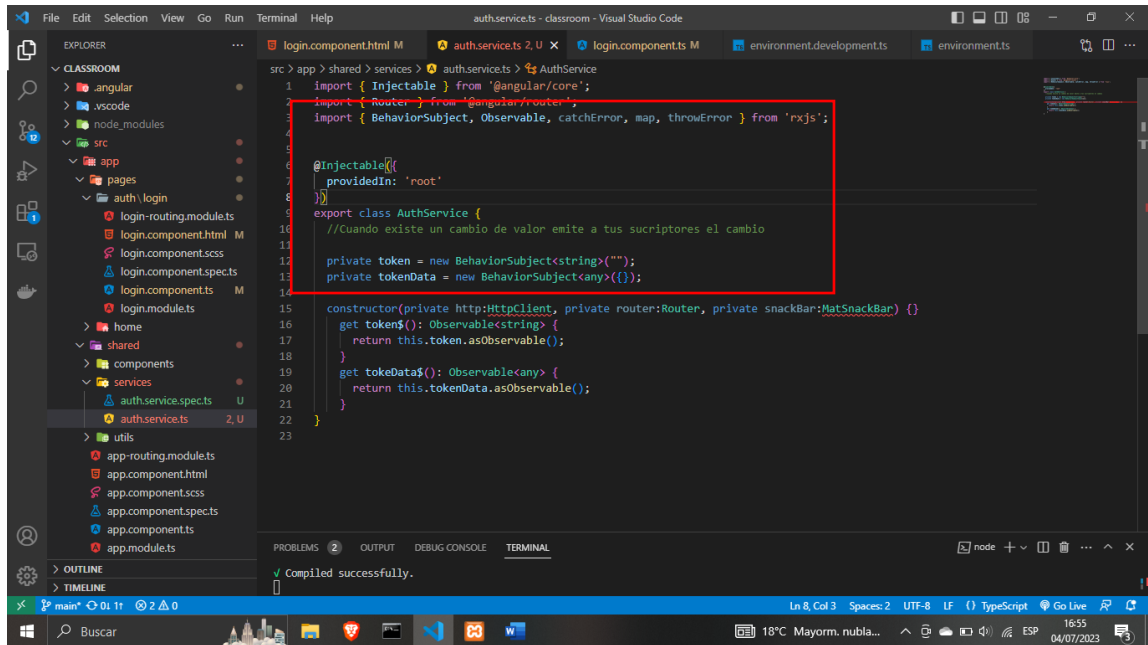
Agregar las propiedades a la clase AuthService:

token Es el token encriptado

tokenData Datos de usuario encriptado qué es el Payload

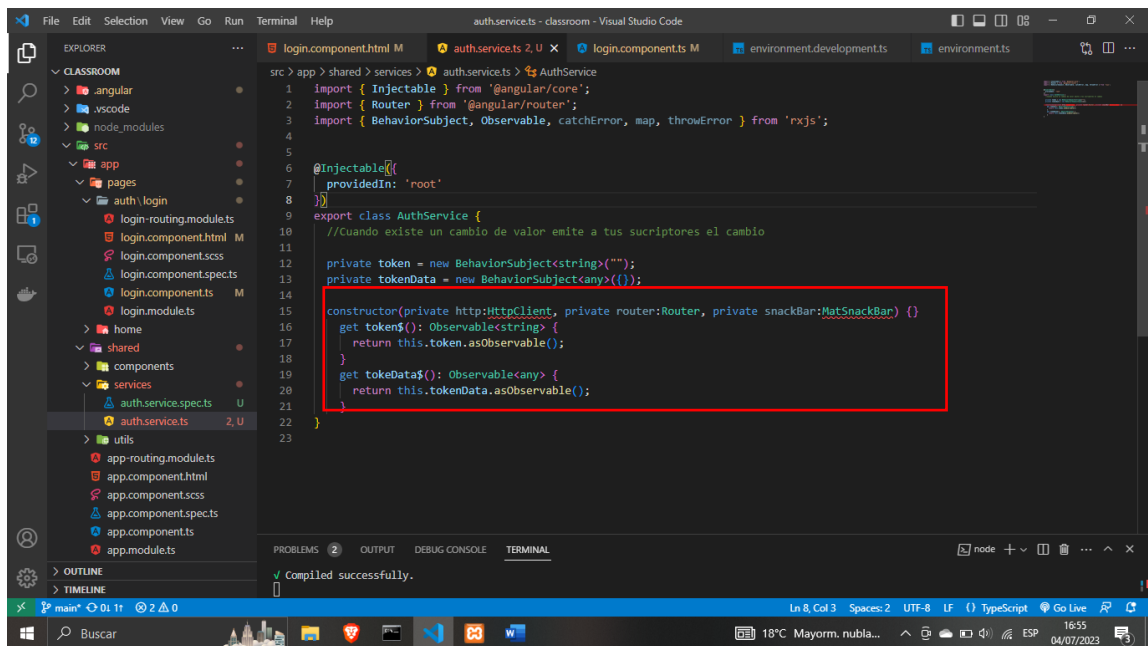
```
@Injectable({  
  providedIn: 'root'  
})  
export class AuthService {  
  //Cuando existe un cambio de valor emite a sus suscriptores el cambio  
  private token = new BehaviorSubject<string>("");  
  private tokenData = new BehaviorSubject<any>({});  
  
  constructor(private http:HttpClient, private router:Router, private snackBar:MatSnackBar) {}  
}
```

Guía de Angular



Agregar las siguientes instrucciones al archivo **auth.service.ts** con el objetivo de definir las propiedades **token** y **tokenData** como Observables y permitir suscripciones y saber de sus valores

