

Temas puntuales de la sección

El objetivo de la sección es trabajar con un proceso de autenticación por token tradicional, veremos temas como:

- Validar formularios
- Tokens
- LocalStorage
- Borrar tokens
- Caducidad de tokens
- Creación de usuarios
- Posteos
- Firebase REST API

También trabajaremos con un diseño elaborado elegante para nuestro login, y poder reconstruir un proyecto usando un repositorio externo.

2. Inicio del proyecto: nos lo bajamos desde el curso de moodle y lo abrimos.

Instalamos las dependencias de node:

```
$ npm install
```

```
iMac-de-Jose:~ jsersan$ cd /Users/jsersan/Desktop/htdocs/appLoginFireBase
iMac-de-Jose:appLoginFireBase jsersan$ npm install
npm WARN deprecated core-js@2.6.11: core-js@<3 is no longer maintained and not recommended for usage due to the number of issues. Please, upgrade your dependencies to the actual version of core-js@3.
) . fetchMetadata: sill pacote version manifest for sockjs@0
```

Una vez instaladas las dependencias lanzamos la app:

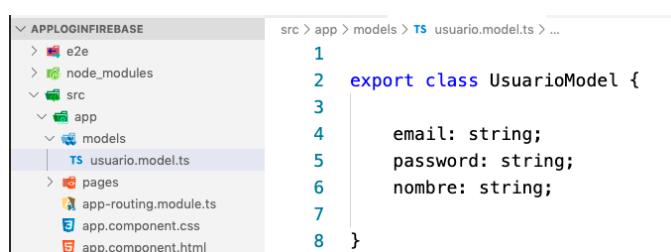
CREAR NUEVA CUENTA



La abrimos en Visual Code:



Creo dentro la carpeta *app*, otra carpeta *models* y dentro el archivo **usuario.model.ts** creamos el modelo de datos:



2.- Conectar con el modelo de usuario:

Nos vamos al fichero **registro.component.ts** para crear una nueva instancia del **usuario.model**:

```
registro.component.ts ×
src > app > pages > registro > registro.component.ts > RegistroComponent > ngOnInit
1 import { Component, OnInit } from '@angular/core';
2 import { UsuarioModel } from '../../../../../models/usuario.model';
3
4 @Component({
5   selector: 'app-registro',
6   templateUrl: './registro.component.html',
7   styleUrls: ['./registro.component.css']
8 })
9 export class RegistroComponent implements OnInit {
10
11   usuario: UsuarioModel;
12
13   constructor() { }
14
15   ngOnInit() {
16     this.usuario = new UsuarioModel();
17
18     this.usuario.email = 'jsersan@gmail.com';
19     this.usuario.nombre = 'Txema Serrano';
20     this.usuario.password = '123456';
21   }
22
23 }
24 }
```

En **registro.component.html** debemos conectar el formulario de registro con una instancia del modelo de usuario:

```
<!-- <span class="text-danger">El correo es obligatorio</span> -->
<div class="wrap-input100 m-b-16">
  <input class="input100"
    type="text"
    name="email"
    [(ngModel)]="usuario.email"
    placeholder="Email">

  <span class="focus-input100"></span>
</div>

<!-- <span class="text-danger">El nombre es obligatorio</span> -->
<div class="wrap-input100 m-b-16">
  <input class="input100"
    type="text"
    name="nombre"
    [(ngModel)]="usuario.nombre"
    placeholder="Nombre y apellidos">

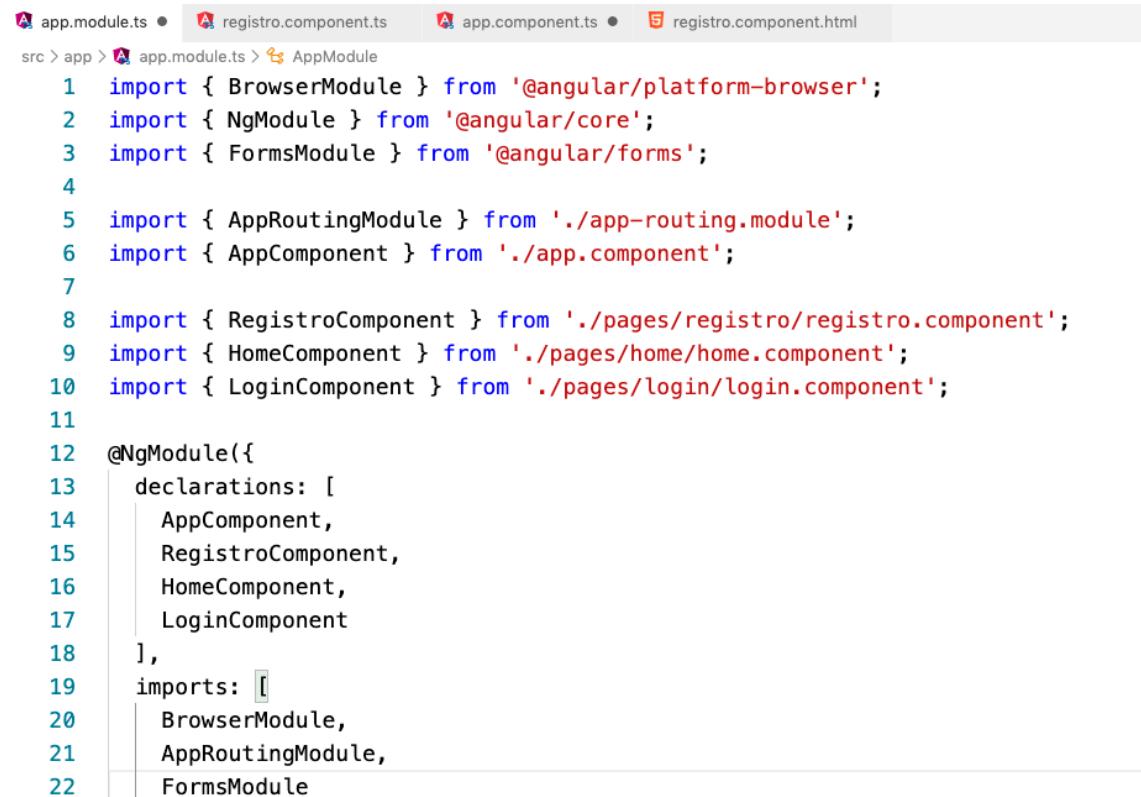
  <span class="focus-input100"></span>
</div>

<!-- <span class="text-danger">La contraseña debe de ser más de 6
letras</span> -->
<div class="wrap-input100 m-b-16" data-validate = "Password is
required">
  <input class="input100"
    type="password"
    name="pass"
    [(ngModel)]="usuario.password"
    placeholder="Password">
  <span class="focus-input100"></span>
</div>
```

Guardamos:

```
✖ Uncaught Error: Template parse errors:
Can't bind to 'ngModel' since it isn't a known property of 'input'. (" class="wrap-input100 m-b-16">
<input class="input100" type="text" name="email" [ERROR ->] [(ngModel)]="usuario.email" placeholder="Email">
<span class="focus-input100"></>"): ng:/// AppModule/RegistroComponent.html@11:69
Can't bind to 'ngModel' since it isn't a known property of 'input'. (" class="wrap-input100 m-b-16">
<input class="input100" type="text" name="nombre" [ERROR ->] [(ngModel)]="usuario.nombre" placeholder="Nombre y apellidos">
<span class="focus">"): ng:/// AppModule/RegistroComponent.html@18:70
Can't bind to 'ngModel' since it isn't a known property of 'input'. (" at="Password is required">
<input class="input100" type="password" name="pass" [ERROR ->] [(ngModel)]="usuario.password" placeholder="Password">
<span class="focus-input100">"): ng:/// AppModule/RegistroComponent.html@25:72
at cuntasError (controler.js:242)
```

Para que no den errores los ngModels debemos importar @angular/forms en app.module.ts.



```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { FormsModule } from '@angular/forms';
4
5 import { AppRoutingModule } from './app-routing.module';
6 import { AppComponent } from './app.component';
7
8 import { RegistroComponent } from './pages/registro/registro.component';
9 import { HomeComponent } from './pages/home/home.component';
10 import { LoginComponent } from './pages/login/login.component';
11
12 @NgModule({
13   declarations: [
14     AppComponent,
15     RegistroComponent,
16     HomeComponent,
17     LoginComponent
18   ],
19   imports: [
20     BrowserModule,
21     AppRoutingModule,
22     FormsModule
23   ]
24 })
```

Los errores desaparecen:

CREAR NUEVA CUENTA



jsersan@gmail.com

Txema Serrano

.....

Recordar mi usuario [¿Ya tienes cuenta? / Ingresar](#)

CREAR CUENTA

Para capturar la información del formulario, en **registro.component.html**:

```

app.module.ts    registro.component.ts    app.component.ts    registro.component.html
src > app > pages > registro > registro.component.html > div.limiter > div.container-login100 > div.wr
1  <div class="limiter">
2      <div class="container-login100">
3          <div class="wrap-login100 p-t-50 p-b-90">
4              <form (ngSubmit)="onSubmit()">
5                  class="login100-form validate-form flex"
6
7                  <span class="login100-form-title p-b-51">
8                      Crear nueva cuenta
9                  </span>

```

Creo esta función en el typeScript **registro .component.ts**:

```

15  ngOnInit() {
16      this.usuario = new UsuarioModel();
17
18      this.usuario.email = 'jsersan@gmail.com';
19      this.usuario.nombre = 'Txema Serrano';
20      this.usuario.password = '123456';
21  }
22
23  onSubmit() {
24      console.log('Formulario enviado');
25      console.log(this.usuario);
26  }

```

Guardo los cambios y pruebo el botón:

```

Angular is running in the development mode. Call enableProdMode() to enable the production mode.
Formulario enviado
▼ UsuarioModel {email: "jsersan@gmail.com", nombre: "Txema Serrano", password: "123456"} ⓘ
  email: "jsersan@gmail.com"
  nombre: "Txema Serrano"
  password: "123456"
  ► __proto__: Object

```

3.- Validar la información antes de enviarla al servidor.

