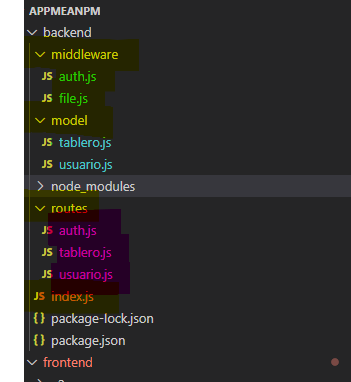
**Taller NodeJS/Express MongoDB Angular**

-Mockup

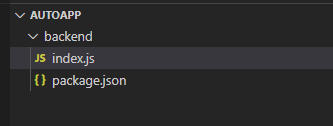
-Model DataBase

-Controlador

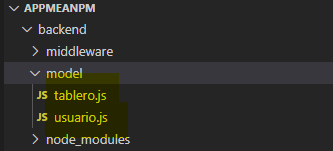
-Front (Angular)

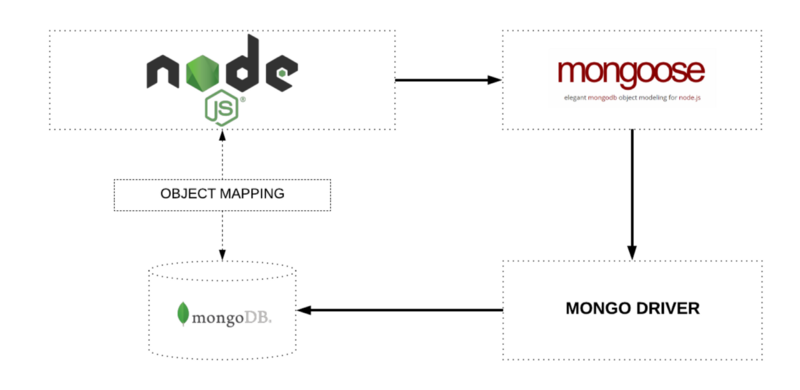


1. Crear la carpeta APPMEANPM y el subfolder backend (inicializar proyecto NodeJS)
2. APPMEANPM\backend>**npm init** ( se genera automáticamente archivo package.json en la carpeta backend , adicionalmente creamos el archivo index.js.

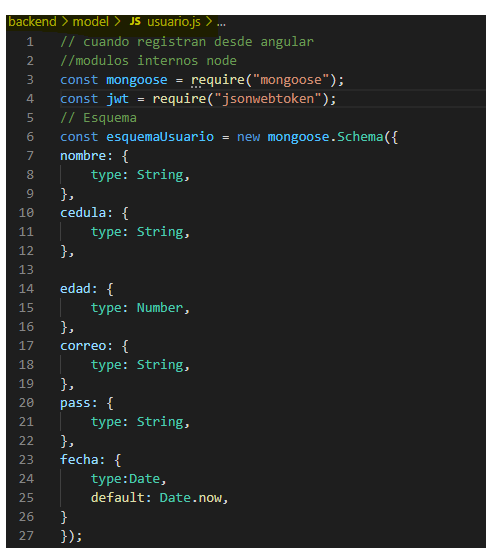


1. Instalar **express** al proyecto APPMEANPM\backend >**npm install express**
2. Instalar librería **mongoose** para administrar MONGODB APPMEANPM\backend >**npm install mongoose**
3. Instalar **jsonwebtoken** para encriptar información APPMEANPM\backend >**npm install jsonwebtoken**
4. Creamos un subcarpeta llamada model con los archivos usuario.js tablero.js modulos que van a guardar las colleciones en MONGODB en la carpeta backend. () similar Entity Java Persistence API

