Proyecto angular2

# Requisitos

1. **NPM** (node package manager) es el gestor de paquetes JavaScript de NODE.JS

* <https://www.npmjs.com/>
* Versión > 3.8

1. **Node.js®** es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo.

* https://nodejs.org/es/

1. Angular cli:

* <https://github.com/angular/angular-cli>
* 4. npm install -g typescript
* NODE JS
* https://nodejs.org/es/
* EDITOR
* visual studio code
* ANGULAR
* https://github.com/angular/angular-cli
* TYPESCRIPT
* npm install -g typescript
* https://www.typescriptlang.org/play/index.html
* GIT

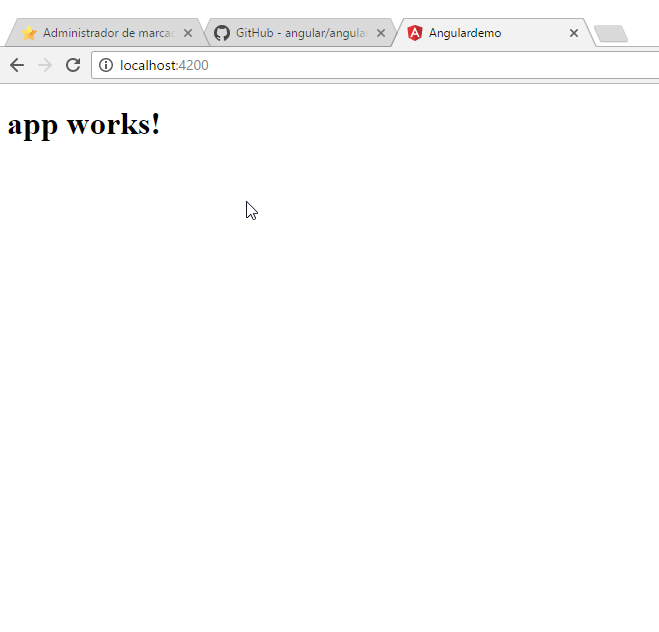
# 

# Crear un Proyecto

**```sh**

ng new PROJECT\\_NAME  
  
cd PROJECT\\_NAME  
  
ng serve  
  
ng serve --host 0.0.0.0 --port 4201

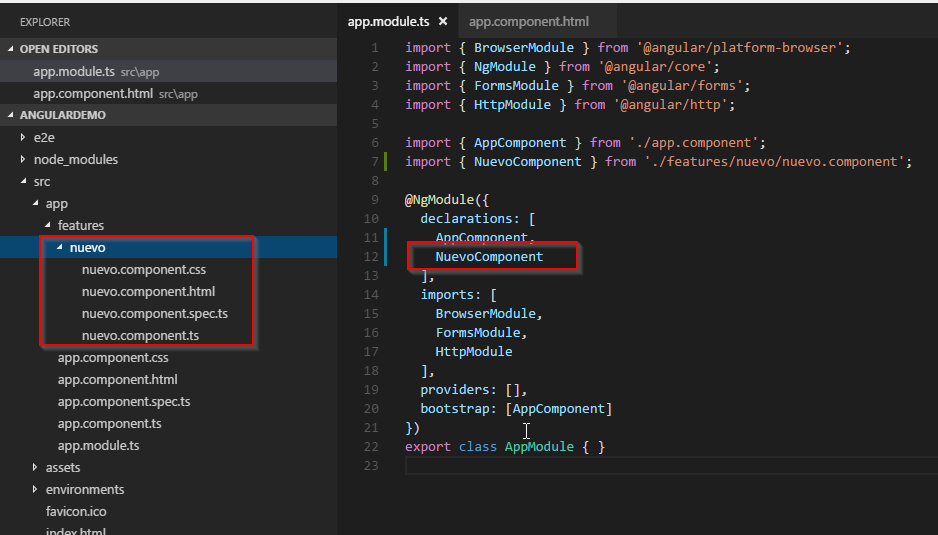
**```**



# Crear un componente

mkdir src/app/features

ng g component features/nuevo #Agregar un nuevo componente en una ubicación a partir de app.



Esto genera el esquema básico de un componente y lo registra en src/app/app.module.ts (¡PILAS MUY IMPORTANTE!), nótese como angular cli añade la palabra component.

Para comprobar que funciona nuestro nuevo componente, podemos ponerlo en el componente donde inicia la aplicación:

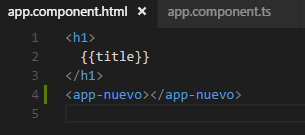
src/app/app.component.ts

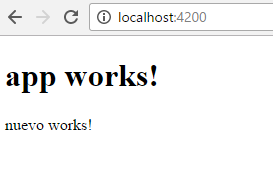
1. Lo importamos en el componente



1. Usamos el selector dentro de nuestra plantilla, y ¡listo!

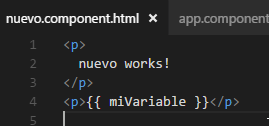






# Crear variables en el componente y mostrar en la vista

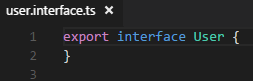


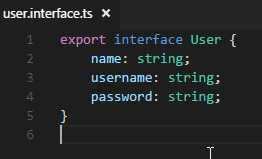




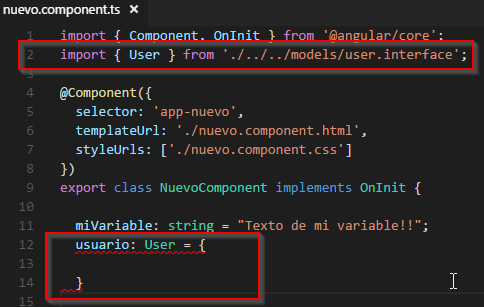
# Crear variables en el componente y mostrar en la vista, ¡Objeto!

ng g interface models/User interface

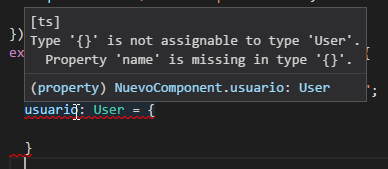




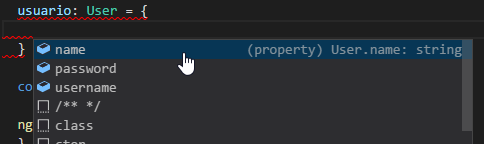
Ahora vamos a crear un Objeto de tipo User en nuestro “nuevo component”



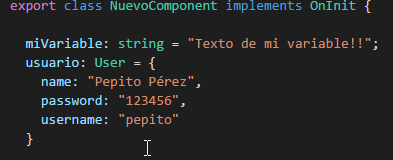
Hacemos el import de nuestra interface, y procedemos a crear la variable usuario de tipo User, notamos que automáticamente TypeScript nos resalta que hay un error, ya que debemos implementar los campos de la interfaz.



