Primero hay que tener instalada la última versión estable de **node js.**

Para comprobar la versión de angular instalada, se utiliza el comando **node –v.**

```
C:\Users\Admin2\Desktop\DWEC\Proyecto Angular 2>node -v
v6.10.0
```

Una vez instalado node, hay que crear la carpeta del proyecto y entrar en ella **desde la línea de comandos.** Una vez dentro, hay que instalar **ángular CLI.** El comando para instalarlo es:

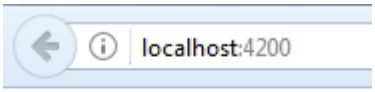
Npm install –g @angular/cli

Después, en el directorio del proyecto, hay que crear un nuevo proyecto de ángular:

ng new final-project

Con el comando **ng serve** se puede iniciar el proyecto para verlo en el navegador.

Se podrá acceder al proyecto a través de **localhost:4200**



app works!

Cualquier modificación en el proyecto aparecerá automáticamente en el navegador, la página se recarga sola.

Una vez que funcione todo, descargo el proyecto inicial de firebase.

Después hay que instalar las dependencias con el comando **npm install.**

Hay que crear un proyecto en Firebase y obtener el código de iniciación que habrá que pegar en el index.html del proyecto Angular.

```
<script src="https://www.gstatic.com/firebasejs/3.6.10/firebase.js"></script>
<script>
    // Initialize Firebase
    var config = {
        apiKey: "AIzaSyCvV5UxFB6jijPRUBcM6tlZiRVT5pPj4xM",
        authDomain: "proyecto-angular2-moises.firebaseapp.com",
        databaseURL: "https://proyecto-angular2-moises.firebaseio.com",
        storageBucket: "proyecto-angular2-moises.appspot.com",
        messagingSenderId: "502423078405"
    };
    firebase.initializeApp(config);
```

Después hay que instalar firebase:

npm install firebase –save

De esta forma se guardan los valores de la base de datos en **snap.** A través de consola se puede ver el contenido con **.val()**

```
var root = firebase.database().ref();

root.on('value', function(snap) {

    console.log(snap.val());

});
```

Después se creerán los métodos para hacer consultas la base de datos con listas y objetos.

Primero habrá que importar:

```
import {firebaseConfig} from '../environments/firebase.config';
import {AngularFire, FirebaseListObservable, FirebaseObjectObservable} from "angularfire2";
import 'rxjs/add/operator/map';
```

Crear las variables de listas y objetos además de una que puede ser de cualquier tipo, donde se guardará la primera fila del listado de maestros.

```
maestro$:FirebaseListObservable<any>;
detalle$:FirebaseObjectObservable<any>;

primerMestro:any;
```

En el constructor se le da valores a las variables que serán los resultados de las consulta de listado a la base de datos.

```
constructor(private af: AngularFire) {

  this.maestro$ = af.database.list('maestro');

  this.maestro$.subscribe(console.log);

  this.detalle$ = af.database.object('detalle/2');

  this.detalle$.subscribe(console.log);

  this.maestro$.map(maestro => maestro[1])
      .subscribe(
          maestro => this.primerMestro = maestro
      );
}
```

Después crear los métodos con las consultas para agregar, eliminar y modificar.

```
agregar() {
  this.maestro$.push({nombre: 'Aitor', apellidos: 'Menta'});
}

borrar(){
  this.maestro$.remove(this.primerMestro);
}

modificar() {
  this.maestro$.update(this.primerMestro, {nombre: 'Alan', apellidos: 'Brito'});
}

actualizarObj() {
  this.detalle$.update({descrip:'Profesor Licenciado en Matemáticas'});
}

agregarObj() { //borra el objeto y lo vuelve a crear con lo que se le pase
  this.detalle$.set({descrip:'Profesor Licenciado en Matemáticas 2'});
}
```

Una vez hecho, hay que crear un nuevo componente, el **home.**

ng generate component home

En el fichero home.component.html se creará el html con los estilos

```
<h2>Todos los profesores</h2>

<h4>Profesores Totales: TODO</h4>

<input class="search-bar" placeholder="Buscar">

<div class="lessons-list-container v-h-center-block-parent">

    <table class="table lessons-list card card-strong">
        <tbody>
        <tr>
            <td class="lessons-title"> TODO</td>
            <td class="duration">
                <i class="md-icon duration-icon">acces_tiem</i>
            </td>
        </tr>
        </tbody>
    </table>

</div>
```