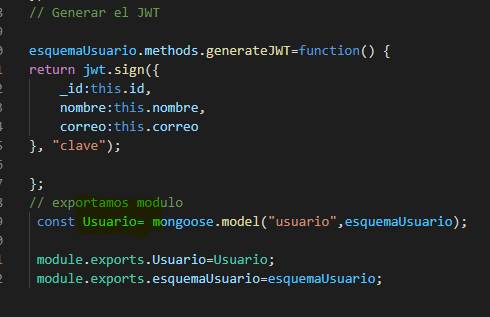


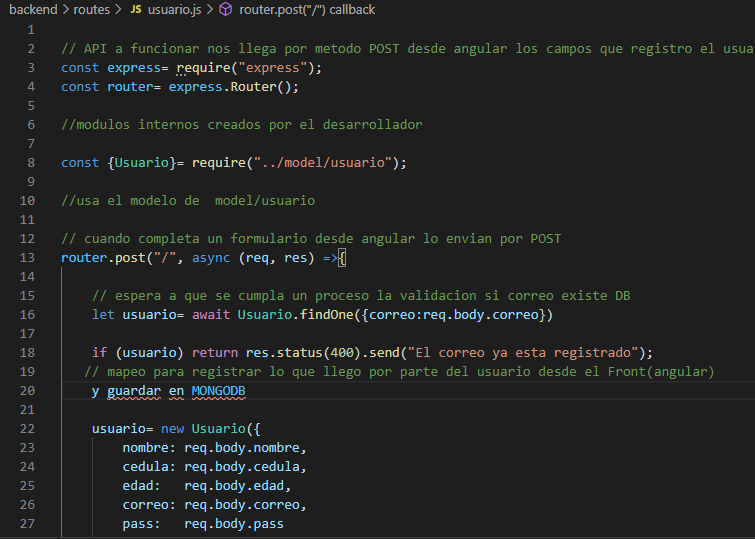


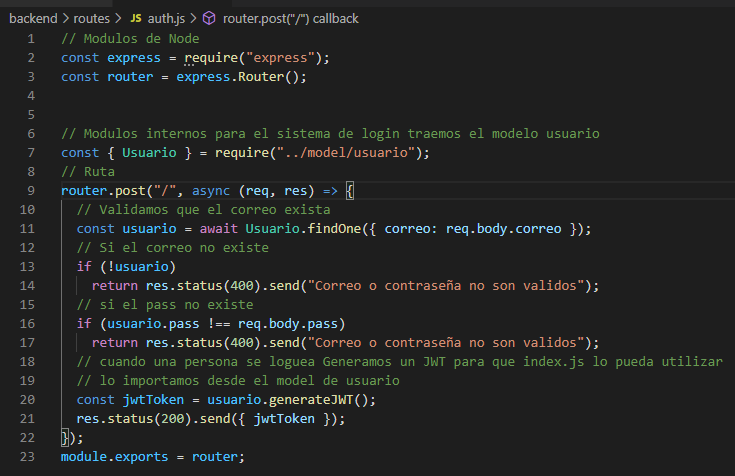
1. Crear el modelo de datos en el archivo usuario.js , los módulos internos y el esquema del usuario.



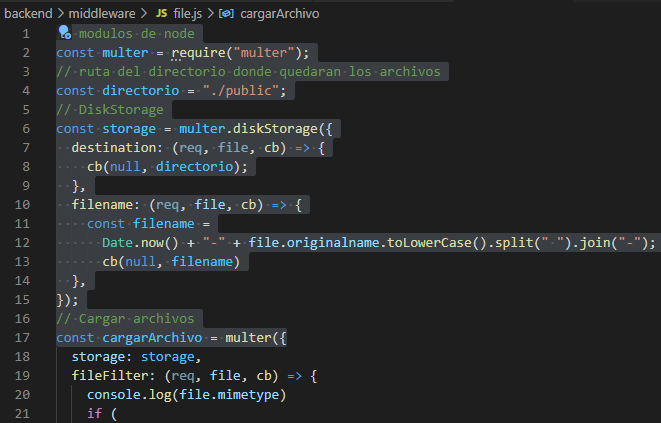
1. Generar en el esquema de usuario.js el método de encriptación JasonWebToken con la exportación del modulo con el esquema del modelo base de datos.



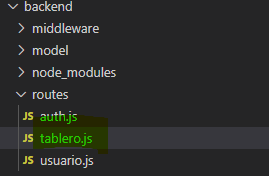
1. CONTROLADOR Realizar la rutas URL de la API(como se va a registrar un usuario CRUD) para la colección usuario.js crear la subcarpeta routers y exportar el modulo router(propiedad de EXPRESS)
2. En la subcarpeta routers crear el archivo auth.js para validar y crear token de usuario y exportar el modulo router(propiedad de EXPRESS)