Debemos validar la información de cada uno de los campos. Nombre, email y password de registro.component.html. Para el email, cambiamos el tipos al email (aparecerá @):

```
<!-- <span class="text-danger">El correo es
obligatorio</span> -->
<div class="wrap-input100 m-b-16">
  <input class="input100"
    type="email"
    name="email"
    [(ngModel)]="usuario.email"
    required
    email
    placeholder="Email">

  <span class="focus-input100"></span>
</div>
```

Cuando no especifico el tipo de validación, asumirá la que está por defecto y será por template. Para ello le asigno una variable #f que es una referencia a todo el formulario:

```
<form (ngSubmit)="onSubmit( f )" 
#f = "ngForm"
class="login100-form validate-form flex-sb flex-w">
```

Ahora en **registro.component.ts**:

```
3 import { NgForm } from '@angular/forms';
4
5 @Component({
6   selector: 'app-registro',
7   templateUrl: './registro.component.html',
8   styleUrls: ['./registro.component.css']
9 })
10 export class RegistroComponent implements OnInit {
11
12   usuario: UsuarioModel;
13
14   constructor() { }
15
16   ngOnInit() {
17     this.usuario = new UsuarioModel();
18
19     this.usuario.email = 'jsersan@gmail.com';
20     this.usuario.nombre = 'Txema Serrano';
21     this.usuario.password = '123456';
22   }
23
24   onSubmit(form: NgForm) {
25     console.log('Formulario enviado');
26     console.log(this.usuario);
27     console.log(form);
28 }
```

Guardamos cambio, borramos los valores los campos y le damos al botón Crear Cuenta:

CREAR NUEVA CUENTA

Email

Nombre y apellidos

Password

Recordar mi usuario [¿Ya tienes cuenta? / Ingresar](#)

CREAR CUENTA



Miro el objeto *ngForm*:

```

<div>
  <div>Email</div>
  <div>Nombre y apellidos</div>
  <div>Password</div>
  <div><input type="checkbox" checked=""> Recordar mi usuario <a href="#">¿Ya tienes cuenta? / Ingresar</div>
  <div><b>Crear cuenta</b></div>
</div>

```

Para comprobar el estado del formulario. Vemos que es falso porque hay campos *obligatorios* que están vacíos:

```
  - form: FormGroup
    parent: ...
    valid: false
    ...
    ...
```

Si ponemos una dirección válida para email:

```
jsersan@gmail.com
```

En la consola:

```
  - NgForm {submitted: true, _directives: Array(3), ngSubmit: EventEmitter, form: FormGroup} ⓘ
    formDirective: ...
    control: ...
    path: ...
    controls: ...
    value: ...
    valid: true
    invalid: ...
    pending: ...
    disabled: ...
```

Añadimos las validaciones al resto de los campos: por ejemplo ponemos required y minlength:

```
<!-- <span class="text-danger">El nombre es
obligatorio</span> -->
<div class="wrap-input100 m-b-16">
  <input class="input100"
    type="text"
    name="nombre"
    [(ngModel)]="usuario.nombre"
    required
    minlength="2"
    placeholder="Nombre y apellidos">

  <span class="focus-input100"></span>
</div>

<!-- <span class="text-danger">La contraseña debe de
ser más de 6 letras</span> -->
<div class="wrap-input100 m-b-16"
  data-validate="Password is required">
  <input class="input100"
    type="password"
    name="password"
    [(ngModel)]="usuario.password"
    minlength="6"
    required
    placeholder="Password">
  <span class="focus-input100"></span>
</div>
```

En el typeScript **registro.component.ts**:

```

24  onSubmit(form: NgForm) {
25
26    if ( form.invalid ) {return; } // formulario inválido no hace nada
27    console.log('Formulario enviado');
28    console.log(this.usuario);
29    console.log(form);
30  }

```

Probamos y venos que no hace nada hasta que rellenemos los campos del formulario.

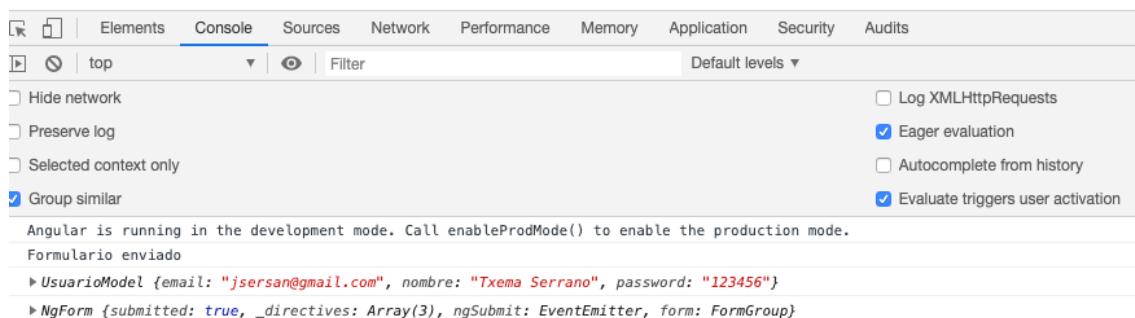
CREAR NUEVA CUENTA

jsersan@gmail.com

Txema Serrano

Recordar mi usuario [¿Ya tienes cuenta?](#) / [Ingresar](#)

CREAR CUENTA



4.-Mostrar errores por pantalla.

En **registro.components.ts**, eliminamos la inicialización de los campos. Obtenemos esta salida:

CREAR NUEVA CUENTA

Email

Nombre y apellidos

Password

Recordar mi usuario [¿Ya tienes cuenta? / Ingresar](#)

CREAR CUENTA

Las validaciones las haremos en el HTML por template.

```
<span class="text-danger">El correo es obligatorio</span>
<div class="wrap-input100 m-b-16">
  <input class="input100" type="email" name="email" [
    (ngModel)]="usuario.email" required email
    placeholder="Email">

  <span class="focus-input100"></span>
</div>
```

Sale esto:

El correo es obligatorio

Email

Vamos a sacar este mensaje cuando vayamos a validar realmente y ha ya errores.

```
<span *ngIf="f.controls['email'].errors" class="text-danger">El correo es
obligatorio</span>
<div class="wrap-input100 m-b-16">
  <input class="input100" type="email" name="email" [(ngModel)]
    ="usuario.email" required email placeholder="Email">

  <span class="focus-input100"></span>
</div>
```

Está evaluando algo que todavía no se ha enviado por *submit*:

```
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
✖ ▶ ERROR TypeError: Cannot read property 'errors' of undefined
    at Object.eval [as updateDirectives] (RegistroComponent.html:10)
    at Object.debugUpdateDirectives [as updateDirectives] (core.js:23911)
    at checkAndUpdateView (core.js:23307)
    at callViewAction (core.js:23548)
    at execComponentViewsAction (core.js:23490)
    at checkAndUpdateView (core.js:23313)
    at callViewAction (core.js:23548)
    at execEmbeddedViewsAction (core.js:23511)
    at checkAndUpdateView (core.js:23308)
    at callViewAction (core.js:23548)
✖ ▶ ERROR CONTEXT ▶ DebugContext_ {view: {...}, nodeIndex: 11, nodeDef: {...}, elDef: {...}, elView: {...}}
```

Para solucionarlo:

```
<span *ngIf="f.submitted && f.controls['email'].errors" class="text-danger">
  El correo es obligatorio</span>
<div class="wrap-input100 m-b-16">
  <input class="input100" type="email" name="email" [(ngModel)]="usuario.email" required email placeholder="Email">
    <span class="focus-input100"></span>
</div>
```

Probamos:

El correo es obligatorio

Email

En cuanto haya un correo válido esta información desaparece:

jsersan@gmail.com

Vamos a ponerle una clase que me indique cuando el campo es erróneo y esté animada.

```
<span *ngIf="f.submitted && f.controls['email'].errors"
  class="text-danger animated fadeIn">
```

Validamos el resto de los campos:

```

<span *ngIf="f.submitted && f.controls['nombre'].errors" class="text-danger
animated fadeIn">
    El nombre es obligatorio</span>
<div class="wrap-input100 m-b-16">
    <input class="input100" type="text" name="nombre" [(ngModel)]
    ="usuario.nombre" required minlength="2" placeholder="Nombre y apellidos">

    <span class="focus-input100"></span>
</div>

<span *ngIf="f.submitted && f.controls['password'].errors" class="text-danger
animated fadeIn">La contraseña debe de ser más de 6 letras</span>
<div class="wrap-input100 m-b-16">
    <input class="input100" type="password" name="password" [(ngModel)]
    ="usuario.password" minlength="6" required placeholder="Password">
    <span class="focus-input100"></span>
</div>

```

El resultado es:

<h3>CREAR NUEVA CUENTA</h3> <div style="background-color: #e0e0e0; padding: 10px; margin-bottom: 10px;"> <p>El correo es obligatorio</p> <input type="text" value="Email"/> </div> <div style="background-color: #e0e0e0; padding: 10px; margin-bottom: 10px;"> <p>El nombre es obligatorio</p> <input type="text" value="Nombre y apellidos"/> </div> <div style="background-color: #e0e0e0; padding: 10px; margin-bottom: 10px;"> <p>La contraseña debe de ser más de 6 letras</p> <input type="text" value="Password"/> </div> <div style="margin-top: 10px;"> <input type="checkbox"/> Recordar mi usuario ¿Ya tienes cuenta? / Ingresar </div>	<h3>CREAR NUEVA CUENTA</h3> <div style="background-color: #e0e0e0; padding: 10px; margin-bottom: 10px;"> <input type="text" value="jsersan@gmail.com"/> </div> <div style="background-color: #e0e0e0; padding: 10px; margin-bottom: 10px;"> <input type="text" value="Txema Serrano"/> </div> <div style="background-color: #e0e0e0; padding: 10px; margin-bottom: 10px;"> <input type="password" value="*****"/> </div> <div style="margin-top: 10px;"> <input type="checkbox"/> Recordar mi usuario ¿Ya tienes cuenta? / Ingresar </div>
---	--

Todo el botón y tengo toda la información:

```

Angular is running in the development mode. Call enableProdMode() to enable the production mode.
Formulario enviado
▶ UsuarioModel {email: "jsersan@gmail.com", password: "123456", nombre: "Txema Serrano"}
▶ NgForm {submitted: true, _directives: Array(3), ngSubmit: EventEmitter, form: FormGroup}
>

```

5.- Pantalla login y validaciones.

Desde este enlace podemos ir a la pantalla de login.

Recordar mi usuario [¿Ya tienes cuenta? / Ingresar](#)

Modificamos **registro.component.html**:

```
<div>
  <a routerLink="/login" class="txt1">
    ¿Ya tienes cuenta? / Ingresar
  </a>
</div>
..
```

Y en **login.component.html**:

```
<div>
  <a routerLink="/registro" class="txt1">
    ¿No tienes cuenta?
  </a>
</div>
```

Le ponemos una animación al principio:

```
src > app > pages > login > login.component.html > div.limiter.animated.fadeInLeft
1 <div class="limiter animated fadeInLeft">
2   <div class="container-login100">
```

También en el registro:

```
src > app > pages > registro > registro.component.html > div.limiter.animated.fadeInRight
1 <div class="limiter animated fadeInRight">
2   <div class="container-login100">
```

Para cambiarle la velocidad de la animación, en **vendor/animate/animate.css**:

```
EXPLORADOR registro.component.html animate.css login.component.html app.module.ts registro.component.ts
src > assets > vendor > animate > animate.css > .animated
1 @charset "UTF-8";
2
3 /*
4 * animate.css -http://danieleden.me/animate
5 * Version - 3.5.2
6 * Licensed under the MIT license - http://opensource.org/licenses/MIT
7 *
8 * Copyright (c) 2017 Daniel Eden
9 */
10
11 .animated {
12   animation-duration: 0.3s;
13 }
```

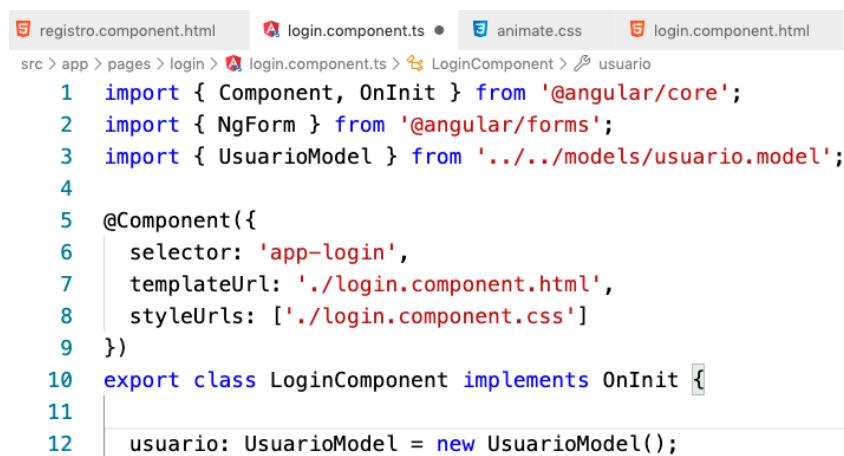
Cuando le demos al botón ingresar, en **login.component.ts**:

```

16  ngOnInit() {
17  }
18
19  // Fomulario válido
20
21  // Email Válido
22
23  // Contraseña de mínimo 6 caracteres
24
25  login(form: NgForm){
26  |  console.log('Fommulario válido')
27  }
~~