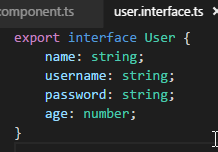
El mensaje empieza a decirnos que la propiedad “name” no está en nuestro objeto.

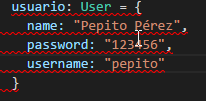


Al hacer Ctrl + espacio, el editor nos sugiere los campos necesarios, así como su tipo. Procedemos a llenarlos.

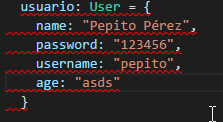


Si agregamos un nuevo campo a la interfaz por ejemplo la edad, tenemos, un error.



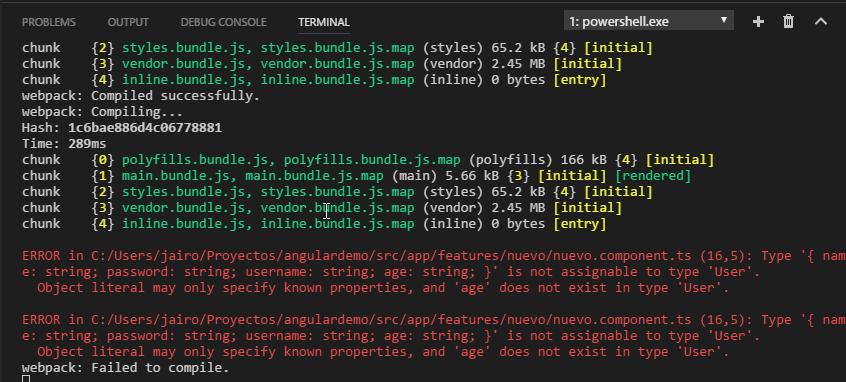


A propósito vamos a asignar la variable age, con un string

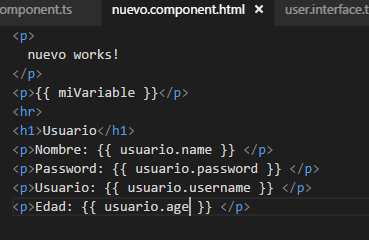


Si guardamos, el compilador nos arrojará un error, tanto en el navegador como en la consola.





Ahora, en el template de nuestro componente,

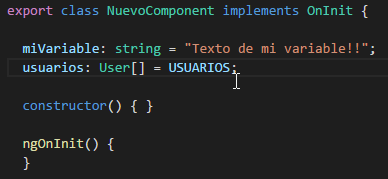




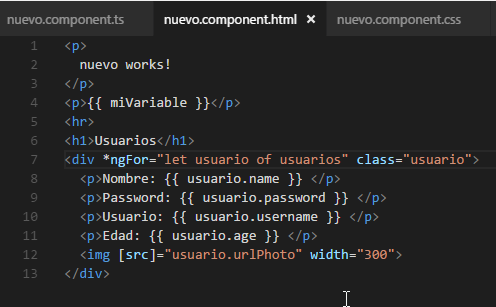
Si queremos que el ejemplo sea un poco más completo, podemos declarar una constante como un array de usuarios, podría ser así:



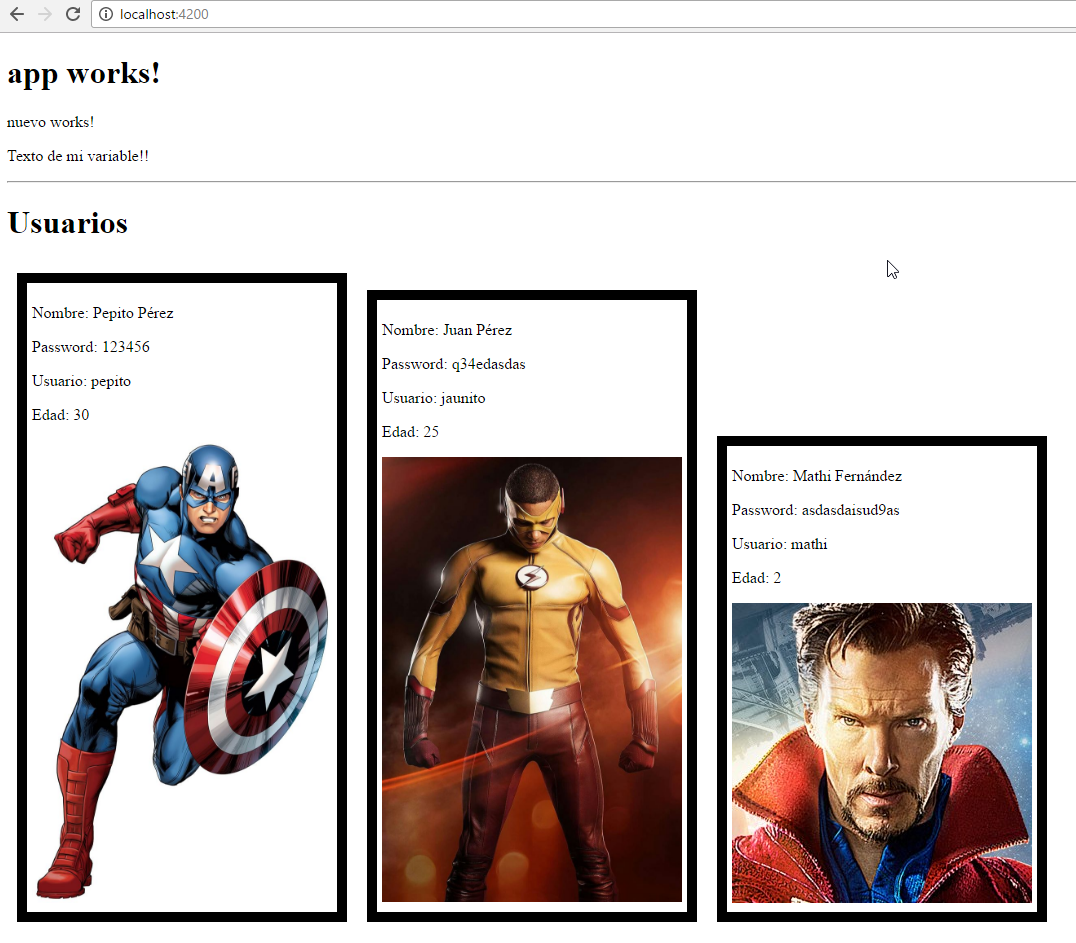
Cambiamos el objeto usuario por usuarios



Y ajustamos el template para que se muestren los datos, usando la directiva \*ngFor