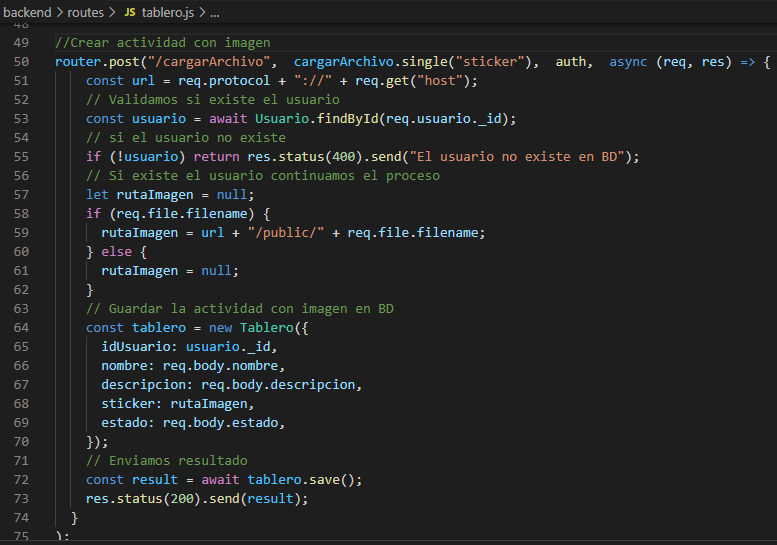


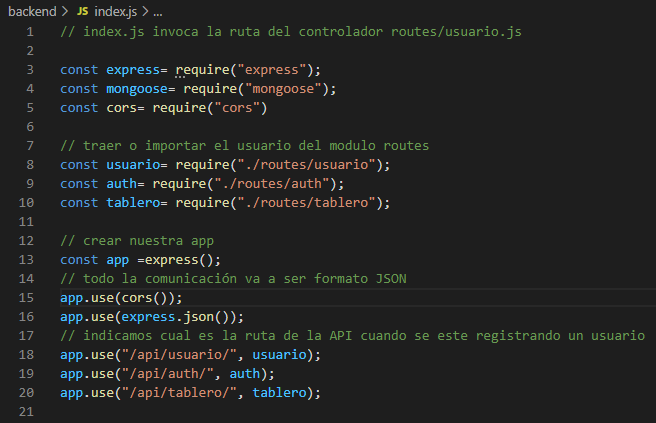
1. En la subcarpeta middleware crear el archivo auth.js para el traspaso de información de usuario con el token de validación creando una función.
2. En la subcarpeta middleware crear el archivo file.js para el traspaso de información y subir los sticker(archivos imagen) instalamos libreria **MULTER** APPMEANPM\backend>npm i multer.

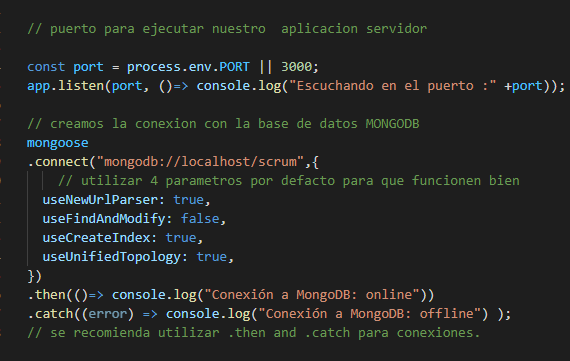


1. CONTROLADOR Realizar la rutas URL de la API(como se va crear (**POST**), listar(**GET**), actualizar(**PUT**)borrar(**DELETE**) actividad con o sin imagen CRUD) para la colección tablero.js crear la subcarpeta routers y exportar el modulo router(propiedad de EXPRESS)





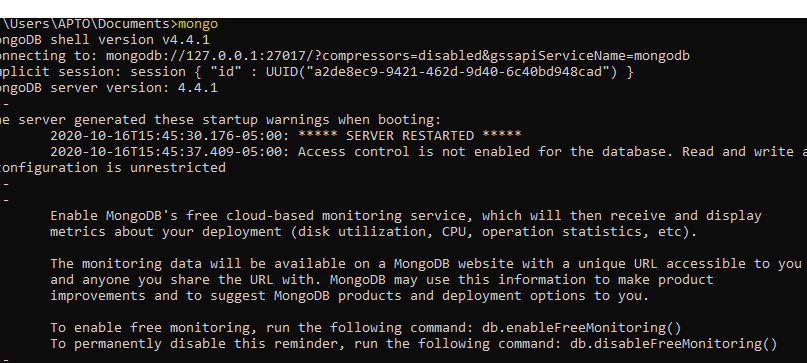
13.En el archivo modulo index.js que es la creación de nuestra APP con el pool de la conexión de la la Base de Datos en MongoDB.