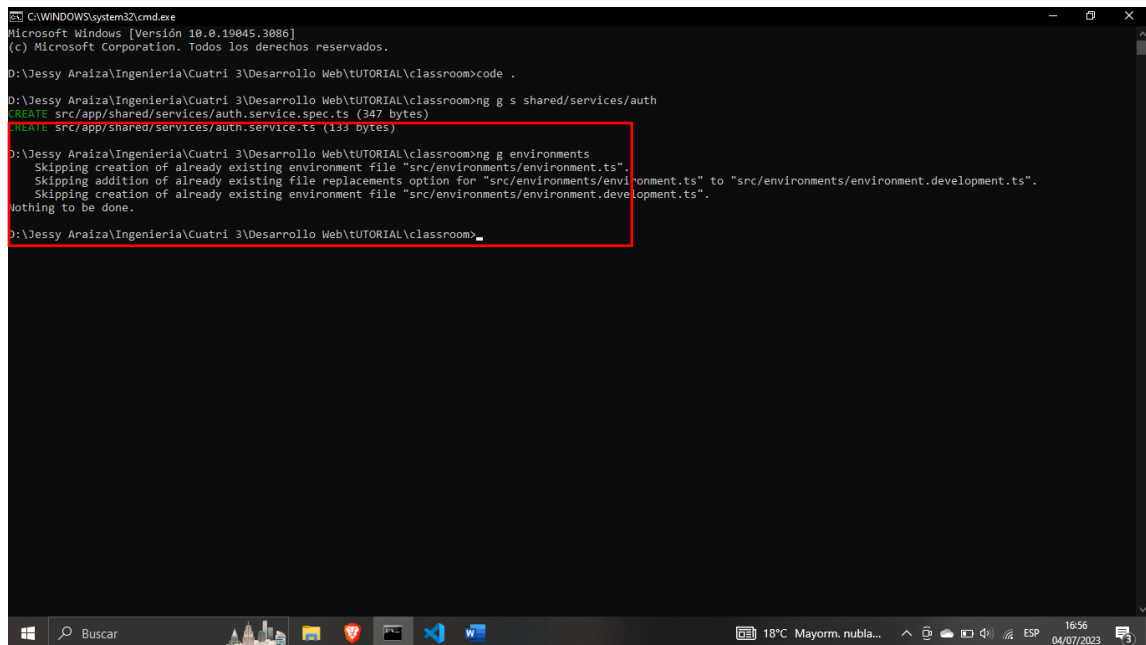
```
15 constructor(private http: HttpClient, private router: Router, private snackBar: MatSnackBar) { }
16
17 get token$(): Observable<string> {
18   return this.token.asObservable();
19 }
20
21 get tokenData$(): Observable<any> {
22   return this.tokenData.asObservable();
23 }
```



el utilizar environments en tu aplicación es con el objetivo de utilizar API KEY para entornos de pruebas y de producción durante el desarrollo de la aplicación.

En la terminal ejecutar el siguiente comando: **ng g environments**

Guía de Angular



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.3086]
(c) Microsoft Corporation. Todos los derechos reservados.

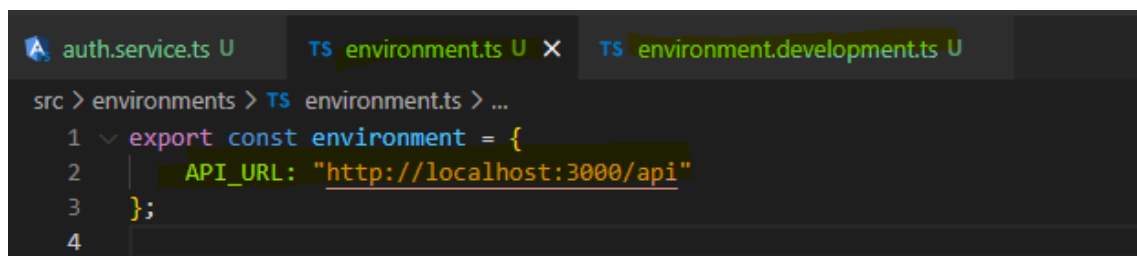
D:\Jessy Araiza\Ingenieria\CuatrI 3\Desarrollo Web\tUTORIAL\classroom>code .

D:\Jessy Araiza\Ingenieria\CuatrI 3\Desarrollo Web\tUTORIAL\classroom>ng g s shared/services/auth
CREATE src/app/shared/services/auth.service.spec.ts (347 bytes)
CREATE src/app/shared/services/auth.service.ts (133 bytes)

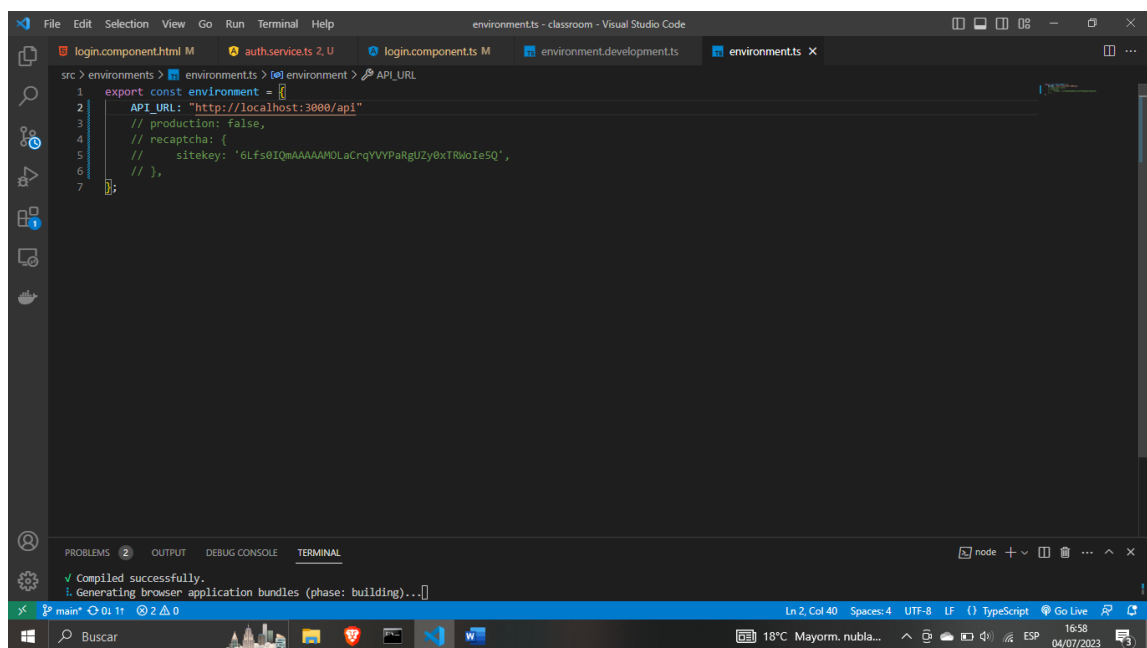
D:\Jessy Araiza\Ingenieria\CuatrI 3\Desarrollo Web\tUTORIAL\classroom>ng g environments
Skipping creation of already existing environment file "src/environments/environment.ts".
Skipping addition of already existing file replacements option for "src/environments/environment.ts" to "src/environments/environment.development.ts".
Skipping creation of already existing environment file "src/environments/environment.development.ts".
Nothing to be done.