Si todo va bien:



Agregar variables tipo Input en un componente, para que interactúe con otros componentes

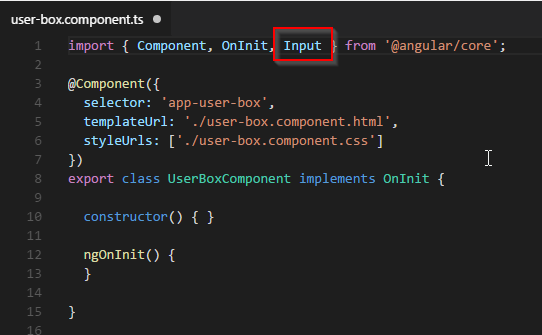
Ahora vamos a convertir las cajas como un componente que usemos en nuestro “nuevo componente”, lo llamaremos “user-box”

**```sh**

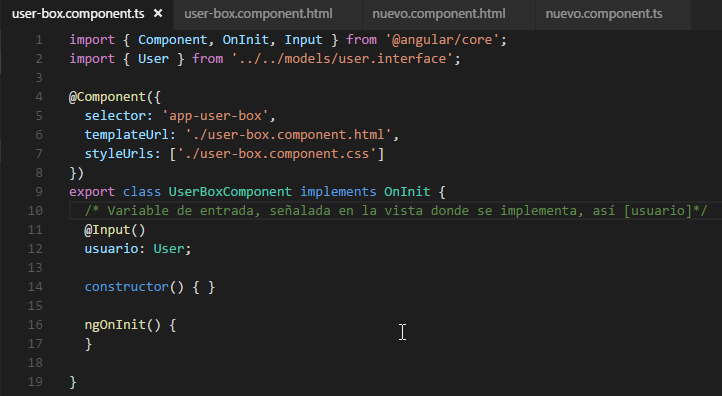
ng g component features/user-box

**```**

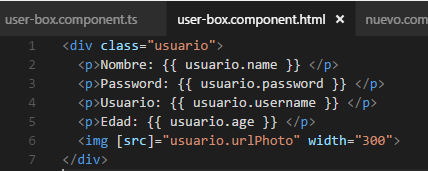
Ahora agregamos el decorador “input” de @angular/core



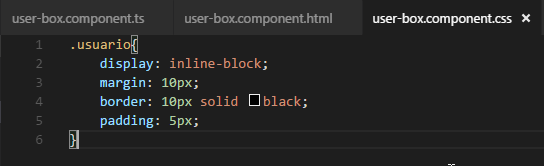
Ya que queremos interactuar con este componente, agregamos una variable de tipo @Input()



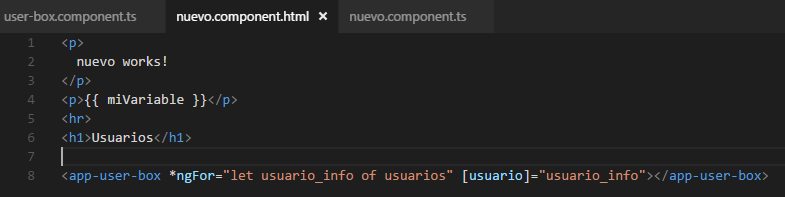
Pasamos el fragmento html que queremos sacar de “nuevo componente” a “user-box”, quitando el \*ngFor

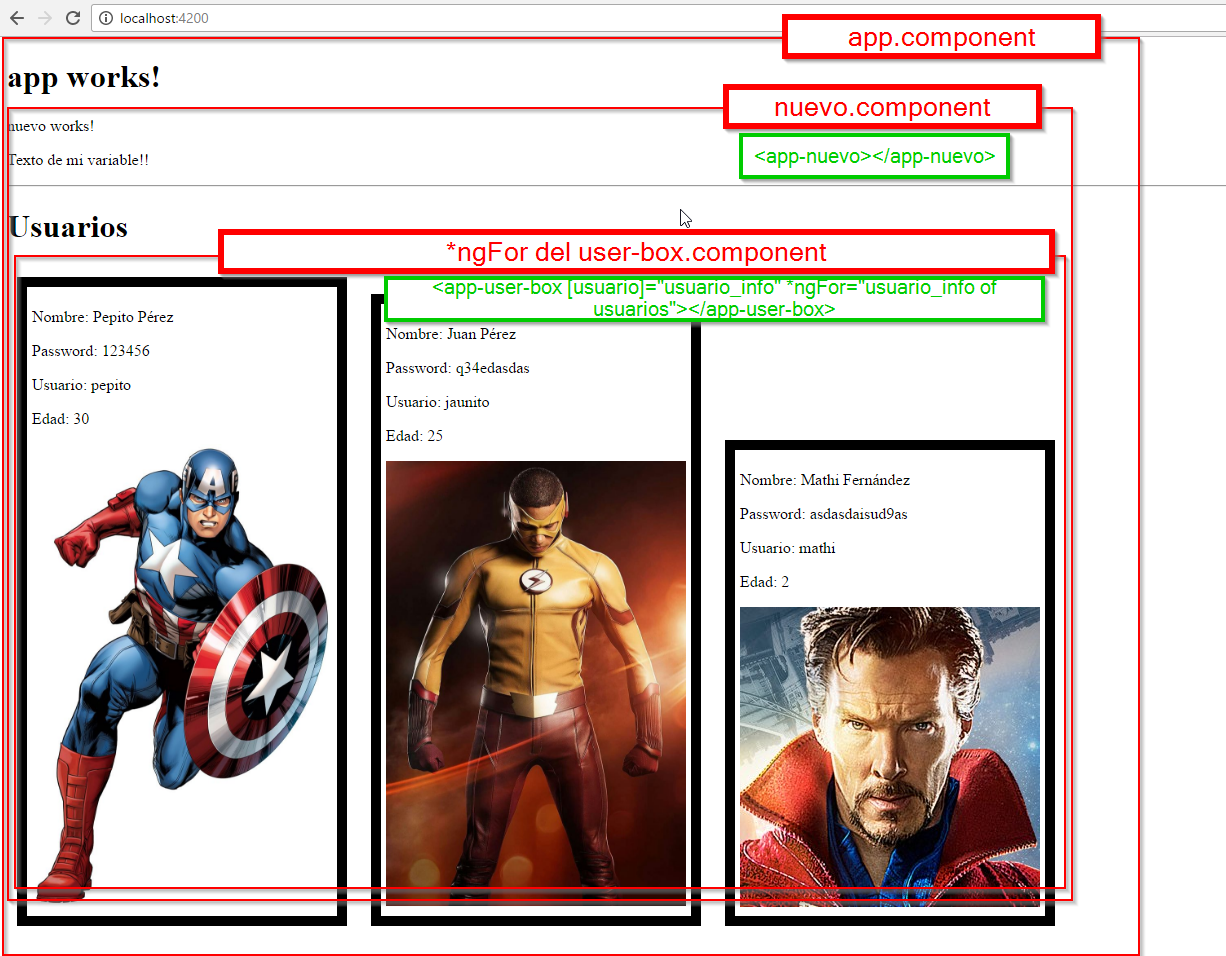


Y pasamos la clase css a user-box.component.css



Y ahora implementamos el componente box, importándolo en “nuevo componente” y agregándolo al template