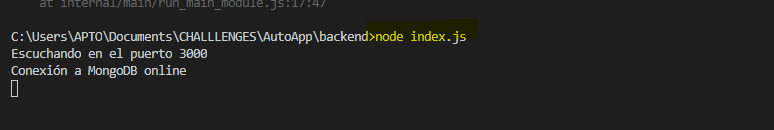
1. Se conecta a mongod mongo ejecutar compass



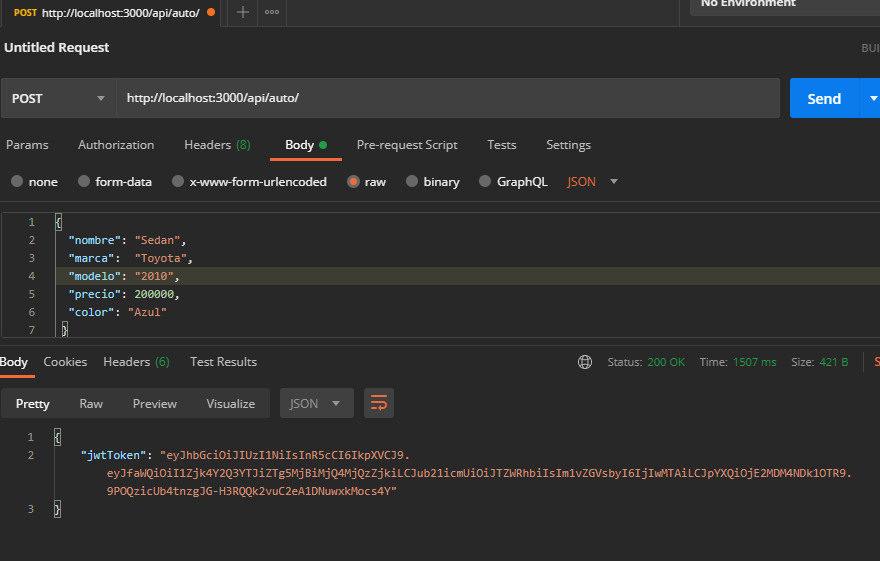
1. Se conecta mongo



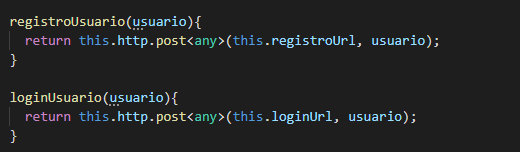
1. Desplegar API con conexión a MongoDB AutoApp\backend>node index.js