D:\Jessy Araiza\Ingenieria\CuatrI 3\Desarrollo Web\tUTORIAL\classroom>
```

Modificar los archivos environments de la siguiente forma



```
src > environments > TS environment.ts > ...
1 export const environment = {
2   API_URL: "http://localhost:3000/api"
3 };
4
```



```
File Edit Selection View Go Run Terminal Help
environments - classroom - Visual Studio Code

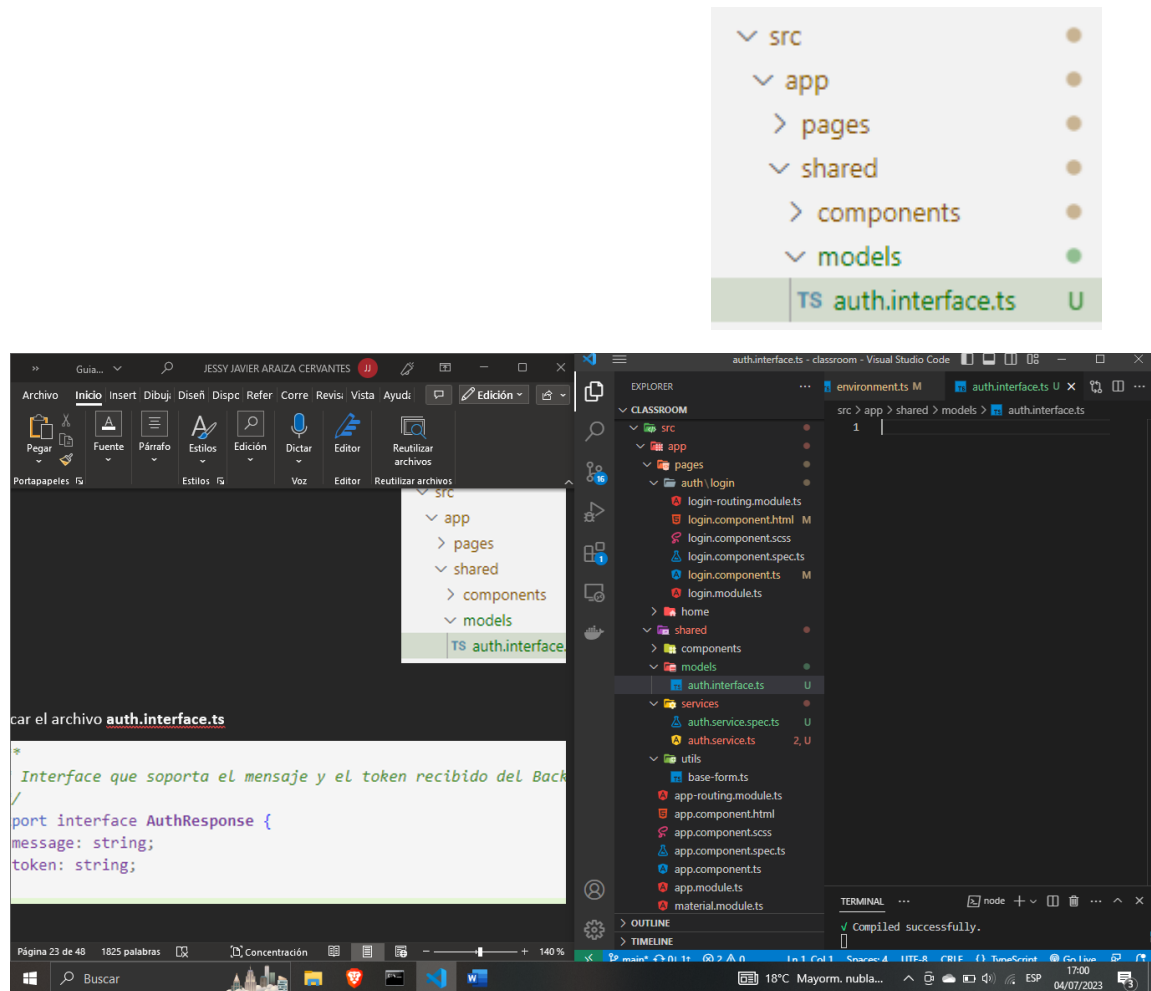
src > environments > TS environment.ts > ...
1 export const environment = {
2   API_URL: "http://localhost:3000/api"
3   // production: false,
4   // recaptcha: {
5   //   sitekey: '6Lfs0IQmAAAAAMOLaCrqYVYPaRgUZY0xTRWoIe5Q',
6   // },
7 };

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL
v Compiled successfully.
! Generating browser application bundles (phase: building)...

Ln 2, Col 40, Spaces: 4, UTF-8, LF, TypeScript, Go Live
```

En carpeta **shared/models** agregar un archivo con el nombre **auth.interface.ts**, en caso que no exista se deberá de crear

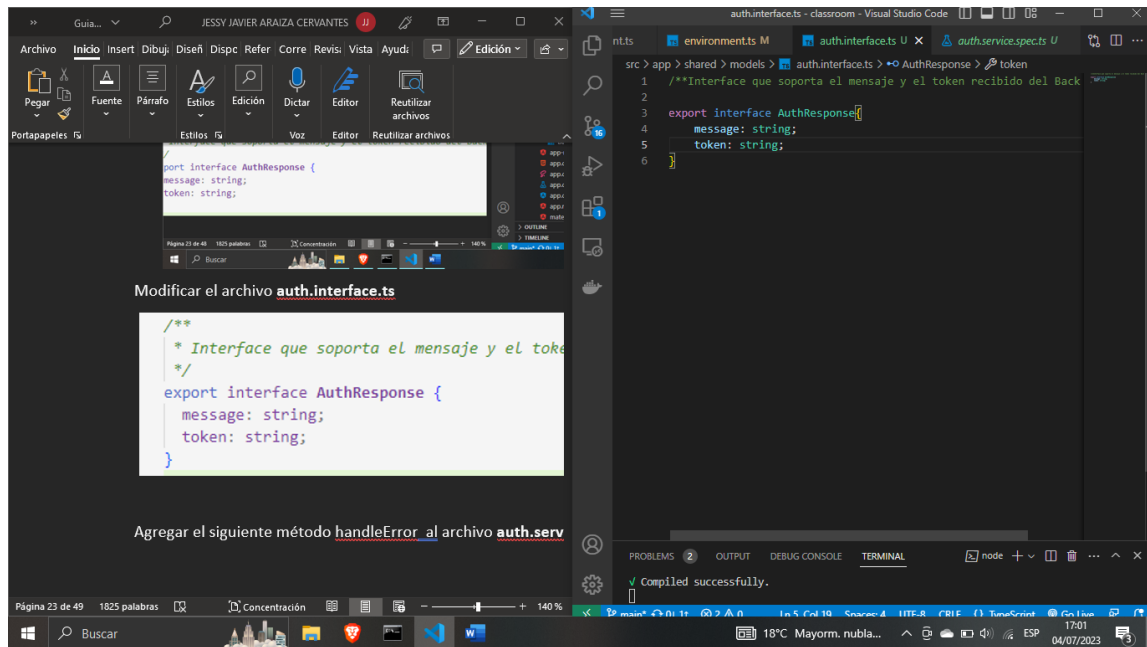
Guía de Angular



Modificar el archivo `auth.interface.ts`