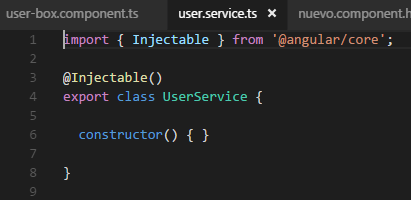
Consulta con un API

Creamos el servicio user:

**```sh**

ng g service services/user

**```**



Este nos genera un archivo que implementa Injectable de @angular/core, y que además no tiene un template. Agregamos la propiedad users, y dos métodos, uno para obtener y otro para añadir.



Ahora, vamos a generar un componente que nos permita visualizar la información de nuestro user.service

**```sh**

ng g component features/user

**```**

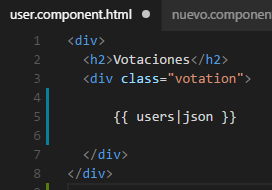
Nuestro servicio nos permite realizar acciones sobre un objeto User ficticio por ahora, en nuestro user.component implementamos nuestro UserService.

Aprovechando la inyección de dependencias, podemos “inyectar” nuestro servicio en el constructor del componente:

constructor(private UserService : UserService) { }

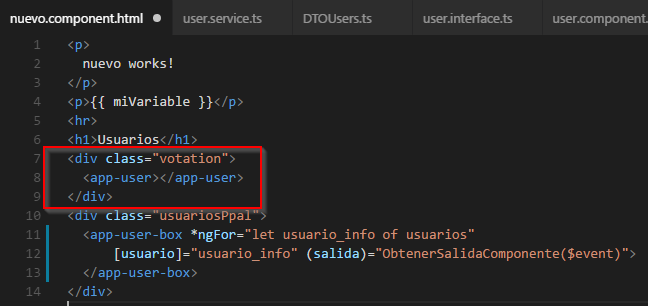
Así, tenemos acceso a todos los métodos públicos de nuestro servicio

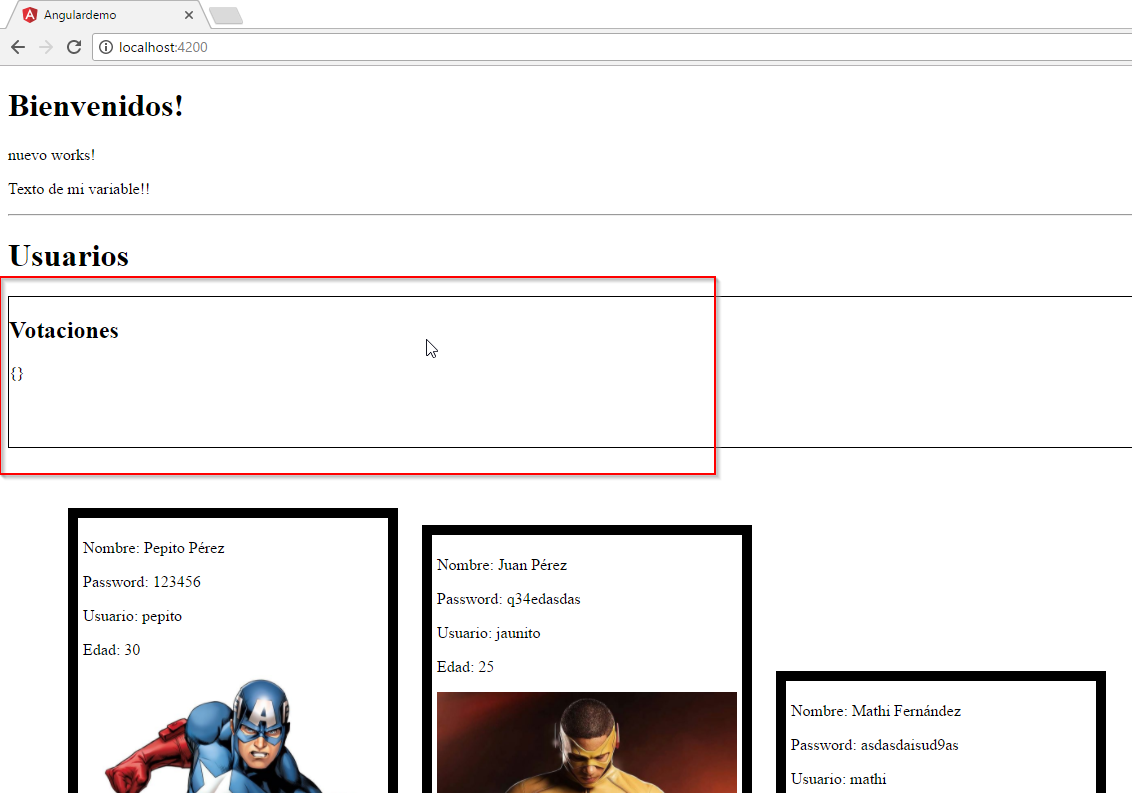


****

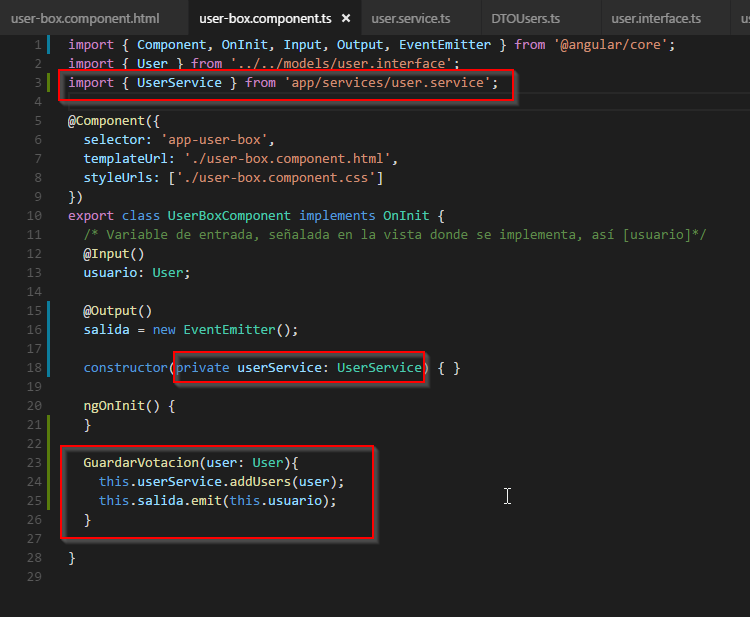
Ahora, este componente nos mostrará la cantidad de veces que presionamos sobre un Usuario, lo implementamos en nuevo componente (Debemos agregarlo como un provider).