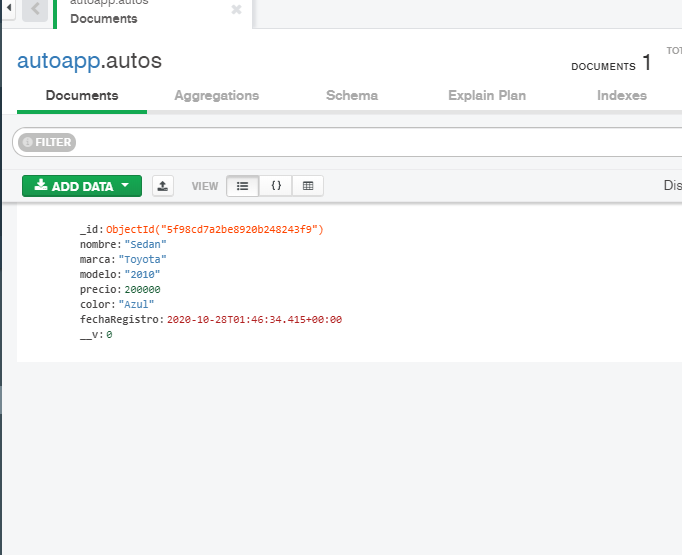


1. Postman convierte el documento auto de la colección autoapp a jsonwebToken

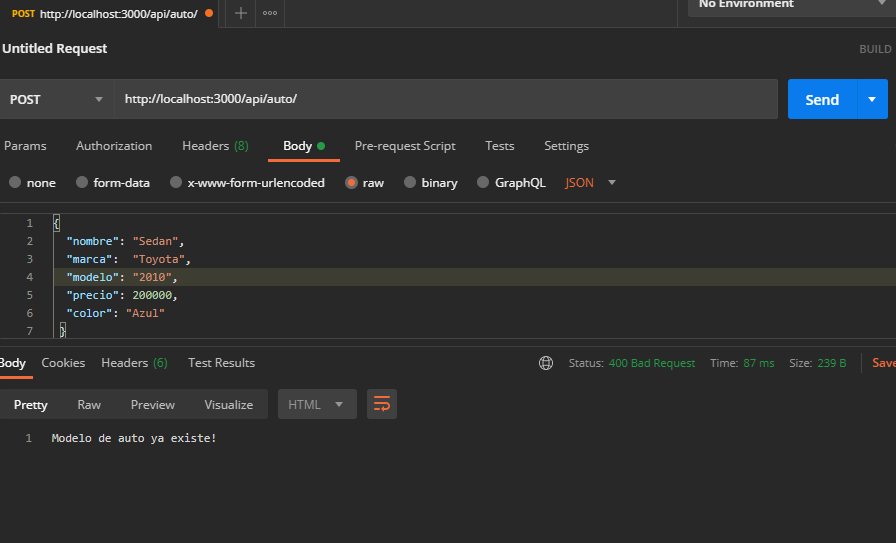


EN ANGULAR método en el constuctor del archivo Auth de servicio

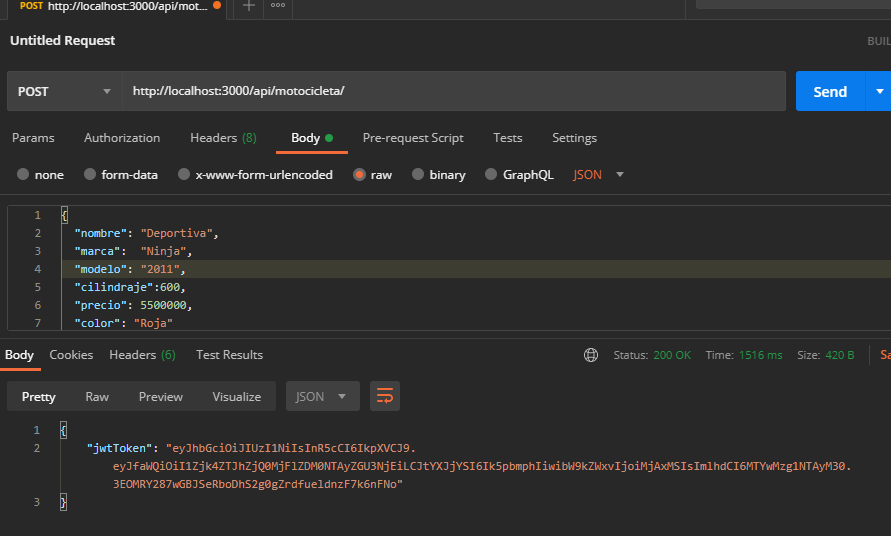


14. Se crea desde la app la base de datos MongoDB con la colección auto automáticamente

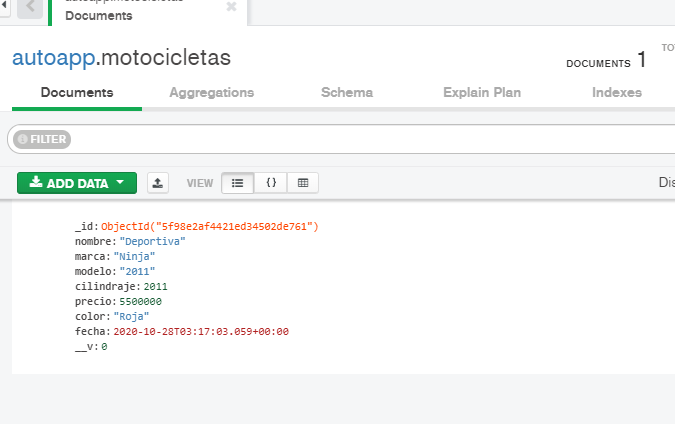
15. Enviar nuevamente el mismo modelo 2010 se genera mensaje de “error”



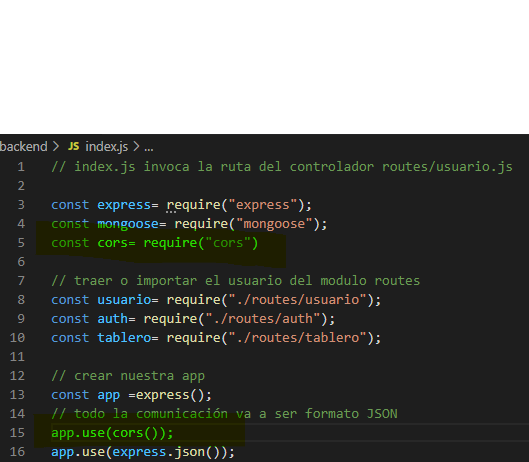
16. Postman convierte el documento auto de la colección autoapp a jsonwebToken



17. Se crea desde la app la base de datos MongoDB con la colección auto automáticamente



Para empezar nuestro frontend con ANGULAR comunicar las diferentes servidores y urls de nuestro backend con el frontend en nuestro caso localhost:3000 en backend con localhost:4000 en frontend instalamos libreria **CORS**APPMEANPM\backend>npm i cors.