```
/**
 * Interface que soporta el mensaje y el token recibido del Backend
 */
export interface AuthResponse {
  message: string;
  token: string;
}
```

Guía de Angular



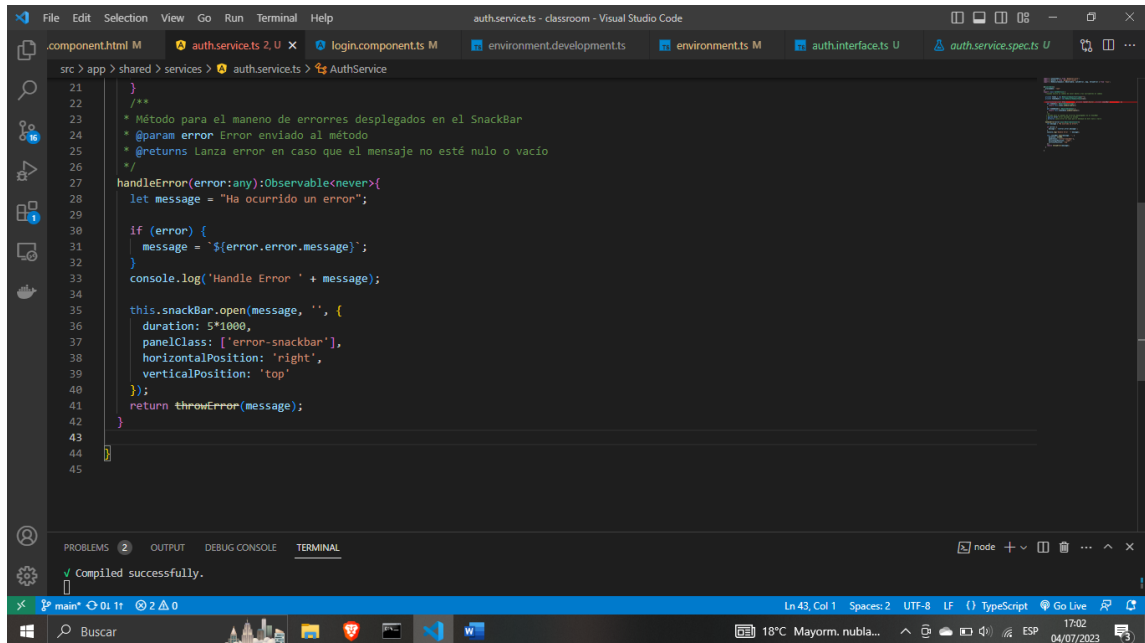
Agregar el siguiente método `handleError` al archivo `auth.service.ts`

```
/**
 * Método para el manejo de errores desplegados en el SnackBar
 * @param error Error enviado al método
 * @returns Lanza error en caso que el mensaje no esté nulo o
vacío
 */
handleError(error:any):Observable<never>{
    let message = "Ha ocurrido un error";

    if (error) {
        message = `${error.error.message}`;
    }
    console.log('Handle Error ' + message);

    this.snackBar.open(message, '', {
        duration: 5*1000,
        panelClass: ['error-snackbar'],
        horizontalPosition: 'right',
        verticalPosition: 'top'
    });
    return throwError(message);
}
```

Guía de Angular



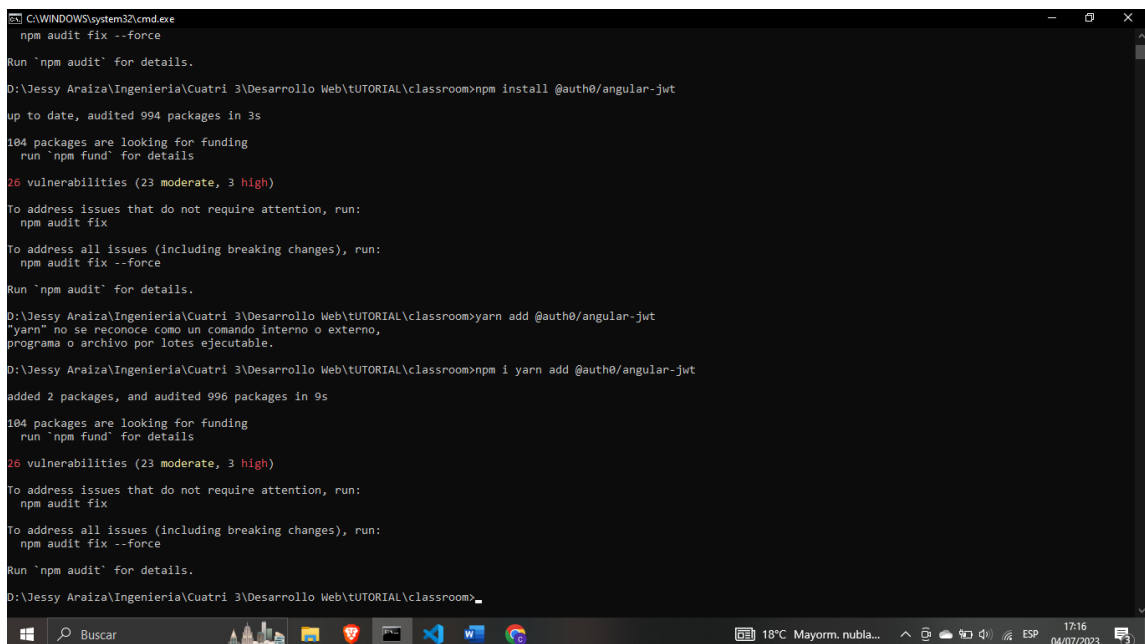
Guardado de Sesión

Instalar el módulo [@auth0/angular-jwt](#)

Installation

```
# installation with npm
npm install @auth0/angular-jwt

# installation with yarn
yarn add @auth0/angular-jwt
```



Importar clase JwtHelperService

```
import { JwtHelperService } from '@auth0/angular-jwt';
```

Agregar una variable llamada helper en auth.service.ts