```

Para resolver esta tarea declaramos una variable llamada usuario



```

src > app > pages > login > login.component.ts > LoginComponent > usuario
1 import { Component, OnInit } from '@angular/core';
2 import { NgForm } from '@angular/forms';
3 import { UsuarioModel } from '../../../../../models/usuario.model';
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   |   usuario: UsuarioModel = new UsuarioModel();

```

Ahora lo puedo enlazar con los campos. Primero email:

```

<div class="wrap-input100 m-b-16">
  <input class="input100"
    type="text"
    name="email"
    [(ngModel)]="usuario.email"
    placeholder="email">

  <span class="focus-input100"></span>
</div>

```

Luego password:

```

<div class="wrap-input100 m-b-16" data-validate="Password
is required">
  <input class="input100"
    type="password"
    name="pass"
    [(ngModel)]="usuario.password"
    placeholder="Password">
  <span class="focus-input100"></span>
</div>

```

Debemos hacer que el formulario reaccione al submit:

```
4   <form (ngSubmit)="login(f)"  
5     #f = "ngForm"  
6     class="login100-form validate-form flex-sb flex-w">  
-
```

Pulsamos el botón Ingresar y nos sale este mensaje:

```
Angular is running in the development mode. Call enableProdMode() to enable the production mode.  
Imprimir SI el formulario es válido
```

Debo cambiar el código del constructor:

```
24   login( form: NgForm ) {  
25     console.log(form);  
26   }
```

La salida:

```
touched: true  
► _onDisabledChange: []  
► controls: {email: FormControl, pass: FormControl}  
► valueChanges: EventEmitter {_isScalar: false, observers: Array(0), closed: false, isScalar: false}  
► statusChanges: EventEmitter {_isScalar: false, observers: Array(0), closed: false, isScalar: false}  
status: "VALID"  
► value: {email: "jsersan", pass: "mfmk2556"}  
► errors: null
```

Voy a obligar que el email sea requerido y validación de email:

```
<div class="wrap-input100 m-b-16">  
  <input class="input100"  
    type="text"  
    name="email"  
    [(ngModel)]="usuario.email"  
    required  
    email  
    placeholder="email">
```

Lo mismo con el password y longitud mínima de 6:

```
<div class="wrap-input100 m-b-16" data-validate="Password  
is required">  
  <input class="input100"  
    type="password"  
    name="pass"  
    [(ngModel)]="usuario.password"  
    required  
    minlength="6"  
    placeholder="Password">  
  <span class="focus-input100"></span>  
</div>
```

Desactivamos los comentarios de las validaciones. Primero para el usuario:

```
<span *ngIf="f.submitted && f.controls['email'].errors"
      class="text-danger animated fadeIn">El correo es
      obligatorio</span>
<div class="wrap-input100 m-b-16">
    <input class="input100"
          type="text"
          name="email"
          [(ngModel)]="usuario.email"
```

Y para la contraseña:

```
<span *ngIf="f.submitted && f.controls['password']
      .errors" class="text-danger animated fadeIn">La
      contraseña debe de ser más de 6 letras</span>
<div class="wrap-input100 m-b-16" data-validate="Password
      is required">
    <input class="input100"
          type="password"
          name="password"
          [(ngModel)]="usuario.password"
          required
          minlength="6"
          placeholder="Password">
    <span class="focus-input100"></span>
</div>
```

Guardo los cambios y hago la validación en login.component.ts:

```
login( form: NgForm ) {
  if (form.invalid) { return; }

  console.log(this.usuario);
  console.log(form);
}
```

La salida es:

LOGIN

El correo es obligatorio

La contraseña debe de ser más de 6 letras

Recordar mi usuario [¿No tienes cuenta?](#)

INGRESAR

Si introducimos datos válidos:

LOGIN

Recordar mi usuario [¿No tienes cuenta?](#)

INGRESAR

Inspector Consola Depurador Red

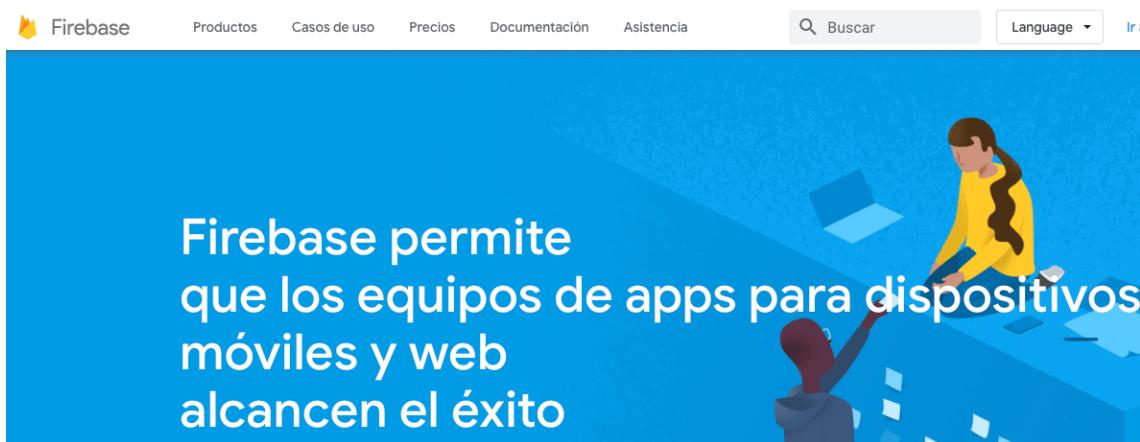
Filtrar salida

Errores Advertencias Registros Información Depurar CSS XHR Peticiones

```
Object { email: "jsersan@gmail.com", password: "123456" } login.component.ts:29:12
Object { submitted: true, _directives: (2) [], ngSubmit: () => void } login.component.ts:30:12
```

6.- FireBase y servicios REST.

Utilizaremos como **Backend Service FireBase**. Para esto:



Vamos a la consola:



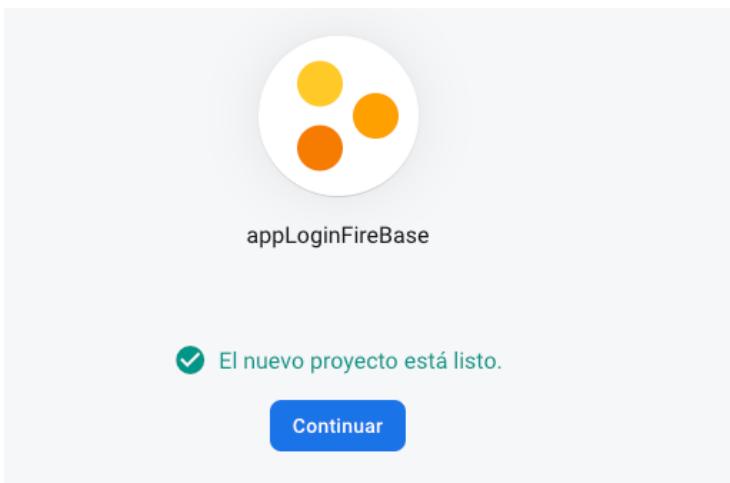
Creamos un nuevo proyecto:



Le damos a continuar:



Cuando termina:



Automáticamente va al panel:

A screenshot of the Firebase Project Overview page for the "appLoginFireBase" project. The left sidebar shows navigation options like "Project Overview", "Desarrollo" (Development), "Calidad" (Quality), and "Analíticas" (Analytics). The main area displays a banner with the text "Para empezar, añade Firebase a tu aplicación" and icons for iOS, Android, and web development. A "Plan Spark" button is also visible.

Me voy a la parte de autenticación que está en desarrollo:

Authentication

The screenshot shows the 'Authentication' section of the APPLOGIN interface. At the top, there are tabs for 'Usuarios', 'Método de inicio de sesión', 'Plantillas', and 'Uso'. Below the tabs is a search bar with placeholder text 'Buscar por dirección de correo electrónico, número de teléfono o UID de usuario'. To the right of the search bar are buttons for 'Añadir usuario', a refresh icon, and a more options icon. A table header row follows, with columns for 'Identificador', 'Proveedores', 'Fecha de creación', 'Inicio de sesión', and 'UID de usuario ↑'. In the center, there's a purple circular icon with a person's face and a badge, accompanied by text: 'Autentica y administra usuarios de una gran variedad de proveedores sin necesidad de utilizar ningún código del servidor'. Below this are links for 'Más información' and 'Consultar la documentación'. A blue button at the bottom says 'Configura el método de inicio de sesión'.

Le damos Método de Inicio de sesión:

The screenshot shows the 'Proveedores de inicio de sesión' (Providers of session start) configuration page. It has a table with two columns: 'Proveedor' (Provider) and 'Estado' (State). The providers listed are 'Correo electrónico/contraseña' (Email/password), 'Teléfono' (Phone), 'Google', and 'Play Juegos' (Play Games). All four providers are marked as 'Inhabilitado' (Disabled).

Proveedor	Estado
Correo electrónico/contraseña	Inhabilitado
Teléfono	Inhabilitado
Google	Inhabilitado
Play Juegos	Inhabilitado

Habilitamos *correo electrónico/contraseña*:

The screenshot shows the configuration for the 'Correo electrónico/contraseña' (Email/password) provider. It includes a toggle switch labeled 'Habilitar' (Enable) which is turned on. Below the switch is a descriptive text: 'Permite a los usuarios registrarse con su dirección de correo electrónico y contraseña. Nuestros SDK también proporcionan la verificación de la dirección de correo, la recuperación de contraseñas y las primitivas de cambio de dirección de correo.' There is also a link for 'Más información'. Another toggle switch labeled 'Enviar enlace por correo electrónico (inicio de sesión sin contraseña)' (Send link by email (login without password)) is also turned on. At the bottom are 'Cancelar' (Cancel) and 'Guardar' (Save) buttons.

El resultado:

Proveedores de inicio de sesión	
Proveedor	Estado
✉ Correo electrónico/contraseña	Habilitada

Nos permitirá invocar una API que nos permita crear usuarios. Dentro de autenticación nos quedamos en la pantalla de Usuarios:

Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario ↑
Este proyecto aún no tiene usuarios				

Por otro lado, consultamos la documentación de los servicios REST para interactuar con **Firebase Auth**.

Documentation

- Presentación
- Guías
- Referencia**
- Ejemplos
- Libraries

API Reference

CLI Reference

iOS – Swift

iOS – Objective-C

Android

JavaScript

Node.js (Client)

C++

Unity

Firebase > Docs > Referencia

☆☆☆☆☆

Firebase Auth REST API

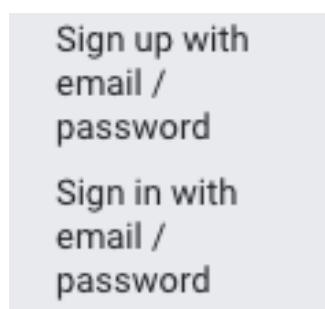
API Usage

You can query the Firebase Auth backend through a REST API. This can be used for various operations such as creating new users, signing in existing ones and editing or deleting these users.