En el index.js en el backend los configuramos 

ANGULAR tiene herramienta para optimizar FLEX y GRID LAYOUT

Instalar APPMEANPM\frontend>npm install -s @angular/flex-layout @angular/cdk

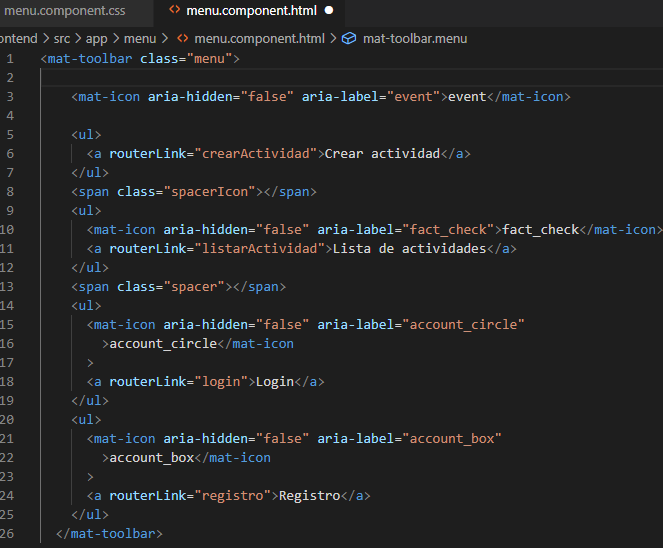
found 1 high severity vulnerability (Para arreglar vulnerabilidad)

APPMEANPM\frontend>npm audit fix

<https://material.io/resources/icons/?style=baseline> (Ejemplos de iconos a utilizar)

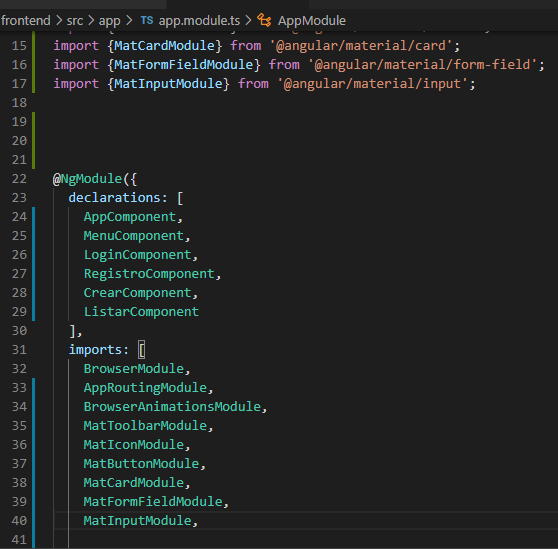
menú componente.css

menú.componente.html (**asociamos los iconos al menú**)



Registrar un usuario registro.component.ts y que se pueda loguear para generar JsonWebToken

App.module importamos material/card'; form-field'; /input';



Se trabaja con formulario se import {FormsModule} from '@angular/forms';

Se adiciona al imports: [FormsModule,

Para utilizar javascript consumir el backend comunicar con el backend

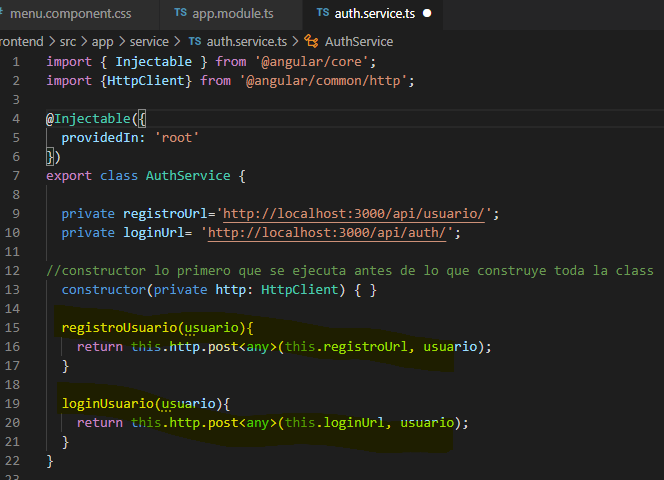
APPMEANPM\frontend>ng g s service/auth

Para comunicarnos con backend API REST necesitamos importar import {HttpClientModule} from '@angular/common/http/';