```
const helper = new JwtHelperService();
@Injectable({
  providedIn: 'root'
})
export class AuthService {
```

LocalStorage

El LocalStorage permite almacenar datos en el navegador web. Y que estos persistan y estén disponibles durante la navegación en la aplicación web, hasta que esta información sea borrada del navegador.

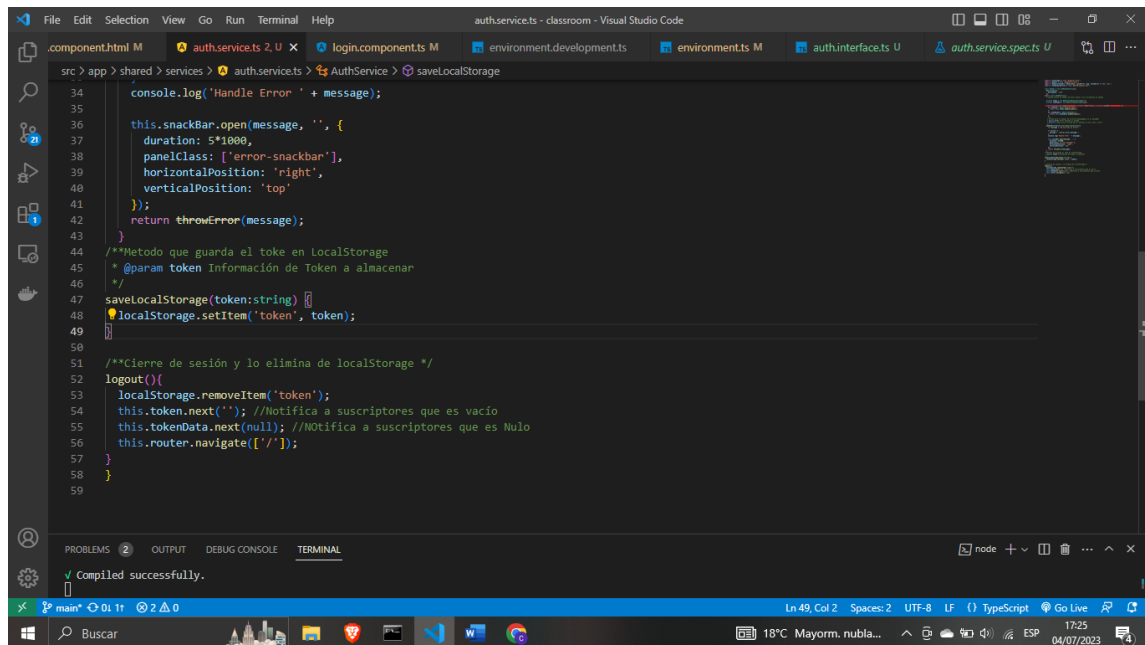
Agregar el método **saveLocalStorage** en archivo **auth.service.ts**

```
/**
 * Método que guarda el token en LocalStorage
 * @param token Información de Token a almacenar
 */
saveLocalStorage(token:string) {
  localStorage.setItem('token', token);
}
```

Agregar el método **logout** en archivo **auth.service.ts**

```
/**
 * Cierre de sesión y lo elimina de LocalStorage
 */
logout() {
  localStorage.removeItem('token');
  this.token.next(''); //Notifica a suscriptores que es vacío
  this.tokenData.next(null); //Notifica a suscriptores que es Nulo
  this.router.navigate(['/']);
}
```

Guía de Angular



Agregar el método **checklogin** en archivo **auth.service.ts**

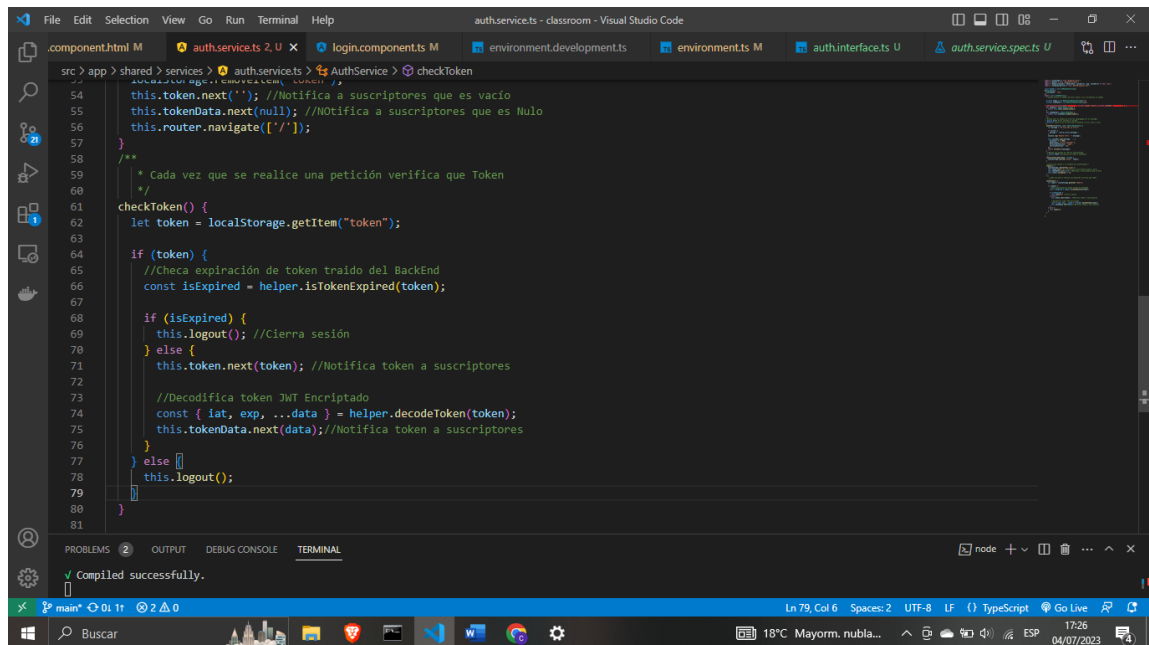
```
/**
 * Cada vez que se realice una petición verifica que Token
 */
checkToken() {
  let token = localStorage.getItem("token");

  if(token) {
    //Checa expiración de token traído del Backend
    const isExpired = helper.isTokenExpired(token);

    if(isExpired) {
      this.logout(); //Cierra sesión
    } else {
      this.token.next(token); //Notifica token a suscriptores