Throughout this document, `API_KEY` refers to the Web API Key, which can be obtained on the [project settings](#) page in your admin console.

★ [HTTPS is required. Firebase only responds to encrypted traffic so that your data remains safe.](#)

En esta página trabajaremos con estos servicios en la parte derecha de la página:



En el momento de imprimir este manual esta es la relación de usuarios registrados:

The screenshot shows the Firebase Authentication console under the 'Users' tab. It displays a table of registered users with the following columns: Identificador, Proveedores, Fecha de creación, Fecha de acceso, and UID de usuario. The table contains six rows of data. A search bar at the top allows filtering by email, phone number, or UID. A blue 'Agregar usuario' (Add user) button is located in the top right corner of the table area. At the bottom, there are pagination controls for 'Filas por página' (Rows per page) set to 50, and a page indicator showing '1 - 6 of 6'.

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
txitxi@correo.es	✉️	21 may 2022	21 may 2022	hy4LXWguCUheYfQbFP1JjXduV1a2
tutxo@correo.es	✉️	15 may 2022	15 may 2022	wp5DnwfPSaeVaW9xEp1O9lxMw...
txatxi@correo.es	✉️	15 may 2022	21 may 2022	HPuQrUKZifh10cAD0DPbZtxMLC22
jsersan@gmail.com	✉️	15 may 2022	23 may 2022	pdnBjgcwxNRLNdmBaONK0rv1Y1...
txutxi@correo.es	✉️	14 may 2022	20 may 2022	fEnvNmSPb1MKHTQqtw4uyhRRo...
txema@gmail.com	✉️	14 may 2022	15 may 2022	JFpdnrcKkZZvMtjm8z5tJl67fQJ2

Sign up User:

Regístrate con correo electrónico / contraseña

Puede crear un nuevo usuario de correo electrónico y contraseña mediante la emisión de una solicitud HTTP `POST` al punto final Auth `signupNewUser`.

Método: POST

Tipo de contenido: aplicación/json

Endpoint

```
https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API_KEY]
```



Carga útil del cuerpo de la solicitud

Nombre de la propiedad	Escribe	Descripción
Email	cuerda	El correo electrónico para que el usuario cree.
contraseña	cuerda	La contraseña para que el usuario cree.
volverSecureToken	booleano	Si devolver o no un ID y un token de actualización. Siempre debe ser cierto.

En `app.module.ts` debemos importar el `HttpClientModule`:

```
4 import {HttpClientModule} from '@angular/common/http';
5
6 import { AppRoutingModule } from './app-routing.module';
7 import { AppComponent } from './app.component';
8
9 import { RegistroComponent } from './pages/registro/registro.component';
10 import { HomeComponent } from './pages/home/home.component';
11 import { LoginComponent } from './pages/login/login.component';
12
13 @NgModule({
14   declarations: [
15     AppComponent,
16     RegistroComponent,
17     HomeComponent,
18     LoginComponent
19   ],
20   imports: [
21     BrowserModule,
22     AppRoutingModule,
23     FormsModule,
24     HttpClientModule
```

Me creo un nuevo servicio para manejar todo lo relacionado con la autenticación.

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
iMac-de-Jose:appLoginFireBase jsersan$ ng g s services/auth
Your global Angular CLI version (8.3.21) is greater than your local
version (7.3.9). The local Angular CLI version is used.
```

```
To disable this warning use "ng config -g cli.warnings.versionMismatch false".
CREATE src/app/services/auth.service.spec.ts (323 bytes)
CREATE src/app/services/auth.service.ts (133 bytes)
iMac-de-Jose:appLoginFireBase jsersan$ █
```

Copiamos de la documentación:

Endpoint

```
https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API_KEY]
```

En **auth.service.ts**:

```
6  export class AuthService {
7
8    // Crear nuevos usuarios
9
L0   https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API_KEY]
L1
L2   constructor() { }
L3 }
```

El otro, **Sign in email/password** también lo copio:

Endpoint

```
https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=[API_KEY]
```



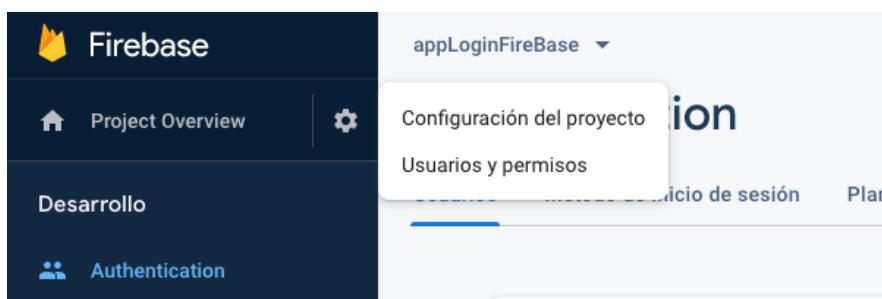
```
ml x  A auth.service.ts ●  A login.component.ts  animate.css  login.component.html  app.module.ts  registro.component.ts
> app > services > A auth.service.ts > AuthService
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class AuthService {
8
9   // Crear nuevos usuarios
10  // https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API_KEY]
11
12  // Login
13  // https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=[API_KEY]
14
15  constructor( private http: HttpClient) { }
16 }
```

Creamos dos propiedades privadas:

- a) **url**, para copiar lo común en las dos direcciones:

```
7  export class AuthService {
8
9    private url = 'https://identitytoolkit.googleapis.com/v1/accounts:';
10
11   // Crear nuevos usuarios
12   // https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=\[API\_KEY\]
13 }
```

- b) **API_KEY**: la recuperamos en firebase en Configuración del Proyecto:



Copiamos el valor de la clave de API de la web:

Clave de API de la web AIzaSyA3azkR5_1Y7BRiE-cKAQy3vYDkLNufSvU

Queda en nuestro servicio:

```
7  export class AuthService {
8
9    private url = 'https://identitytoolkit.googleapis.com/v1/accounts:';
10   private API_KEY = 'AIzaSyA3azkR5_1Y7BRiE-cKAQy3vYDkLNufSvU';
11 }
```

A continuación, preparamos los servicios a utilizar posteriormente en el servicio:

```
18
19  constructor( private http: HttpClient) { }
20
21  logout() {
22
23  }
24
25  login( usuario: UsuarioModel) {
26
27  }
28
29  nuevoUsuario(usuario: UsuarioModel ) {
30
31  }
32
33 }
```

Registrar nuevo usuario:

Para que el servicio sea reconocido por nuestra app, bajamos el servidor y lo volvemos a subir.

En esta parte debemos crear el usuario y que aparezca aquí en firebase:

Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario ↑
Este proyecto aún no tiene usuarios				

Revisando la documentación nos dice que tenemos que mandar:

Endpoint

[https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=\[API_KEY\]](https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API_KEY])

Request Body Payload

Property Name	Type	Description
email	string	The email for the user to create.
password	string	The password for the user to create.
returnSecureToken	boolean	Whether or not to return an ID and refresh token. Should always be true.

Así en servicio:

```

29   nuevoUsuario(usuario: UsuarioModel ) {
30
31     const athData = {
32       email: usuario.email,
33       password: usuario.password,
34       returnSecureToken: true
35     };
36
37   }

```

Email y password ya vienen del objeto usuario. Entonces:

```

31   const athData = {
32     ...usuario,
33     returnSecureToken: true
34   };

```

Le va a proporcionar las tres propiedades del objeto: email, password, nombre. Este último *Firebase* no lo tendrá en cuenta ya que no lo tenemos en nuestra base de datos.

Llamamos a nuestro servicio mediante una petición POST para esta petición:

Method: POST

Content-Type: application/json

Endpoint

```
https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=[API_KEY]
```

Request Body Payload

Property Name	Type	Description
email	string	The email for the user to create.
password	string	The password for the user to create.
returnSecureToken	boolean	Whether or not to return an ID and refresh token. Should always be true.

Debo formar la dirección del servicio con los valores de nuestra API en FireBase:

Así en el servicio con la API_KEY y url definidos en la clase:

```

19   constructor(private http: HttpClient) { }
20
21   logout(){}
22
23   login(usuario: UsuarioModel){}
24
25   nuevoUsuario(usuario: UsuarioModel){
26     const athData = {
27       ...usuario,
28       returnSecureToken: true
29     }
30
31     return this.http.post(` ${this.url}signUp?key=${this.API_KEY}`,
32     athData
33   );
34
35 }
```

Ahora debemos conectar lo que es nuestro servicio con nuestra página de registro. En **registro.component.ts** para inyectar nuestro *AuthService*:

```

4 import { AuthService } from '../../../../../services/auth.service';
5
6 @Component({
7   selector: 'app-registro',
8   templateUrl: './registro.component.html',
9   styleUrls: ['./registro.component.css']
10 })
11 export class RegistroComponent implements OnInit {
12
13   usuario: UsuarioModel;
14
15   constructor(private auth: AuthService) { }

```

Ahora que sé que la información es correcta:

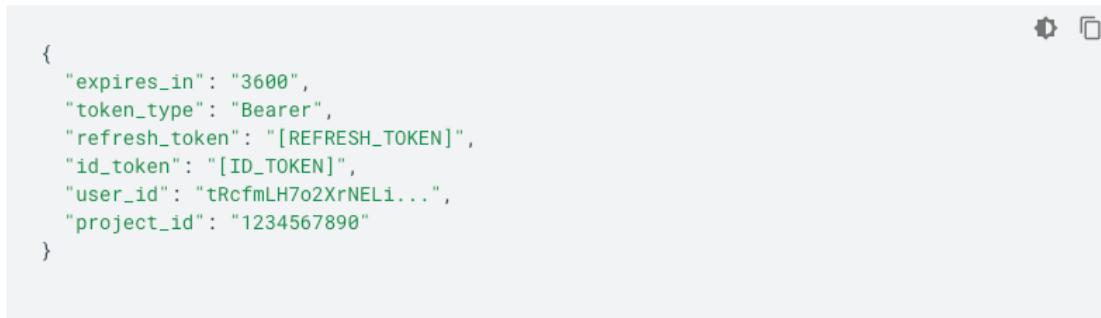
```

21   onSubmit(form: NgForm) {
22
23     if ( form.invalid ) {return; } // formulario inválido no hace nada
24
25     this.auth.nuevoUsuario(this.usuario).subscribe();
26   }

```

En el subscribe tengo la información que manda Firebase. Si es exitosa:

Sample response



```

{
  "expires_in": "3600",
  "token_type": "Bearer",
  "refresh_token": "[REFRESH_TOKEN]",
  "id_token": "[ID_TOKEN]",
  "user_id": "tRcfmLH7o2XrNELi...",
  "project_id": "1234567890"
}

```

Lo que me interesa el *id_token*. Comprobamos cuál es la respuesta:

```

21   onSubmit(form: NgForm) {
22
23     if ( form.invalid ) {return; } // formulario inválido no hace nada
24
25     this.auth.nuevoUsuario(this.usuario)
26       .subscribe(resp =>{
27         console.log(resp);
28       });
29   }

```

Intentamos crear un nuevo usuario y me da este error:

```
Angular is running in the development mode. Call enableProdMode() to enable the production mode.