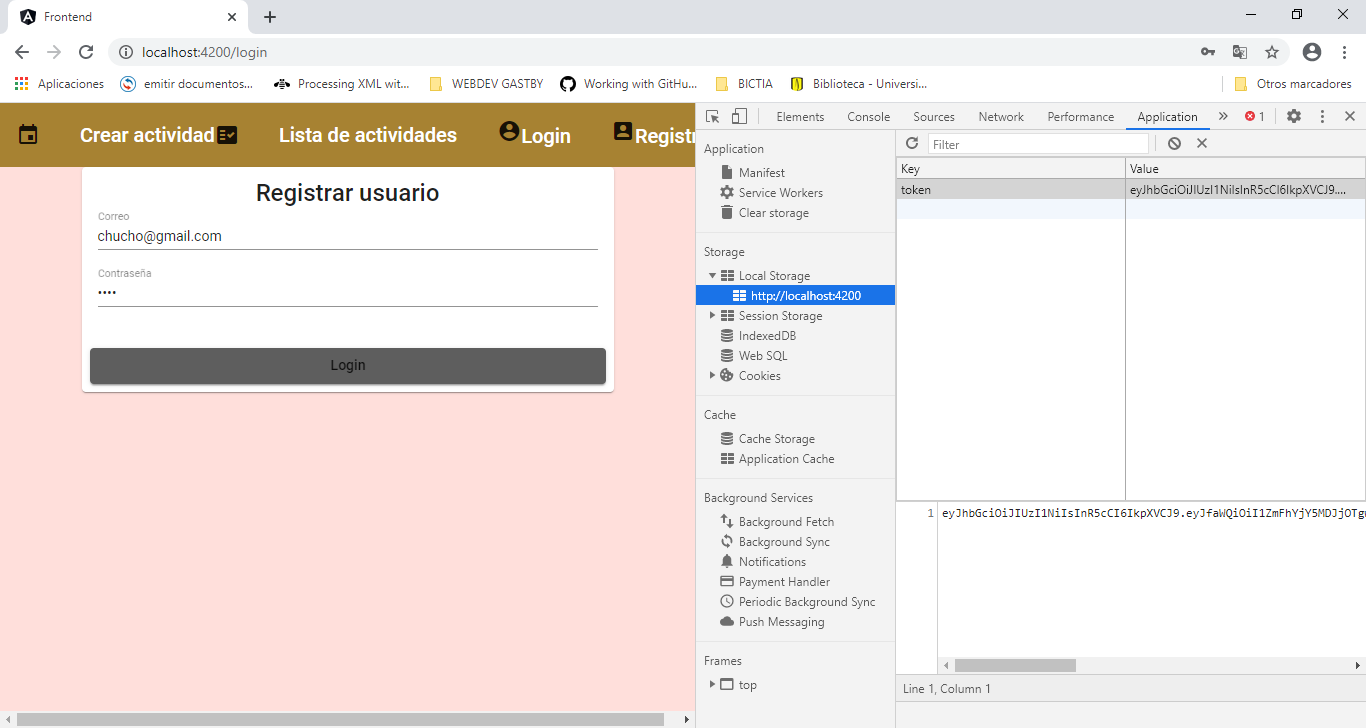
LOS SERVICIOS VAN EN PROVIDER

AuthService Provee un servicio y se configura en providers: [AuthService],

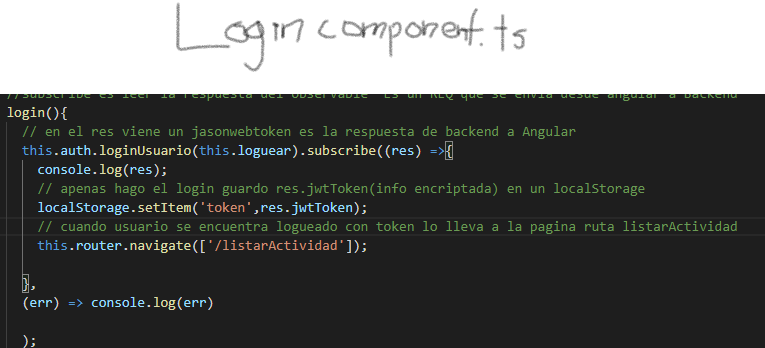
Boton de registrar usuario en



Cuando usuario se loguea se genera un JasonWebToken que se guarda con un local storage



Apenas me logue veo mi lista de actividades , cuando me registro me debe llevar al login.(2rutas)