      //Decodifica token JWT Encriptado
      const {iat, exp, ...data} = helper.decodeToken(token);
      this.tokenData.next(data); //Notifica token a suscriptores
    }
  } else {
    this.logout();
  }
}
```

Guía de Angular



Modificar método login de la siguiente forma:

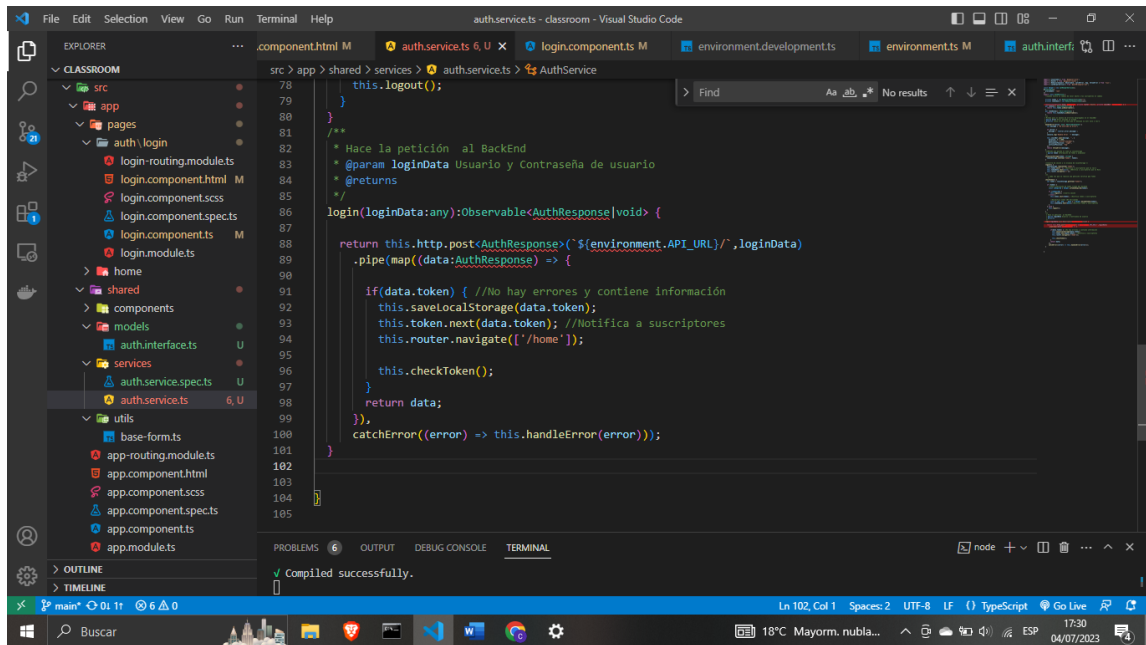
```
/**
 * Hace la petición al BackEnd
 * @param loginData Usuario y Contraseña de usuario
 * @returns
 */
login(loginData:any):Observable<AuthResponse|void> {

  return
  this.http.post<AuthResponse>(`${environment.API_URL}/`, loginData)
    .pipe(map((data:AuthResponse) => {

      if(data.token) { //No hay errores y contiene información
        this.saveLocalStorage(data.token);
        this.token.next(data.token); //Notifica a suscriptores
        this.router.navigate(['/home']);

        this.checkToken();
      }
      return data;
    })),
    catchError((error) => this.handleError(error)));
}
```

Guía de Angular



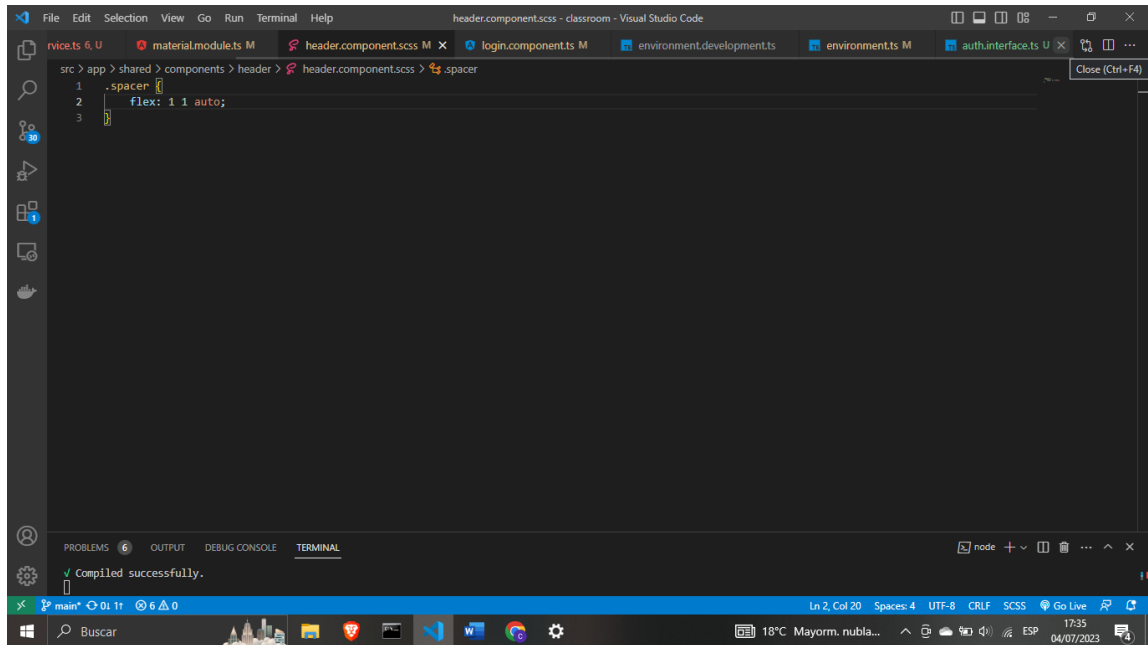
Modificar el archivo **material.module.ts** y agregar **MatToolbarModule**