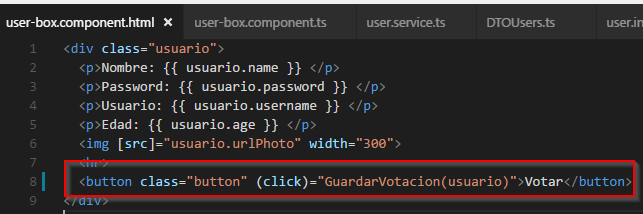


****

****

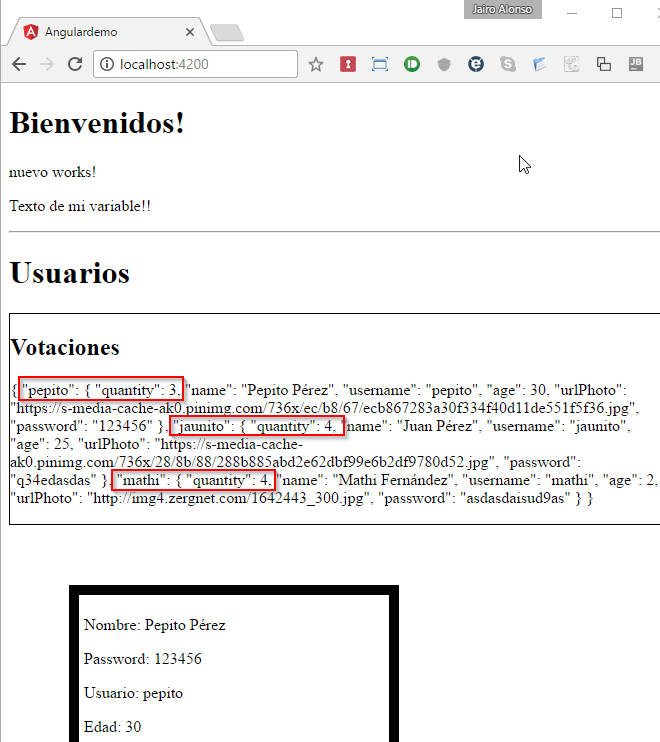
El componente user, que muestra las votaciones, no realizará ninguna acción, solo mostrará la información que otro componente esté accionando, en este caso, Desde user-box.componente, agregaremos un botón, para agregar la votación.



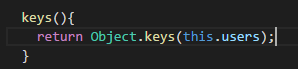




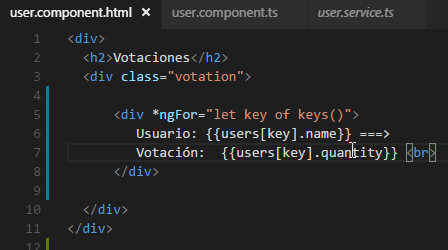
Ahora al presionar varias veces sobre Votar veremos la cantidad

****

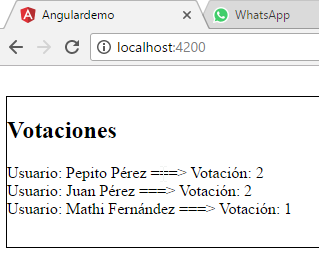
Para mejorar la lectura de los datos, debemos recorrer el objeto, sin embargo, no podemos iterar un objeto con ngFor en angular2, entonces podemos obtener las llaves y recorrerlas



Así



Obteniendo lo siguiente:

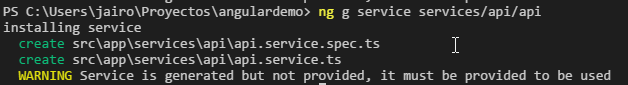


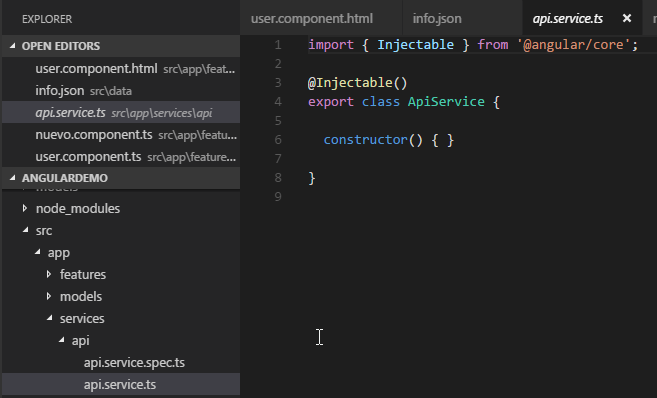
**COMUNICANDO CON UN API**

Importamos HttpModule de angular/http, en app.module

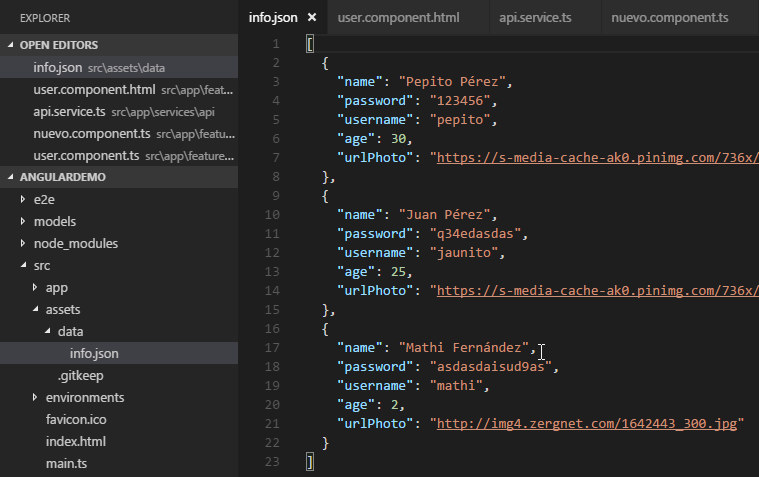


Creamos un servicio

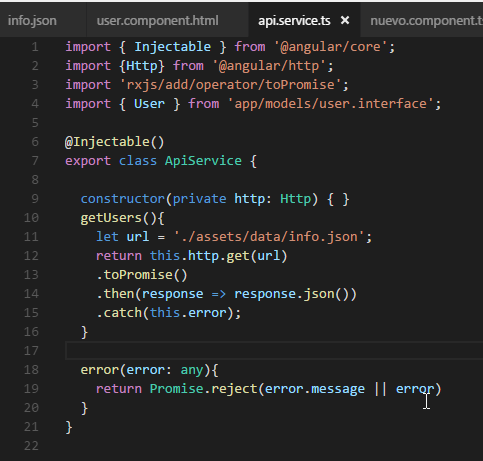




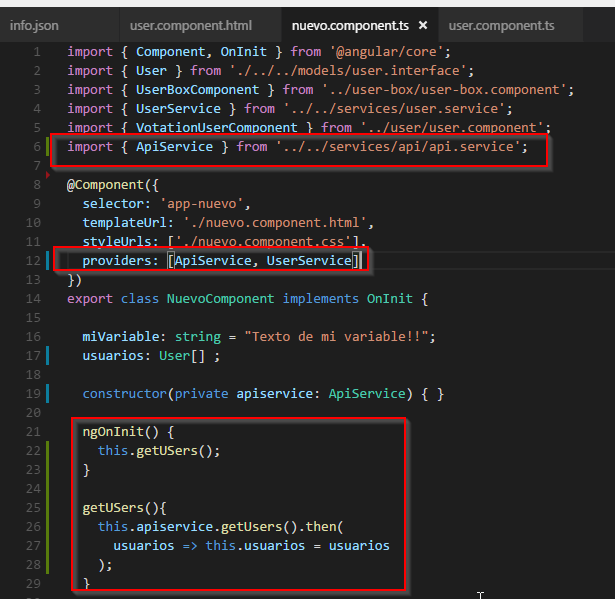
Vamos a emular el API, para ellos creamos un archivo info.json en el directorio data dentro de assets

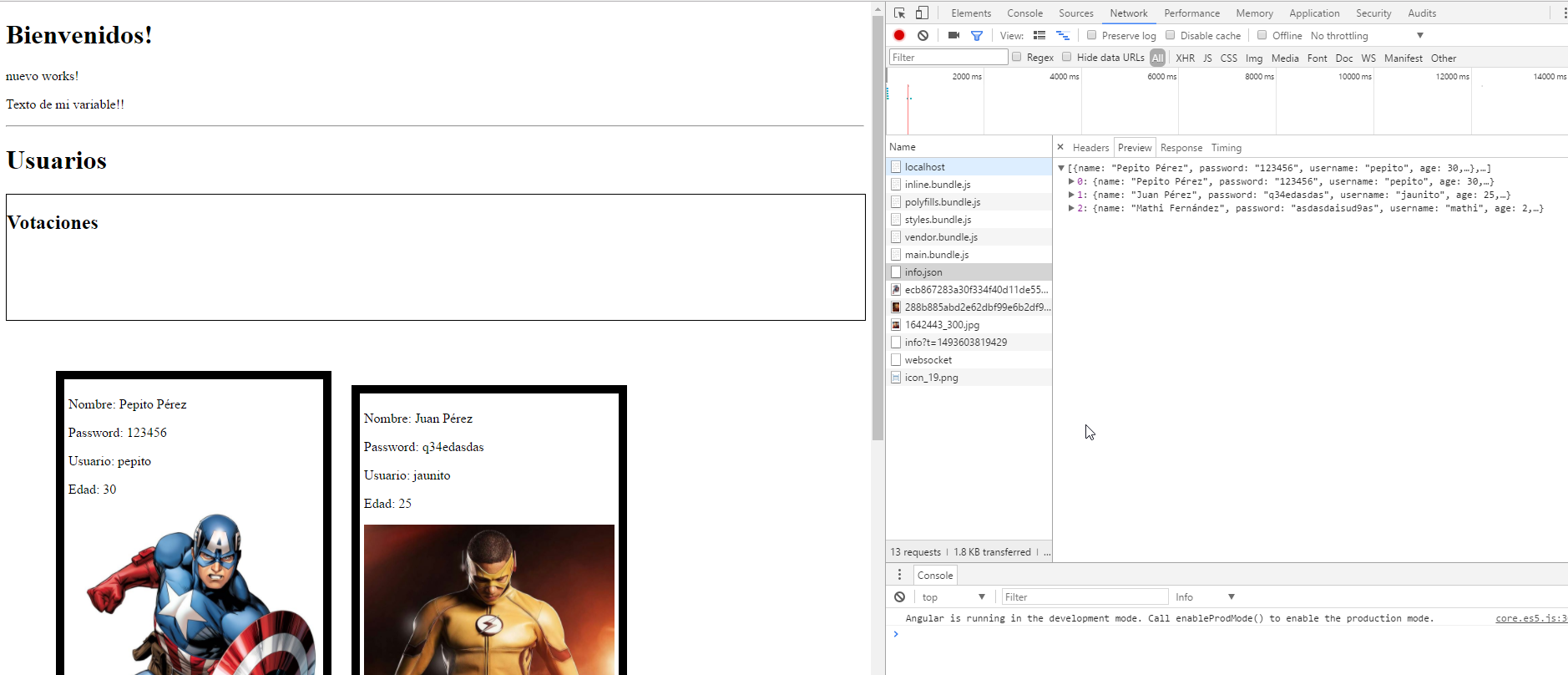


Y ajustamos api.service

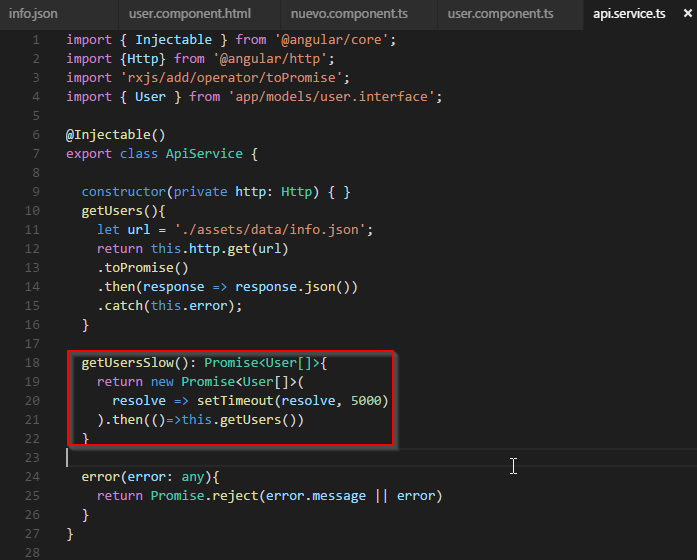


Finalmente, vamos a reemplazar el objeto que usamos para pintar los usuarios en nuevo.component





Podemos emular una conexión lenta



Así comprobamos que la página cargue sin importar que el servidor no halla respondido con la petición de los usuarios.