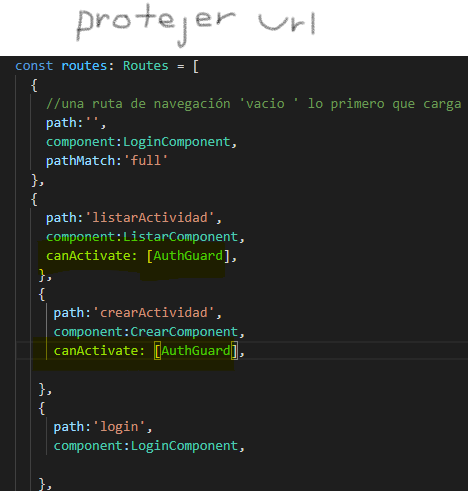


Si escriben cualquier url <http://localhost:4200/crearActividad> al usuario le debe salir login

Necesitamos un sistema que nos indique que el usuario este logeado, ANGULAR tiene una librería GUARD para proteger si tenemos un token permita ver las URL

Instalar \APPMEANPM\frontend>ng g guard guard/auth

Nuestro sistema de guardia para proteger URL , ejemplo registro y login son publicas (no necesitan guard)



En Backend con el POSTMAN para subir una imagen usuario logueado se pega token en Autorization con el bearer.

Se debe instalar un servicio APPMEANPM\frontend>ng g s service/tokenInterceptor

<https://material.angular.io/components/expansion/api>

en APP module la importamos

import { MatExpansionModule } from '@angular/material/expansion'; y lo declaramos en import

Los cambios de los botones de estado se hacen desde listar.component.html

// es un css accordion display ejemplo estilo cards se vean pequeños en la mitad

display: flex;

justify-content: center;

align-items: center;

<div class="accordion\_display"> los divs no organizan los espacios

// css accordion size

    <div class="accordion\_size">

Mat accordion propiedad multi=”true” multiples acordiones

      <mat-accordion multi="true" \*ngIf="lista.length"> si hay una lista es un array con 2 actividades lista.length 2 actividades para pintar los accordions

        <mat-expansion-panel \*ngFor="let lista of lista" hideToggle="false">

recurro a un for para iterar la lista y mostrar las actividades y cada una pintarla hideToogle de material

          <mat-expansion-panel-header>

            <mat-panel-title>

              <span class="text\_center">{{ lista.nombre }}</span>

Interpolación el front muestra el valor de la variable

              <span class="spacer"></span> spacer aplicar un espacio

  3 BOTONES

<button

background-color: #3cb44b;

color: #ffffff;

                class="btDone" Dependiendo el estado cambia el color

                \*ngIf="lista.estado == 'terminada'" botón aparece si lista estado dice terminada

                mat-raised-button

              >

                {{ lista.estado }}

              </button>

              <button

                class="btInProgress"

                \*ngIf="lista.estado == 'iniciada'"

                mat-raised-button

              >

                {{ lista.estado }}

              </button>

              <button

                class="btToDo"

                \*ngIf="lista.estado == 'asignada'"

                mat-raised-button

              >

                {{ lista.estado }}

              </button>

            </mat-panel-title>

Botones Parte de la Descripción de nuestra actividad

          </mat-expansion-panel-header>

          <mat-panel-description>

            {{ lista.descripcion }}

          </mat-panel-description>

Nos permite hacer acciones tener eventos

          <mat-action-row>

            <button

              class="btSelectTodo" Me va aparecer la palabra con el estado

              mat-button

              (click)="cambiarEstado(lista, 'asignada')"

Click para activar cambiarEstado recibo lista la actividad (hacer login)que estamos ahora y el estado que quiero cambiar a asignada

            >

              Asignada

            </button>

            <button

              class="btSelectInProgress"

              mat-button

              (click)="cambiarEstado(lista, 'iniciada')"

Click para activar cambiarEstado recibo lista la actividad (hacer login)que estamos ahora y el estado que quiero cambiar a iniciada

            >

              Iniciada

            </button>

            <button

              class="btSelectDone"

              mat-button

              (click)="cambiarEstado(lista, 'terminada')"

Click para activar cambiarEstado recibo lista la actividad (hacer login)que estamos ahora y el estado que quiero cambiar a terminada

            >

              Terminada

            </button> el botón seria un icono

            <mat-icon para darle a un icono un evento

              color="primary"

              (click)="eliminar(lista)".ed

              aria-label="delete\_forever"

              >delete\_forever</mat-icon

            >

          </mat-action-row>

        </mat-expansion-panel>

      </mat-accordion>

    </div>

  </div>