```
8 import { MatToolbarModule } from '@angular/material/toolbar';
9
10 const myModules :any = [
11   MatButtonModule,
12   MatCardModule,
13   MatInputModule,
14   MatIconModule,
15   MatGridListModule,
16   MatSnackBarModule,
17   MatToolbarModule
18 ];
```

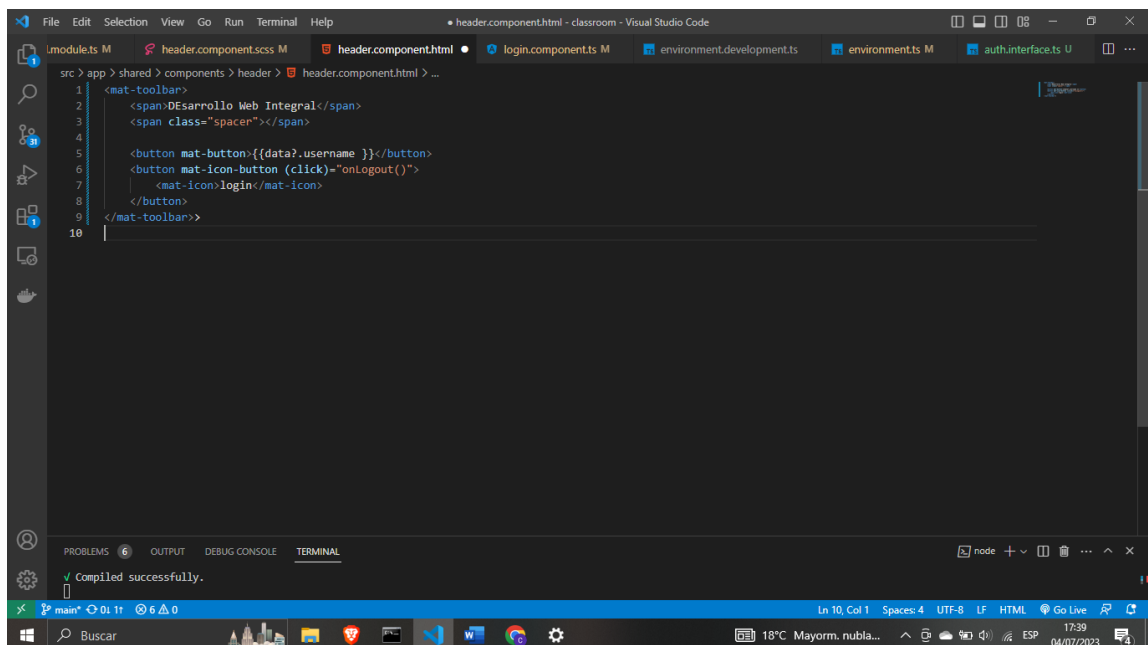
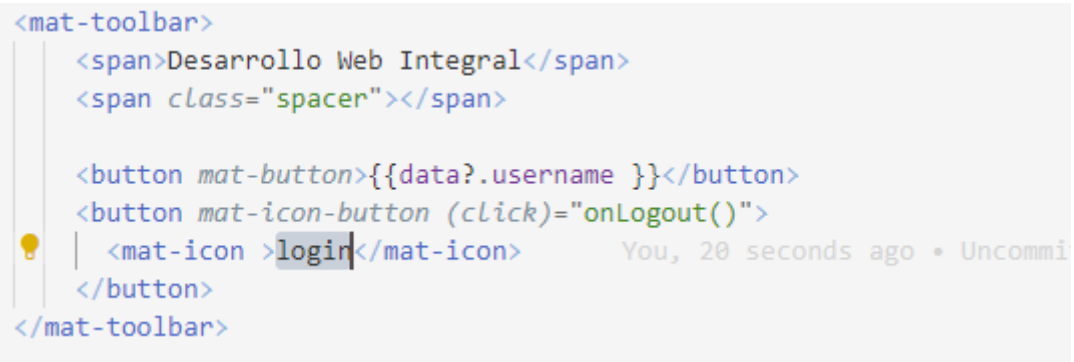
Modificar el archivo **header.component.scss**

```
1 .spacer {
2   flex: 1 1 auto;
3 }
```


Guía de Angular



Modificar el archivo `header.component.html`



Modificar el archivo `header.component.ts`

```

8   export class HeaderComponent implements OnInit {
9
10      constructor() { }
11
12      ngOnInit(): void { }
13
14      onLogout() { }
15
16  }

```

Modificar el archivo **auth.service.ts** y agregar al constructor la siguiente línea:

```
this.checkToken();
```

Modificar el archivo header.component.ts

```

export class HeaderComponent implements OnInit {

  data: any = {}; //Datos de Usuario

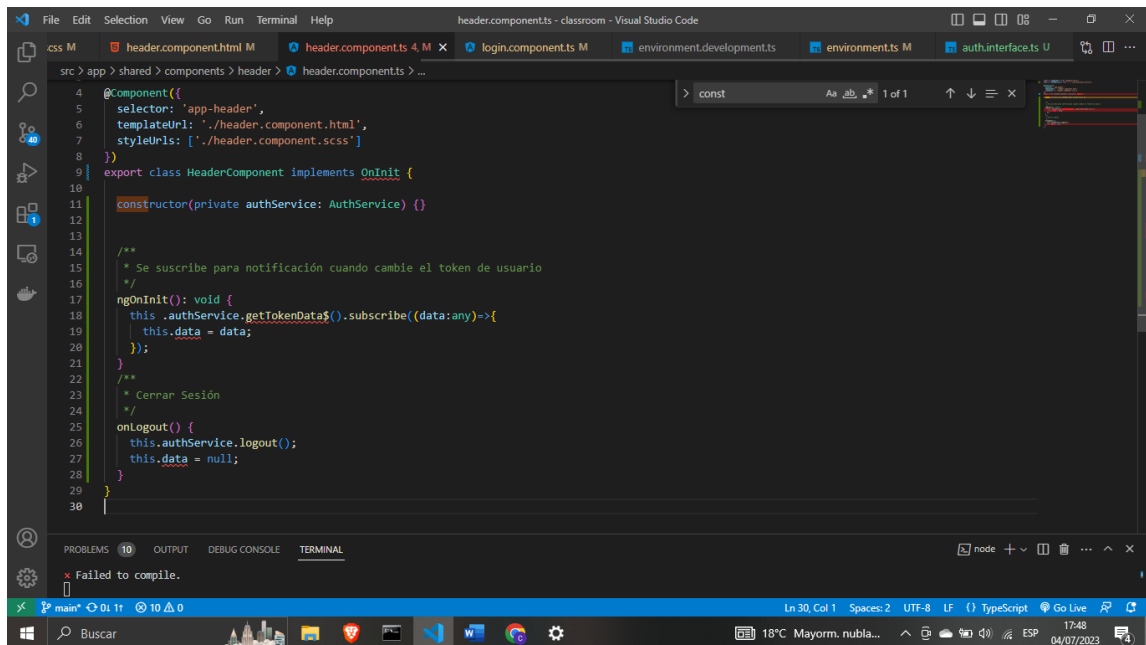
  constructor(private authService: AuthService) {
  }

  /**
   * Se suscribe para notificación cuando cambie el token de usuario.
   */
  ngOnInit(): void {
    this.authService.getTokenData$.subscribe((data:any)=> {
      this.data = data;
    });
  }