**OUTPUTS**

Debemos importar Output y EventEmitter de angular/core:

import { Component, Output, EventEmitter } from '@angular/core';

@Output()

salida = new EventEmitter();

**Luego en un método donde se haga un cambio,**

GuardarVotacion(){

this.salida.emit(this.algunObjeto);

}

<button class="button" (click)="GuardarVotacion()">Votar</button>

**Ahora en el componente que lo usa agregamos el output ()**

<app-user-box \*ngFor="let usuario\_info of usuarios" [usuario]="usuario\_info" (salida)="ObtenerSalidaComponente($event)"></app-user-box>

En el componente debemos agregar el método ObtenerSalidaComponente

ObtenerSalidaComponente(event){

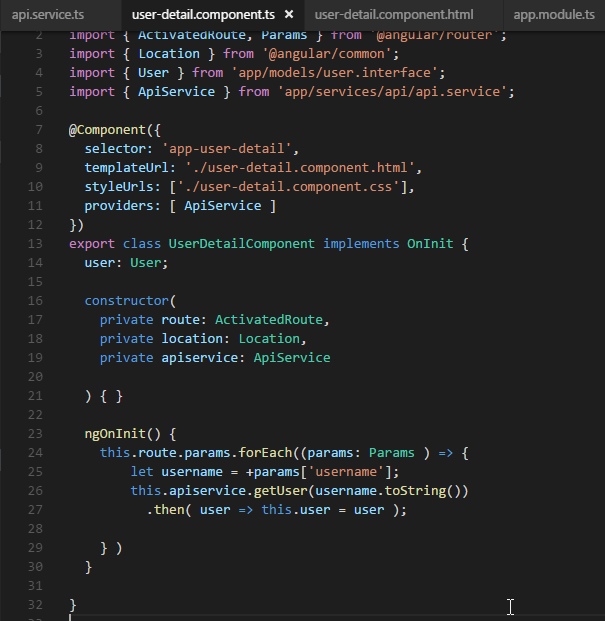
console.log("Recibido " + JSON.stringify(event));

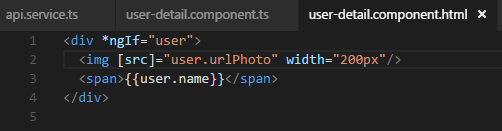
}

**ROUTING**

* 1. Generamos un componente de bienvenida
  2. Generamos un componente de detalle de cada usuario

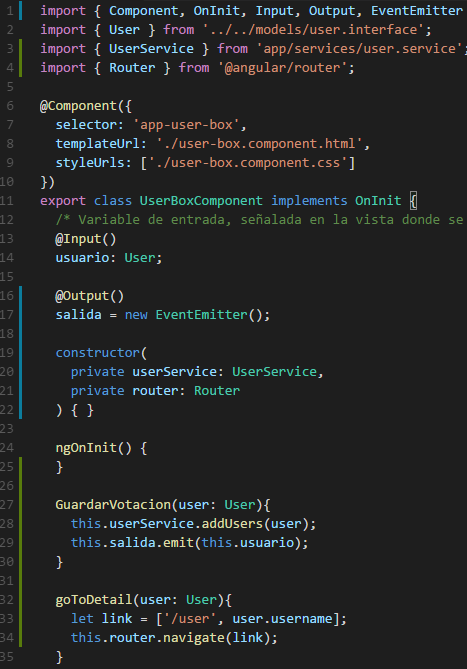
Importamos ActivatedRoute, Params de angular/router y Location de angular/common



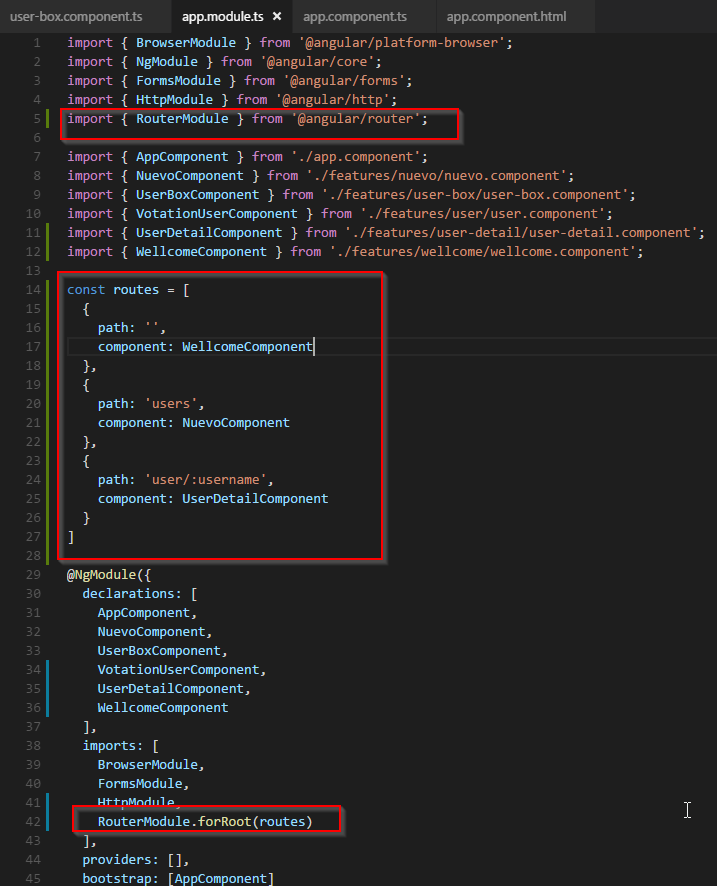


Ahora, en el componente user-box, añadiremos la acción de ir al detalle,

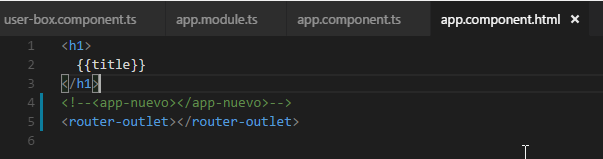
Agregamos de angular/router: Router y un método de ir al detalle (goToDetail)



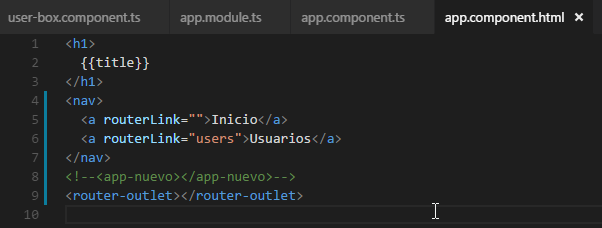
Para poder usar las rutas debemos ajustar nuestra aplicación, empezamos con app.module