✖ ▶ ERROR TypeError: Cannot read property 'subscribe' of undefined
    at RegistroComponent.push../src/app/pages/registro/registro.component.ts.RegistroComponent.onSubmit
    at Object.eval [as handleEvent] (RegistroComponent.html:4)
    at handleEvent (core.js:123107)
    at callWithDebugContext (core.js:24177)
    at Object.debugHandleEvent [as handleEvent] (core.js:23904)
    at dispatchEvent (core.js:20556)
    at core.js:122046
    at SafeSubscriber.schedulerFn [as _next] (core.js:13527)
    at SafeSubscriber.push../node_modules/rxjs/_esm5/internal/Subscriber.js.SafeSubscriber.__tryOrUnsus
    at SafeSubscriber.push../node_modules/rxjs/_esm5/internal/Subscriber.js.SafeSubscriber.next (Subscri
✖ ▶ ERROR CONTEXT ▶ DebugContext_ {view: {...}, nodeIndex: 3, nodeDef: {...}, elDef: {...}, elView: {...}}
```

Para solucionarlo bajamos el servicio y lo volvemos a levantar:

CREAR NUEVA CUENTA

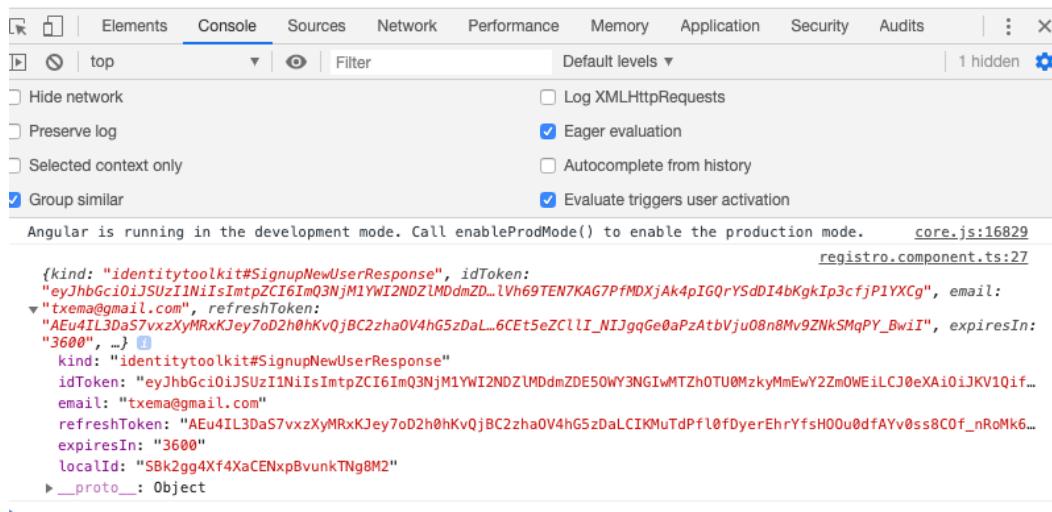
txema@gmail.com

Txema Serrano

•••••••

Recordar mi usuario [¿Ya tienes cuenta?](#) / [Ingresar](#)

CREAR CUENTA



La información:

```
idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6Im
email: "txema@gmail.com"
refreshToken: "AEu4IL3DaS7vxzXyMRxKJey7o
expiresIn: "3600"
localId: "SBk2gg4Xf4XaCENxpBvunkTNg8M2"
```

- **email.**
- **idToken.**
- **expiresIN:** 1 hora
- **localId:** id único que me genera firebase que me garantiza que no habrá otro usuario con este id.
- **refresToken:** es un token para renovarlo en el caso que se desee.

Si limpio la consola y le vuelvo a dar al botón me da un error:

```
✖ ▶ POST https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyA3azkR5_1Y7BRiE-cKAQy3vY
✖ ▶ ERROR
  ▼ HttpErrorResponse {headers: HttpHeaders, status: 400, statusText: "OK", url:
    "https://identitytoolkit.googleapis.com/v1/accounts...signUp?key=AIzaSyA3azkR5_1Y7BRiE-cKAQy3vY
      ▶ headers: HttpHeaders {normalizedNames: Map(0), lazyUpdate: null, lazyInit: f}
        status: 400
        statusText: "OK"
        url: "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyA3azkR5_1Y7BRiE-cKAQy3vY"
      ok: false
      name: "HttpErrorResponse"
      message: "Http failure response for https://identitytoolkit.googleapis.com/v1/accounts:signUp?ke...
      ▶ error: {error: {}}
      ▶ __proto__: HttpResponseBase
```

Abrimos el objeto del error:

```
  ▶ url: "https://identitytoolkit.googleapis.com/v1/accounts:signUp"
  ▶ ok: false
  ▶ name: "HttpErrorResponse"
  ▶ message: "Http failure response for https://identitytoolkit.googleapis.com/v1/accounts:signUp?ke...
  ▶ ▶ error:
    ▶ ▶ error:
      ▶ ▶ code: 400
      ▶ ▶ message: "EMAIL_EXISTS"
```

Si vamos a firebase:

Buscar por dirección de correo electrónico, número de teléfono o UID de usuario					Añadir usuario		
Identificador	Proveedores	Fecha de creación	Inicio de sesión	UID de usuario ↑			
txema@gmail.com		26 dic. 2019	26 dic. 2019	SBk2gg4Xf4XaCENxpBvunkTNg8...			
Filas por página: 50		1-1 de 1		< >			

Para majear este error, en el *subscribe* voy a capturar el código del mismo:

```

21  onSubmit(form: NgForm) {
22
23    if ( form.invalid ) {return; } // formulario inválido no hace nada
24
25    this.auth.nuevoUsuario(this.usuario)
26      .subscribe(resp => {
27        console.log(resp);
28      }, (err) => {
29        console.log(err);
30      });
31

```

Ejecutamos de nuevo la app. La salida es:

```

HttpErrorResponse {headers: HttpHeaders, status: 400, statusText: "0
https://identitytoolkit.googleapis.com/v1/accounts...signUp?key=AIzaSy
▶ headers: HttpHeaders {normalizedNames: Map(0), lazyUpdate: null, l
status: 400
statusText: "OK"
url: "https://identitytoolkit.googleapis.com/v1/accounts:signUp?ke
ok: false
name: "HttpErrorResponse"
message: "Http failure response for https://identitytoolkit.google
▶ error: {error: {...}}
▶ __proto__: HttpResponseBase

```

El mensaje del error se encuentra en:

```

----- -----
message: "Http failure response for https://identity...
▼ error:
  ▼ error:
    code: 400
    message: "EMAIL_EXISTS"
-----
```

El texto está en **error.error.message**:

```

25  this.auth.nuevoUsuario(this.usuario)
26    .subscribe(resp => {
27      console.log(resp);
28    }, (err) => {
29      console.log(err.error.error.message);
30    });
31

```

Así el mensaje de la consola:

```

Angular is running in the development mode. Call enableProdMode() to enable the core.js:16829
production mode.
✖ ▶ POST https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyA3azkRS5 zone.js:3243
  _1Y7BRiE-cKAQy3vYDkLNufSvU 400
EMAIL_EXISTS registro.component.ts:29

```

Login de Usuarios:

Para realizarlo debemos tener un usuario creado en firebase.



The screenshot shows the Firebase Authentication console with a single user listed. The columns are: Identificador (txema@gmail.com), Proveedores (Email), Fecha de creación (26 dic. 2019), Inicio de sesión (26 dic. 2019), and UID de usuario (SBk2gg4Xf4XaCENxpBvunkTNg8...). There is a search bar at the top, an 'Añadir usuario' button, and a table header with sorting arrows. At the bottom, there are pagination controls: 'Filas por página: 50', '1-1 de 1', and navigation arrows.

Trabajamos en la parte del servicio **auth.service.ts** y hacemos la función de *login* que es muy parecida a la *nuevoUsuario* que ya hemos hecho:

Endpoint

[https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=\[API_KEY\]](https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=[API_KEY])

Lo implementamos en **auth.service.ts**:

```

26  login(usuario: UsuarioModel){
27    const authData = {
28      ...usuario,
29      returnSecureToken: true
30    }
31
32    return this.http.post(
33      `${this.url}/verifyPassword?key=${this.API_KEY}`,
34      authData
35    ).pipe(
36      map( resp => { // Transforma la data
37        console.log('Entro en el map de rxjs')
38        this.guardarToken(resp['idToken']);
39        return resp;
40      })
41    );
42  }

```

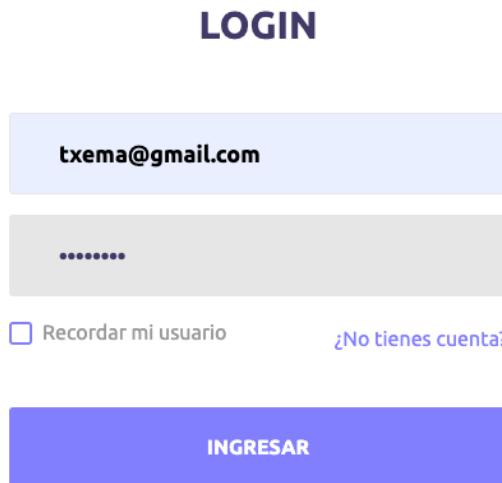
Para hacer la prueba en **login.component.ts**. Recuerden importar AuthService:

```

27  login( form: NgForm ) {
28    if (form.invalid) { return; }
29
30    this.auth.login( this.usuario )
31      .subscribe( resp => {
32        console.log(resp);
33      }, (err) =>{
34        console.log(err.error.error.message);
35      });
36  }

```

Lo probamos:



La salida por consola:

```
login.component.ts:32
{
  kind: "identitytoolkit#VerifyPasswordResponse",
  localId: "SBk2gg4Xf4XaCENxpBvunkTNg8M2",
  email: "txema@gmail.com",
  displayName: "",
  idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6ImQ3NjM1YWI2NDZlMDdmZD...sVledqXEQkzALbfNZjCZtng00c5oIFdrL5eci",
  ...
}
kind: "identitytoolkit#VerifyPasswordResponse"
localId: "SBk2gg4Xf4XaCENxpBvunkTNg8M2"
email: "txema@gmail.com"
displayName: ""
idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6ImQ3NjM1YWI2NDZlMDdmZD...50WY3NGIwMTZhOTU0MzkyM..."
registered: true
refreshToken: "AEu4IL1c3WBrrkkZkULaky9MDEIDi_EHxa7s2IdQoWQHFHtkkS-DmDGY_0_VW9kJDvJcC..."
expiresIn: "3600"
▶ __proto__: Object
```

Ya tengo el **id_token** que es lo que necesito para validar todas mis peticiones con el **BackEnd**. Debiéramos grabarlo en el localStorage para poderlo utilizar cuando lo necesite.

El logout es la destrucción de este id_token.

Si pongo una contraseña incorrecta:

```
✖ ▶ POST https://identitytoolkit.googleapis.com/v1/acc
INVALID_PASSWORD
```

Guardar el token en LocalStorage:

En este punto ya tenemos el **token_id** y los vamos a guardar en el *LocalStorage*.

Trabajaremos en **auth.service.ts**. Recordemos que estas peticiones al servicio retornaban un observable donde puedo hacer ciertas modificaciones antes de notificar a la página de registro o de login. Cuando el login lance la suscripción podrá lanzar un código antes. Ahí es donde almacenaré el token. Creamos dos métodos después de esta parte:

```
39   return this.http.post(
40     `${this.url}signUp?key=${this.API_KEY}`,
41     authData
42   );
```

En el servicio creamos una variable **userToken** inicializado como nulo, sin valor. Este userToken me va ayudar a comprobar a ver si ya existe el idToken.