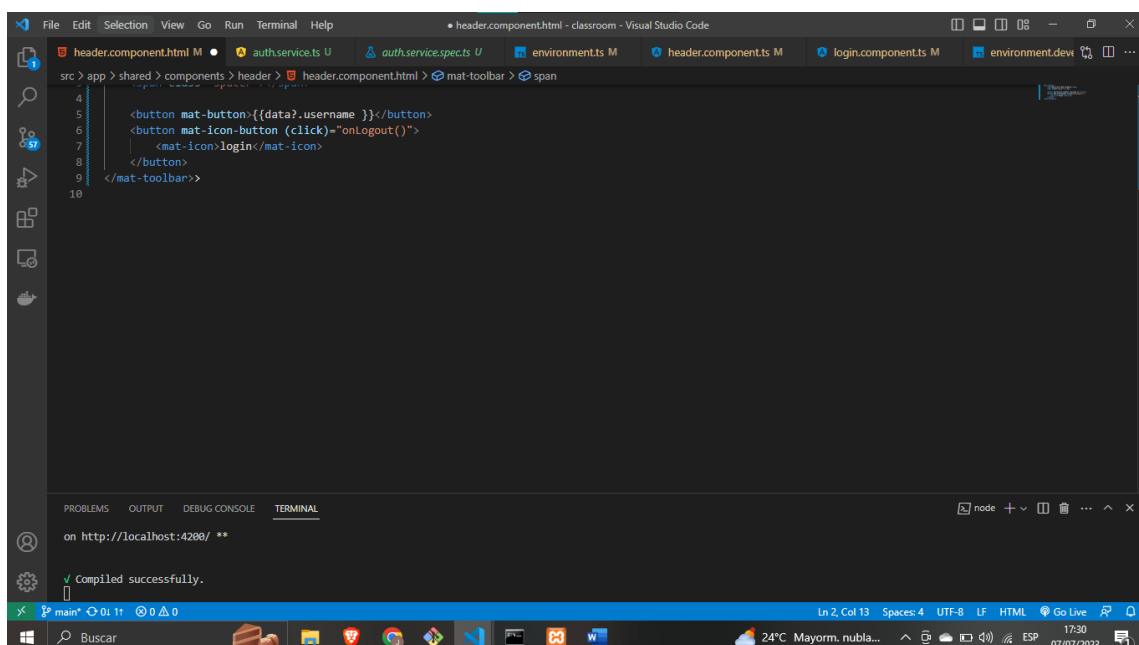
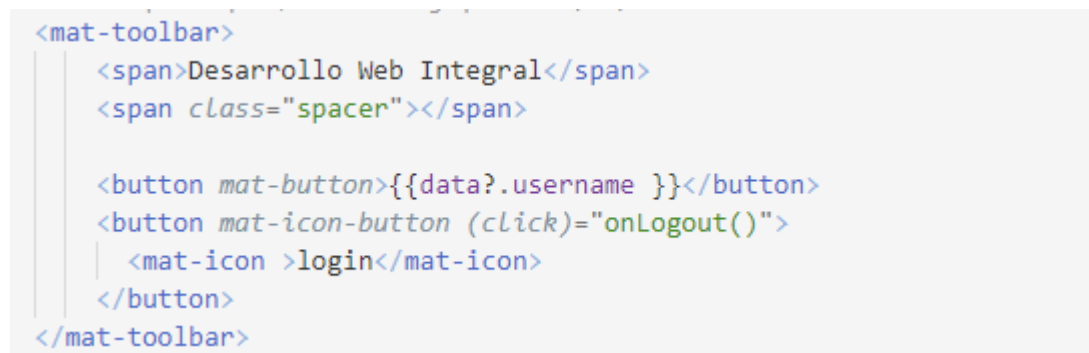
  /**
   * Cerrar sesión
   */
  onLogout() {
    this.authService.logout();
    this.data = null;
  }
}

```

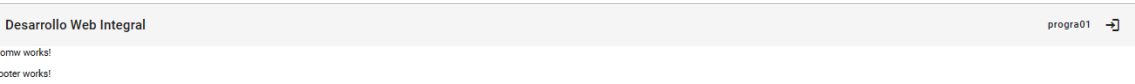
Guía de Angular



Modificar el archivo `header.component.html`

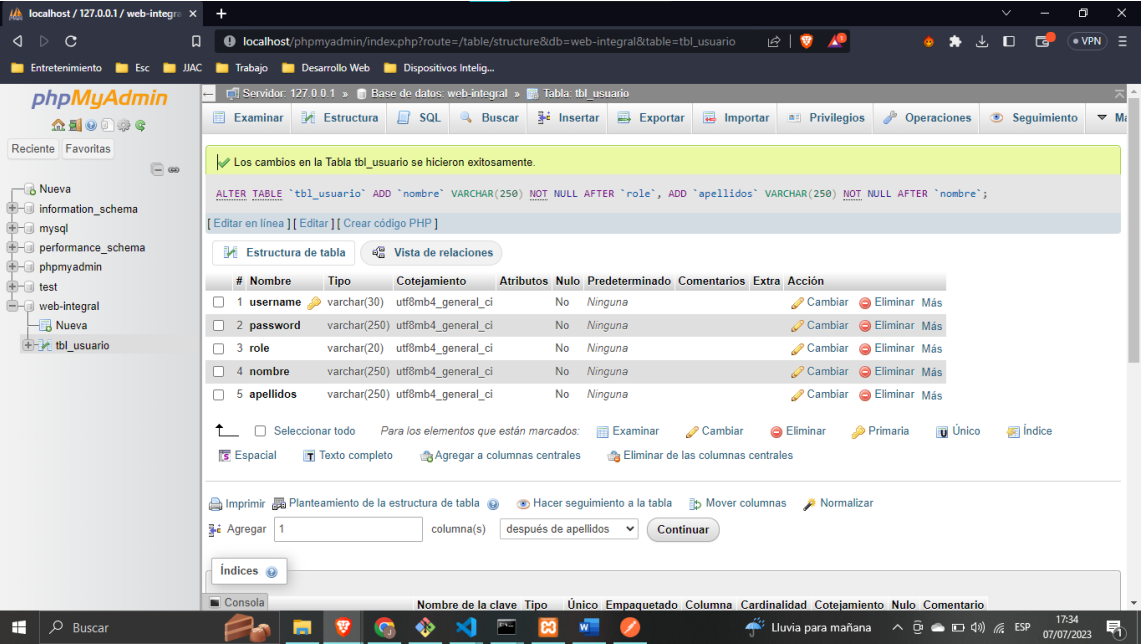


Al finalizar se mostrará de la siguiente manera



Desafío

- Agregar más campos a la tabla tbl_usuario: nombre, apellidos.



- Una vez que está logueado imprimir su nombre completo