Y en app.component.ts, quitamos los componentes que hallamos implementado y ponemos <router-outlet></router-outlet>

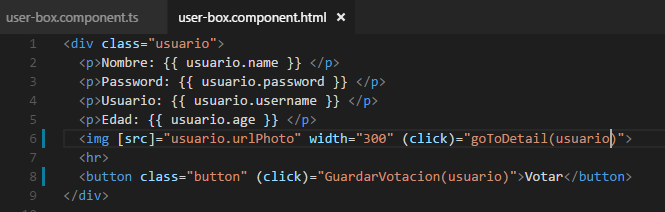


Agregamos un menú, usando la directiva routerLink, que resuelve las rutas por nosotros



Ahora, nos falta en user-box, agregar la acción a la vista, recordemos que el componente







SEGURIDAD

Creamos un auth service

import { Injectable } from '@angular/core';

import { Router } from '@angular/router';

@Injectable()

export class AuthService {

user: AUser[] = USERS;

session: boolean = false;

constructor(private router: Router) { }

login(user: AUser) {

let user\_ext = this.user.find(u => u.email == user.email);

if (user\_ext && user\_ext.password == user.password) {

localStorage.setItem("user", JSON.stringify(user\_ext));

this.session = true;

let link = ['/'];

this.router.navigate(link);

}

}

logout() {

this.session = false;

localStorage.removeItem("user");

this.redirect();

}

redirect() {

let link = ["/login"];

this.router.navigate(link);

}

check(){

if(localStorage.getItem("user") == null){

this.session = false;

this.redirect();

}else{

this.session = true;

}

}

getSession(){

return this.session;

}

getUser(){

return localStorage.getItem("user");

}

}

export interface AUser {

email: string,

password: string

}

const USERS: AUser[] = [

{

email: 'jairo@hotmail.com',

password: '123456'

},

{

email: 'nata@hotmail.com',

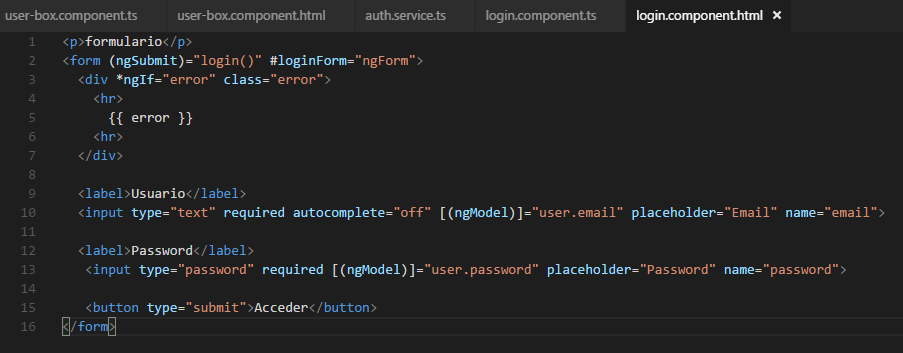
password: '1234'

}

]

Ahora creamos un componente Login e importamos nuestro auth service y usamos las directivas de formularios.

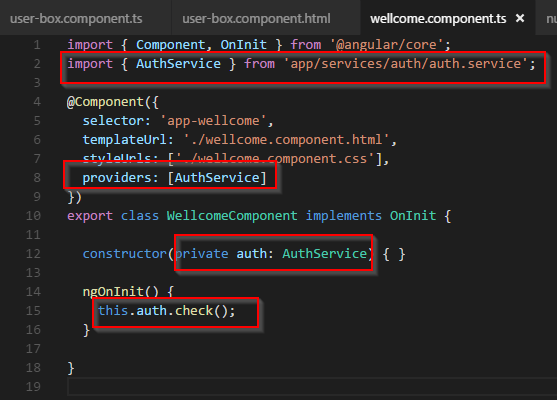




Agregamos el path login, en app.modules



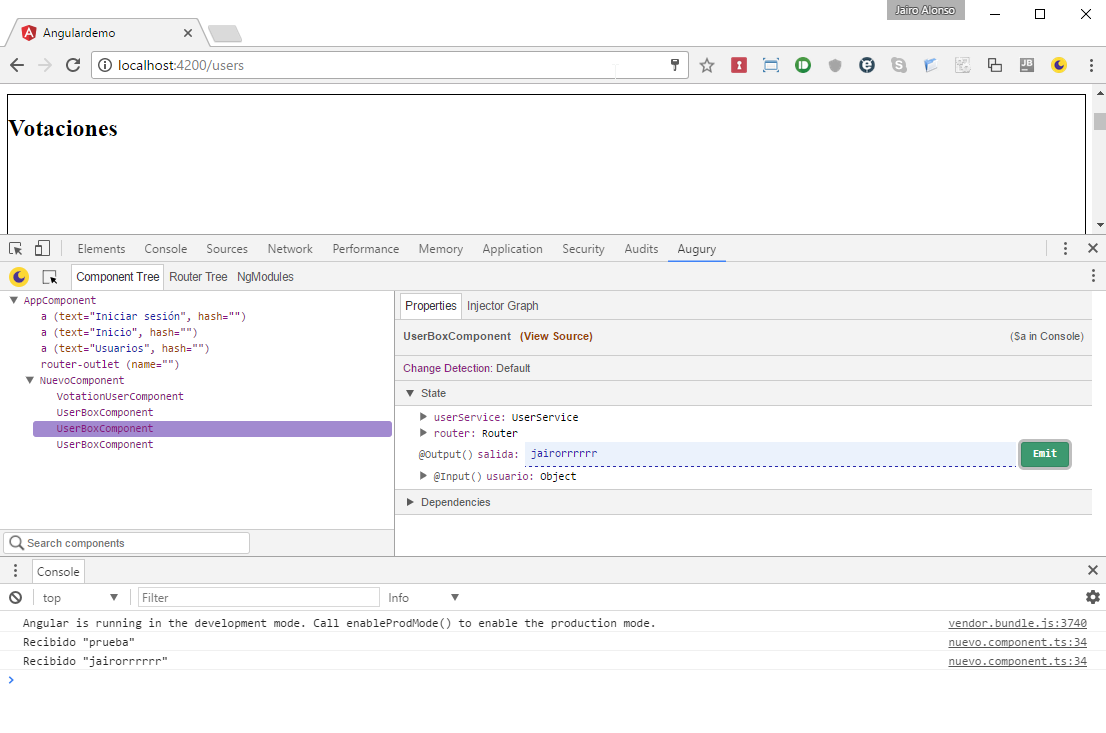
Implementando el modelo de sesión en un componente



Ahora agregamos las rutas de iniciar y cerrar sesión:



ANGULAR AUGURY



OBSERVABLES

<http://reactivex.io/>