```
8  export class AuthService {
9
10  private url = 'https://ide
11  private API_KEY = 'AIzaSyA
12
13  userToken: string;
```

Creamos los dos métodos para leer y escribir y el token:

```
48  private guardarToken( idToken: string ) {
49
50    this.userToken = idToken;
51    localStorage.setItem('token', idToken);
52
53  }
54
55  leerToken() {
56
57    if ( localStorage.getItem('token') ) {
58      this.userToken = localStorage.getItem('token');
59    } else {
60      this.userToken = '';
61    }
62
63    return this.userToken;
64 }
```

Hay que invocar el método `guardarToken`. Para eso en el método de `nuevoUsuario` del servicio:

```
41  return this.http.post(
42    `${this.url}signUp?key=${this.API_KEY}`,
43    authData
44  ).pipe(
45    map() // Transforma una data como intermedio
46  );
```

El map me está marcando un error por lo que debemos importarlo:

```
auth.service.ts ●
src > app > services > auth.service.ts > ...
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3  import { UsuarioModel } from '../models/usuario.model';
4
5  import { map } from 'rxjs/operators';
6
```

El map va a invocar a una función que sólo se ejecutará si tenemos éxito en la petición:

```
43  return this.http.post(
44    `${this.url}signUp?key=${this.API_KEY}`,
45    authData
46  ).pipe(
47    map( resp => {
48      console.log('Entro en el mapa del RXJS')
49      this.guardarToken( resp['idToken'] );
50      return resp;
51    }) // Transforma una data como intermediario
52  );
```

Copiamos el pipe y lo copiamos en el login del servicio:

```
46  ).pipe(
47    map( resp => {
48      console.log('Entro en el mapa del RXJS')
49      this.guardarToken( resp['idToken'] );
50      return resp;
51    }) // Transforma una data como intermediario
52  );
```

Resultado:

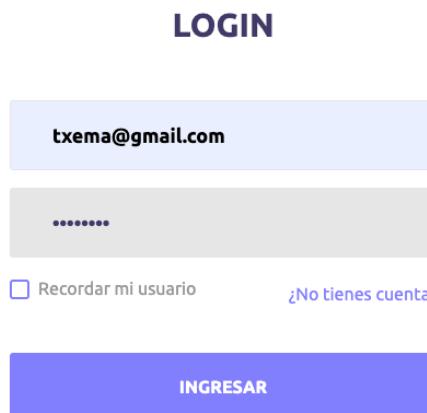
```
23  login( usuario: UsuarioModel ) {
24    const authData = {
25      ...usuario,
26      returnSecureToken: true
27    };
28
29    return this.http.post(
30      `${this.url}signInWithPassword?key=${this.API_KEY}`,
31      authData
32    ).pipe(
33      map( resp => {
34        console.log('Entro en el mapa del RXJS')
35        this.guardarToken( resp['idToken'] );
36        return resp;
37      }) // Transforma una data como intermediario
38    );
39  }
```

Guardamos los cambios y regresamos a la aplicación. Pongo un login válido:

```
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
Entro en el mapa del RXJS

▼ {kind: "identitytoolkit#VerifyPasswordResponse", localId: "SBk2gg4Xf4XaCENxpBvunkTNg8M2", email: "txema@gmail.com"
  "eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlNwNlZDZlNDBkY2QxZW...nAzGxIX8YL3qvL11vR7sQQAvHoj5Kfm50W40MiCdVqGSbYrQA", ...} ⓘ
  kind: "identitytoolkit#VerifyPasswordResponse"
  localId: "SBk2gg4Xf4XaCENxpBvunkTNg8M2"
  email: "txema@gmail.com"
  displayName: ""
  idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlNwNlZDZlNDBkY2QxZWZmNDA3MDQ4ODY3YjFlZDFlNzA2Njg2YTAiLCJ0eXAiOiJKV1Qi
  registered: true
  refreshToken: "AEu4IL0YtpIVoprRvmt2d_jCDkeggDca2vMniYoAyiik8rPkZZSKaqCsZlC0bERCcgxGphTacPdY7MGZ7lu4aj9GkW7Njgzl
  expiresIn: "3600"
  ► __proto__: Object
>
```

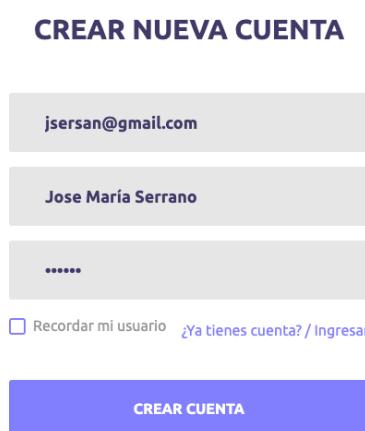
Si miramos ahora en el localStorage:



Sources	Network	Performance	Memory	Application	Security	Audits
C	Filter					
Key	Value					

token: eyJhbGciOiJSUzI1NiIsImtpZCI6ImQ3NjM1YWl2NDZlMDdmZDE5OWY3NGlwMTZhOTU0MzkyMmEwY2ZmOWEiLCJ0eXAiOiJKV1QiFQ.eyJpc3MiOiJodHRwczovL3.

Si creamos un nuevo usuario:



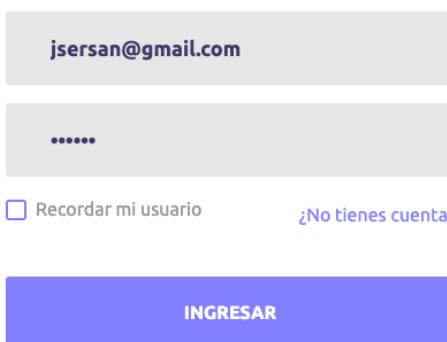
Le damos a crear cuenta:

Entro en el mapa del RXJS

```
{
  kind: "identitytoolkit#SignupNewUserResponse",
  idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlNWNlZDZlNDBkY2QxZW...yESShVG8i
  "jsersan@gmail.com", refreshToken: "AEu4IL2DjYfJ-Tc2jngH3o0_NKKM407JN9LcWBWPFcK_Cmb5...qgr09eG04lCoh02Rk4DUDKyzn2AZFu"
  kind: "identitytoolkit#SignupNewUserResponse"
  idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlNWNlZDZlNDBkY2QxZW...yESShVG8i
  "jsersan@gmail.com", refreshToken: "AEu4IL2DjYfJ-Tc2jngH3o0_NKKM407JN9LcWBWPFcK_Cmb5...qgr09eG04lCoh02Rk4DUDKyzn2AZFu"
  email: "jsersan@gmail.com"
  refreshTokens: "AEu4IL2DjYfJ-Tc2jngH3o0_NKKM407JN9LcWBWPFcK_Cmb5...qgr09eG04lCoh02Rk4DUDKyzn2AZFu"
  expiresIn: "3600"
  localId: "YVYAfzb0QkZJ5Dg9ebSYxOenrA1"
  ► __proto__: Object
```

Y si entro con esa cuenta de usuario:

LOGIN



Resultado:

Entro en el mapa del RXJS

```
{
  kind: "identitytoolkit#VerifyPasswordResponse",
  localId: "YVYAfzb0QkZJ5Dg9ebSYxOenrA1",
  email: "jsersan@gmail.com", disp
  "eyJhbGciOiJSUzI1NiIsImtpZCI6ImQ3NjM1YWI2NDZlMDdmZD...fZBBI6G--biRg8Tf7S-98u2INl0ok-liHlVMd1HYBYQRTjEGA", ...
  kind: "identitytoolkit#VerifyPasswordResponse"
  localId: "YVYAfzb0QkZJ5Dg9ebSYxOenrA1"
  email: "jsersan@gmail.com"
  displayName: ""
  idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6ImQ3NjM1YWI2NDZlMDdmZD...fZBBI6G--biRg8Tf7S-98u2INl0ok-liHlVMd1HYBYQRTjEGA", ...
  registered: true
  refreshToken: "AEu4IL0IFiVUyEqfYOTA1pUEBwV5JqTt5doW0mrlnNSGQ8sX31ev-30Y0Qv5CetwJpW_7Va2Dwi2Ty2kXTVya_o9Xig_kEV8mqmd_Sn1n"
  expiresIn: "3600"
  ► proto : Object
```

Para terminar con esta parte, podríamos invocar a la función *leerToken* cuando inicializamos el servicio.

```
17  constructor( private http: HttpClient) {
18    this.leerToken();
19 }
```

Justo cuando ingresa ya sabemos si tenemos un token.

SweetAlert: notificaciones al usuario.



Instalación:

```
iMac-de-Jose:appLoginFireBase jsersan$ npm install sweetalert2
+ sweetalert2@9.5.3
added 1 package from 6 contributors and audited 43266 packages in 19.785s

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

found 4 vulnerabilities (1 low, 3 moderate)
  run `npm audit fix` to fix them, or `npm audit` for details
iMac-de-Jose:appLoginFireBase jsersan$ 
```

Vamos a hacer un ingreso con un password erróneo:

```
Angular is running in the development mode. Call enableProdMode() in your
main.ts file to switch to the production environment.
3 POST https://identitytoolkit.googleapis.com/v1/accounts/SignIn
  INVALID_PASSWORD
```

Lo vemos en la consola, pero el usuario no tiene opción de saber si la contraseña es correcta o no. Para ello, vamos a utilizar el sweet alert.

En la página del login:

```
29
30   Swal
```

Debemos importarlo, en la página del a documentación:

```
// ES6 Modules or TypeScript
import Swal from 'sweetalert2'

// CommonJS
const Swal = require('sweetalert2')
```

Lo importamos y desaparece el error:

```
login.component.ts • auth.service.ts index.html
src > app > pages > login > login.component.ts > LoginComponent
1 import { Component, OnInit } from '@angular/core';
2 import { NgForm } from '@angular/forms';
3 import { UsuarioModel } from '../../../../../models/usuario.model';
4 import { AuthService } from '../../../../../services/auth.service';
5 import Swal from 'sweetalert2';
```

Hacemos el alert:

```
21 login( form: NgForm ) {
22   if (form.invalid) { return; }
23
24   Swal.fire({
25     allowOutsideClick: false,
26     text: 'Espere por favor'
27   });
28   Swal.showLoading();
```

El efecto es:



Y entro:

```
⚠ [Deprecation] Resource requests whose URLs contained both removed whitespace (`\n`, `\r`, `\t`) characters and less-than characters encode less-than characters from places like element attribute values in order to load these resources. See https://www.chromes
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
Entro en el mapa del RXJS
>
  ↵ {kind: "identitytoolkit#VerifyPasswordResponse", localId: "YVYAfzb0QkZJ5Dg9ebSYx0enrAll", email: "jsersan@gmail.com", displayl
  ↵ "eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlNWNlZDZlNDKy2QxZW...AvXAzNyKe-GIWGqE-SFKV43wQkDQHhF2sb5YsHgDTwT1ctlJw", ...}
```

Pero el spinner no desaparece. Para quitarlo en el subscribe puedo cancelar el sweetalert:

```
31 this.auth.login( this.usuario )
32   .subscribe( resp => {
33     console.log(resp);
34     Swal.close();
35   }, (err) => {
36     console.log(err.error.error.message);
37   });
38 }
```

Esto mismo debemos hacer en el caso que tengamos un error:



Error al autenticar

INVALID_PASSWORD



Haremos esto mismo en la parte del registro:

```

22  onSubmit(form: NgForm) {
23
24    if ( form.invalid ) {return; } // formulario inválido
25
26    Swal.fire({
27      allowOutsideClick: false,
28      icon: 'info',
29      text: 'Espere por favor'
30    });
31    Swal.showLoading();
32
33    this.auth.nuevoUsuario(this.usuario)
34      .subscribe(resp => {
35        console.log(resp);
36        Swal.close();
37      }, (err) => {
38        Swal.fire({
39          title: 'Email ya existe',
40          icon: 'error',
41          text: err.error.error.message
42        });
43        Swal.showLoading();
44        console.log(err.error.error.message);
45      });
46  }

```

Lo siguiente que nos queda es hacer la navegación. En **registro.component.ts**:

```

6  import { Router } from '@angular/router';
7
8  @Component({
9    selector: 'app-registro',
10   templateUrl: './registro.component.html',
11   styleUrls: ['./registro.component.css']
12 })
13 export class RegistroComponent implements OnInit {
14
15   usuario: UsuarioModel;
16
17   constructor(private auth: AuthService,
18             private router: Router) { }

```

Ya tengo importado mi router y lo voy a usar cuando sé que tengo una autentificación válida. Después de la comprobación se redirige al /home:

```

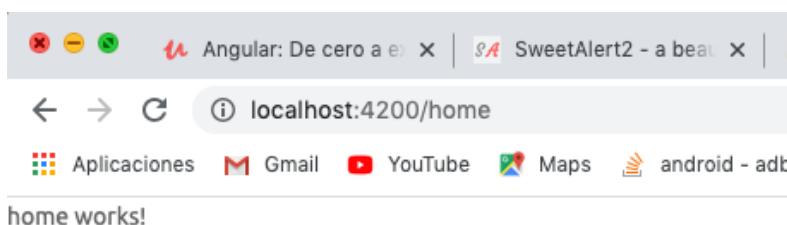
35   this.auth.nuevoUsuario(this.usuario)
36     .subscribe(resp => {
37       console.log(resp);
38       Swal.close();
39       this.router.navigateByUrl('/home');
--
```

Lo mismo hago en login:

```

17  constructor( private auth: AuthService,
18    | | | | | private router: Router ) { }
19
20  ngOnInit() {
21  }
22
23  login( form: NgForm) {
24    if (form.invalid) { return; }
25
26    Swal.fire({
27      allowOutsideClick: false,
28      icon: 'info',
29      text: 'Espere por favor'
30    );
31    Swal.showLoading();
32
33    this.auth.login( this.usuario )
34      .subscribe( resp => {
35        console.log(resp);
36        Swal.close();
37        this.router.navigateByUrl('/home');
```

Después del login:



También funciona después de crear una nueva cuenta:

CREAR NUEVA CUENTA

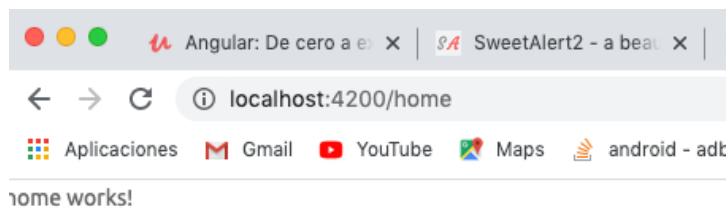
daniel@gmail.com

Daniel Serrano

Recordar mi usuario [¿Ya tienes cuenta? / Ingresar](#)

CREAR CUENTA

Resultado:



La consola:

```
Entro en el mapa del RXJS
{
  "kind": "identitytoolkit#SignupNewUserResponse",
  "idToken": "eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlNWNLZDZLNDBkY2QxZW_G8vU472DMJaOS_dR93JbU6xdRoWrWOrCkMv0
  "daniel@gmail.com",
  "refreshToken": "AEu4IL0hV-wJFCiAddV7aXLIX_gCNd1MmfCvBrjhcyL5aKzH...BmqipHKEvZL61X0rFAtTVZoZph5fRbLty828jJUNhd06Js0",
  "expiresIn": "3600",
  "localId": "itejXR8kmyS0ZTMKrurAq28tUJJ3"
}
▶ __proto__: Object
```

Recordar usuario:

En **login.component.ts** me creo una variable booleana para ver si está activo el *check*.

```
13  export class LoginComponent implements OnInit {
14
15    usuario: UsuarioModel = new UsuarioModel();
16
17    recuerdame = true;
```

La voy a asociar con el check a través del *ngModel* del HTML:

```
<div class="flex-sb-m w-full p-t-3 p-b-24">
  <div class="contact100-form-checkbox">
    <input [(ngModel)]="recuerdame"
      class="input-checkbox100" id="ckb1"
      type="checkbox" name="remember-me">
    <label class="label-checkbox100" for="ckb1">
      Recordar mi usuario
    </label>
  </div>
```

Además, desde el momento que tenga un login válido, voy a grabar en el *localStorage* el email del usuario:

```
36    this.auth.login( this.usuario )
37    .subscribe( resp => {
38      console.log(resp);
39      Swal.close();
40
41      if ( this.recuerdame ) {
42        localStorage.setItem('email', this.usuario.email);
43      }
    }
```

Para que cuando se cargue la página esta información esté disponible en el *ngOnInit()*:

```
23  ngOnInit() {
24    if ( localStorage.getItem('email') ) {
25      this.usuario.email = localStorage.getItem('email');
26      this.recuerdame = true;
27
28    }
29  }
```

Probamos:

The screenshot shows a browser window with the URL `localhost:4200/home`. The page has a "LOGIN" header. There are two input fields: one for email with the value `jsersan@gmail.com` and one for password with the value `.....`. Below the password field is a checkbox labeled "Recordar mi usuario" and a link "¿No tienes cuenta?". A large blue button at the bottom is labeled "INGRESAR".

Le damos hacia atrás en el navegador:

The screenshot shows the same login page as before, but the password input field now contains no text, demonstrating that the previous input was cleared when the user navigated back.

En creación de cuenta, en **registro.component.ts**:

```
13 export class RegistroComponent implements OnInit {
14
15   recuerdame = false;
16   usuario: UsuarioModel;
```

Lo enlazamos con el check en le registro del componente HTML:

```
<div class="flex-sb-m w-full p-t-3 p-b-24">
  <div class="contact100-form-checkbox">
    <input [(ngModel)]="recuerdame"
      class="input-checkbox100" id="ckb1"
      type="checkbox" name="remember-me">
    <label class="label-checkbox100" for="ckb1">
      Recordar mi usuario
    </label>
  </div>
```

Creo un nuevo usuario:

CREAR NUEVA CUENTA

prueba@gmail.com

Prueba Urrutikoetxea

Recordar mi usuario [¿Ya tienes cuenta? / Ingresar](#)

CREAR CUENTA

Se crea correctamente:

```
▼ {kind: "identitytoolkit#SignupNewUserResponse", idToken: "eyJhbGc.
  "prueba@gmail.com", refreshToken: "AEu4IL2tSv1ZYPc1ZVsRH8nUrsjQol-
    kind: "identitytoolkit#SignupNewUserResponse"
  idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlNWNlZDZlNDBkY2QxZWZmNI
  email: "prueba@gmail.com"
  refreshToken: "AEu4IL2tSv1ZYPc1ZVsRH8nUrsjQol-GqoQvD1pCCH6yPIFX"
  expiresIn: "3600"
  localId: "B4Fchll1FvToWz39nmTDtSAhvc13"
▶ __proto__: Object
```

Le doy al botón de login:

LOGIN

prueba@gmail.com

Password

Recordar mi usuario [¿No tienes cuenta?](#)

INGRESAR

Angular: De cero a e | Swe

localhost:4200/home

Aplicaciones Gmail YouTube M

home works!

Le doy a atrás:

LOGIN

prueba@gmail.com

Password

Recordar mi usuario [¿No tienes cuenta?](#)

INGRESAR

Guard para poner la ruta:

Tal como está, cualquiera puede entrar en el *LocalStorage*, borrar la información del *localStorage* e ir al /home.

Sources	Network	Performance	Memory	Application	Security	Audits
<input type="button" value="Filter"/>						
Key	Value					
token	eyJhbGciOiJSUzI1NiIsImtpZCI6ImQ3NjM1YWI2NDZIMDdmZDE5OWY3NGlwMTZhOTU0Mzk					

Para proteger esta ruta debemos hacer **guard**. Lo creamos mediante el angular-cli:

```
iMac-de-Jose:appLoginFireBase jsersan$ ng g guard guards/auth
Your global Angular CLI version (8.3.21) is greater than your local
version (7.3.9). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".
? Which interfaces would you like to implement? (Press <space> to select, < > to toggle all, <> to invert selection)
> CanActivate
  o CanActivateChild
  o CanLoad
```

Nos preguntará si lo queremos activar y le decimos que sí con la barra espaciadora.

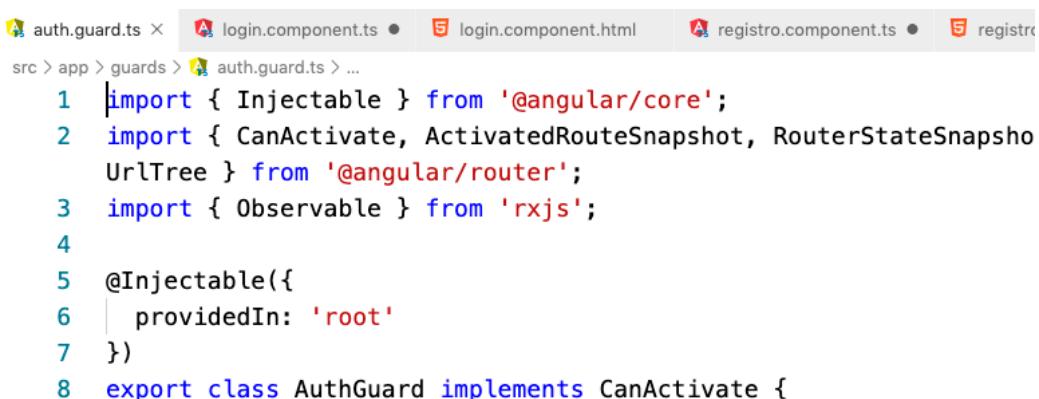
```
To disable this warning use "ng config -g cli.warnings.versionMismatch false".
? Which interfaces would you like to implement?
>● CanActivate
  o CanActivateChild
  o CanLoad
```

Presionamos ENTER:

```
iMac-de-Jose:appLoginFireBase jsersan$ ng g guard guards/auth
Your global Angular CLI version (8.3.21) is greater than your local
version (7.3.9). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".
? Which interfaces would you like to implement? CanActivate
CREATE src/app/guards/auth.guard.spec.ts (346 bytes)
CREATE src/app/guards/auth.guard.ts (456 bytes)
iMac-de-Jose:appLoginFireBase jsersan$ █
```

Lo abrimos:



```
auth.guard.ts ✘ login.component.ts ● login.component.html ✘ registro.component.ts ● registro.component.html ✘ src > app > guards > auth.guard.ts > ...
1 import { Injectable } from '@angular/core';
2 import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot,
  UrlTree } from '@angular/router';
3 import { Observable } from 'rxjs';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class AuthGuard implements CanActivate {
```

Esto es un servicio que implementa el *canActivate* que permite activar las rutas por las que va a pasar:

```

8  export class AuthGuard implements CanActivate {
9    canActivate(
10      next: ActivatedRouteSnapshot,
11      state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> |
12      boolean | UrlTree {
13        return true;
14      }
15  }

```

Estos campos son opcionales:

- **next**: siguiente ruta a la que el usuario quiere navegar.
- **state**: estado actual

Quitando los elementosopcionales, tenemos:

```

1  import { Injectable } from '@angular/core';
2  import { CanActivate } from '@angular/router';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class AuthGuard implements CanActivate {
8
9    canActivate(): boolean {
10      return true;
11    }
12  }
13

```

Crearemos en **auth.service.ts** una rutina que me permita saber si el usuario está autenticado o no, basta con ver si hay información en el userToken:

```

81  estaAutenticado(): boolean {
82
83    return this.userToken.length > 2;
84
85  }

```

En el auth.guard.ts:

```

auth.guard.ts • login.component.ts login.component.html registro.component.ts
src > app > guards > auth.guard.ts > ...
1 import { Injectable } from '@angular/core';
2 import { CanActivate } from '@angular/router';
3 import { AuthService } from '../services/auth.service';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class AuthGuard implements CanActivate {
9
10  constructor( private auth: AuthService) {}
11
12  canActivate(): boolean {
13    return this.auth.estaAutenticado();
14  }
15
16 }
17

```

Ahora en el **app-routing.module.ts** debo especificar qué rutas necesitan autenticación:

```

8 const routes: Routes = [
9   { path: 'home' , component: HomeComponent },
10  { path: 'registro', component: RegistroComponent },
11  { path: 'login' , component: LoginComponent },
12  { path: '**', redirectTo: 'registro' }
13 ];
14

```

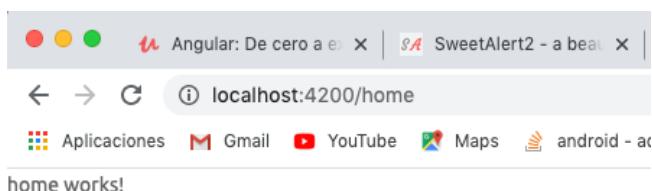
Sólo va a ser el home:

```

5 import { LoginComponent } from './pages/login/login.component';
6 import { AuthGuard } from './guards/auth.guard';
7
8
9 const routes: Routes = [
10  { path: 'home' , component: HomeComponent, canActivate: [AuthGuard] },
11  { path: 'registro', component: RegistroComponent },
12  { path: 'login' , component: LoginComponent },
13  { path: '**', redirectTo: 'registro' }
14 ];

```

Lo probamos y vemos que estamos en el home:



Para ver si lo está haciendo colocamos un **console.log** en el servicio:

```

8  export class AuthGuard implements CanActivate {
9
10    constructor( private auth: AuthService) {}
11
12    canActivate(): boolean {
13      console.log('Guard!!!');
14      return this.auth.estaAutenticado();
15    }
16
17  }

```

El resultado:

```

⚠ [Deprecation] Resource requests whose URLs contained both removed whitespace (`\n`, `\r`, `\t`) characters and less-than characters from places like element attribute values in order to load these resources.
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
Guard!!!
>

```

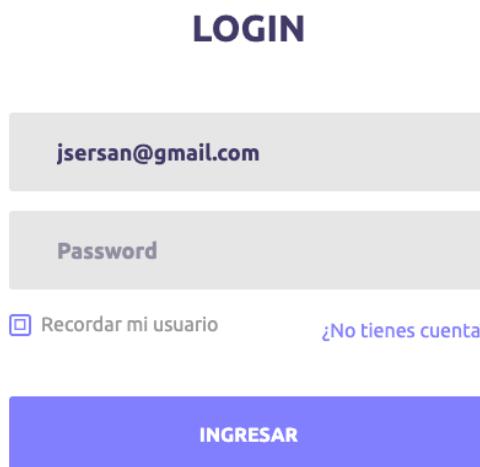
Voy a hacer que si no está autenticado se redirija al login:

```

' '
8  export class AuthGuard implements CanActivate {
9
10    constructor( private auth: AuthService,
11                 | | | | | private router: Router) {}
12
13    canActivate(): boolean {
14      console.log('Guard!!!');
15
16      if ( this.auth.estaAutenticado() ) {
17        return true;
18      } else {
19        this.router.navigateByUrl('/login');
20        return false;
21      }
22    }

```

Tras hacer un login, me redirige a esta página a través del guard:



[Logout- terminar sesión:](#)

Lo único que tenemos que hacer es remover el token.

```

21  logout() {
22    localStorage.removeItem('token');
23    console.log('Sesión finalizada');
24 }
```

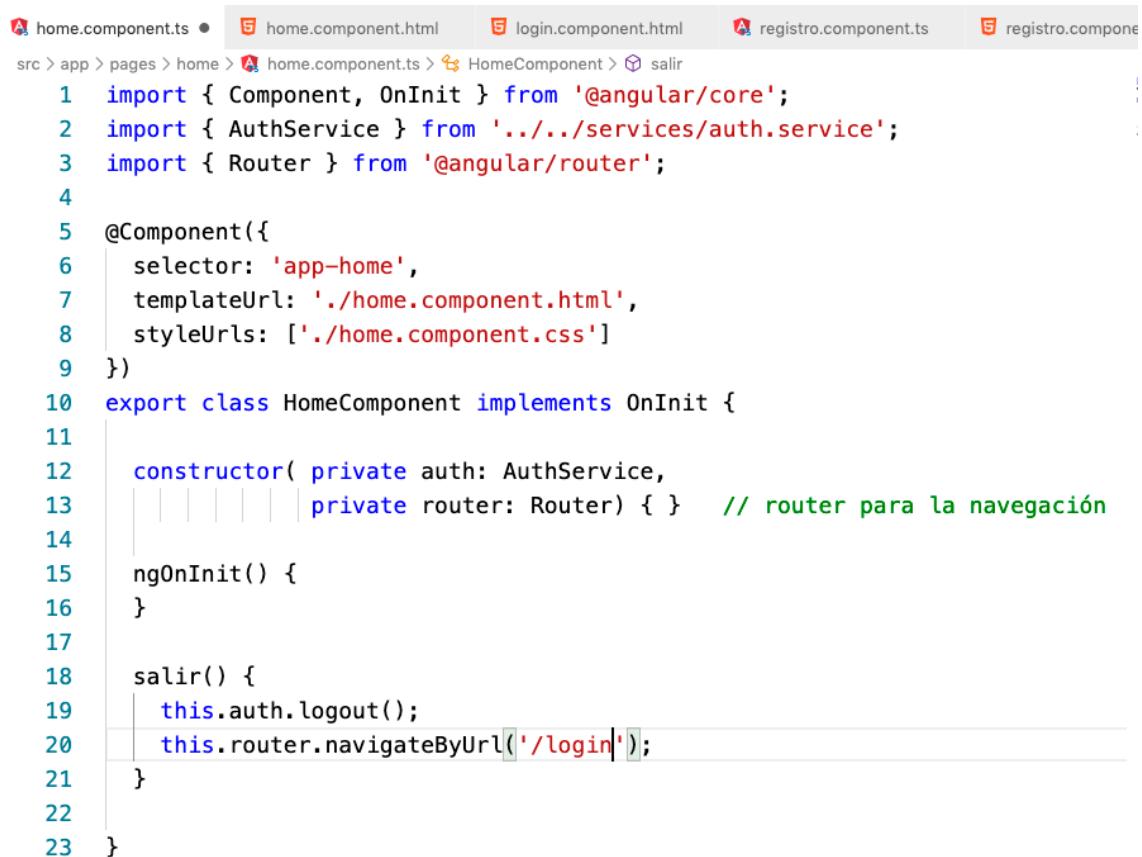
Le cambiamos el diseño al home:



```

home.component.ts      home.component.html      login.component.html      registro.component.ts
src > app > pages > home > home.component.html > div.md-5
1  <div class="md-5">
2
3    <h1>Mi Aplicación secreta</h1>
4    <hr>
5
6    <button (click)="salir()" class="btn btn-outline-danger">
7      Salir
8    </button>
9
10 </div>
```

En [home.component.ts](#) implementamos la función “salir”:



```

home.component.ts ●      home.component.html      login.component.html      registro.component.ts      registro.compon
src > app > pages > home > home.component.ts > HomeComponent > salir
1  import { Component, OnInit } from '@angular/core';
2  import { AuthService } from '../../../../../services/auth.service';
3  import { Router } from '@angular/router';
4
5  @Component({
6    selector: 'app-home',
7    templateUrl: './home.component.html',
8    styleUrls: ['./home.component.css']
9  })
10 export class HomeComponent implements OnInit {
11
12   constructor( private auth: AuthService,
13               private router: Router) {} // router para la navegación
14
15   ngOnInit() {
16   }
17
18   salir() {
19     this.auth.logout();
20     this.router.navigateByUrl('/login');
21   }
22 }
```

Lo probamos:

Tras ingresar:

LOGIN

jsersan@gmail.com

.....

En el LocalStorage:

Cerramos la sesión y desaparece el token:

LOGIN

jsersan@gmail.com

Password

Recordar mi usuario [¿No tienes cuenta?](#)

INGRESAR

Sources	Network	Performance	Memory	Application	Security	Audits
 Filter						
Key	Value					
email	jsersan@gmail.com					

Mejorar la validación del token:

Vamos a manejar la fecha de validez del mismo. Llegado a ese punto el token dejará de ser válido, aunque lo tengamos físicamente. Ingresamos con un usuario:

Entro en el mapa del RXJS

```
{kind: "identitytoolkit#VerifyPasswordResponse", localId: "YVYAfzb0Qk...eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlNWNLZDZlNDBkY2QxZW...nGMr0CyRuplu7NIfo"
  kind: "identitytoolkit#VerifyPasswordResponse"
  localId: "YVYAfzb0QkZJ5Dg9ebSYxOenrAl1"
  email: "jsersan@gmail.com"
  displayName: ""
  idToken: "eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlNWNLZDZlNDBkY2QxZWZmNDA3M...
  registered: true
  refreshToken: "AEu4IL1Z7p4n9oxjGN9m8ZNJQb-3HgZgvKiBWjhBijpuTrJuSbIk...
  expiresIn: "3600"
  ► __proto__: Object
```

Existe un parámetro que indica que expira en una hora. Para comprobarlo en la consola hacemos:

```
> let hoy= new Date();
< undefined
> hoy
< Fri Dec 27 2019 09:30:13 GMT+0100 (hora estándar de Europa central)
> |
```

Si quiero incrementar el valor expiresIn:

```
> let hoy= new Date();
< undefined
> hoy
< Fri Dec 27 2019 09:30:13 GMT+0100 (hora estándar de Europa central)
> hoy.setSeconds(3600);
< 1577439000901
> |
```

Lo convierto a string:

```
> hoy
< Fri Dec 27 2019 09:30:13 GMT+0100 (hora estándar de Europa central)
> hoy.setSeconds(3600);
< 1577439000901
> hoy.getTime().toString();
< "1577439000901"
```

En mi fichero del servicio **auth.service.ts** en typescript:

```

65  private guardarToken( idToken: string ) {
66
67    this.userToken = idToken;
68    localStorage.setItem('token', idToken);
69
70    let hoy = new Date();
71    hoy.setSeconds(3600);
72
73    localStorage.setItem('expira', hoy.getTime().toString());
74
75  }

```

En la fecha de hoy ya tengo cuando va a expirar el token. En la parte de leerToken() tengo que validarla en **estaAutenticado()**:

```

87  estaAutenticado(): boolean {
88
89    if ( this.userToken.length < 2 ){
90      return false;
91    }
92
93    const expira = Number(localStorage.getItem('expira'));
94
95    const expiraDate = new Date();
96    expiraDate.setTime(expira);
97
98    if (expiraDate > new Date()) {
99      return true;
100    } else {
101      // Ya sé que el token no es válido
102      return false;
103    }
104
105  }

```

Para probarlo inicio sesión:

Key	Value
token	eyJhbGciOiJSUzI1NiIsImtpZCI6ImN
expira	1577439780180
email	jsersan@gmail.com