Hay que crear un nuevo servicio con el comando ng g service shared/model/**nombreDelServicio**

Después en esa misma carpeta hay que crear una clase, que será una de las tablas de la base de datos con sus campos.

```
export class Detalle{
    constructor(
        public $key:string,
        public descripcion:string,
        public id_maestro:string){

    }
}
```

En detalles crear la función que obtendrá los datos de la base de datos

```
findAllDetalles(): Observable<Detalle[]>{
  return this.af.database.list('detalle');
}
```

En el componente home llamar la función para guardar los datos en detalles.

```
detalles : Detalle[];

ngOnInit() {
    this.detallesService.findAllDetalles()
        .do(console.log)
        .subscribe(
          detalles => this.detalles = detalles
        );
}
```

En el componente home **html** crear la estructura , un bucle para recorrer todos los datos y mostrarlos.

```
<div class="lessons-list-container v-h-center-block-parent">

    <table class="table lessons-list card card-strong">
        <tbody>
        <tr *ngFor="let detalle of detalles">
            <td class="lessons-title"> {{detalle.descrip}}</td>
            <td class="duration">
                <i class="md-icon duration-icon">access_time</i>
                <span>{{detalle.horas}}</span>
            </td>
        </tr>
        </tbody>
    </table>

</div>
```

Otra forma de hacerlo es crear un componente para las lecciones.

Ng g component lista-profesores

```
import {Component, OnInit, Input} from '@angular/core';
import { Detalle } from  '../shared/model/detalle';

@Component({
  selector: 'lista-profesores',
  templateUrl: './lista-profesores.component.html',
  styleUrls: ['./lista-profesores.component.css']
})
export class ListaProfesoresComponent implements OnInit {

  @Input()
  detalles : Detalle[];

  constructor() { }

  ngOnInit() {
  }

}
```

La tabla con el bucle pasaría al fichero html del nuevo componente lista profesores.

```
<table class="table lessons-list card card-strong">
  <tbody>
  <tr *ngFor="let detalle of detalles">
    <td class="lessons-title"> {{detalle.descrip}}</td>
    <td class="duration">
      <i class="md-icon duration-icon">access_time</i>
      <span>{{detalle.horas}}</span>
    </td>
  </tr>
  </tbody>
</table>
```

El antiguo **home component** queda así:

```
<div class="lessons-list-container v-h-center-block-parent">

    <lista-profesores [detalles]="detalles"></lista-profesores>

</div>
```

Lo siguiente será implementar el buscador.

Hay que poner el evento y función al input además de pasar como parámetro lo que se escribe

```
<input class="search-bar" placeholder="Buscar" (keyup)="buscar(input.value)" #input>
```

Renombrar las variables, un para detalles y otra para el filtrado.

```
    todosDetalles : Detalle[];
    filtrado : Detalle[];

  constructor(private detallesService:  DetallesService) {

  }

  ngOnInit() {
      this.detallesService.findAllDetalles()
          .do(console.log)
          .subscribe(
              detalles => this.todosDetalles = this.filtrado = detalles
          );
  }
```

Después hay que crear la función que hará el filtrado.

```
buscar(buscar:string){
    this.filtrado = this.todosDetalles.filter(detalle => detalle.descrip.includes(buscar));
}
```

Lo siguiente es configurar el router. Así al poner /home, podrá cargar la página home, o al no poner nada haga una redirección.

Hay que crear el fichero **router.config.ts** en la carpeta app y escribir las reglas re ruteo.

```
import {Route} from "@angular/router";
import {HomeComponent} from "./home/home.component";

export const routerConfig : Route[] = [
    {
        path:'home',
        component: HomeComponent
    },

    {
        path:'',
        redirectTo:'home',
        pathMatch: 'full'
    },

    {
        path:'**',
        redirectTo: 'home'
    }
];
```

En el **home.component.html** hay que usar la etiqueta **<router-outlet>**

```
<div class="list centrar">
    <router-outlet></router-outlet>
</div>
```

Ahora hay que hacer el menú de navegación, para ello hay que crear un nuevo componente, **top-menu.**

Crear el html de un menú

```
<header class="l-header v-center-parent">
    <img class="v-center" routerLink="home" src="http://icon-icons.com/icons2/11/PNG/256/customer_person_people_man_user_client_1629.png">

    <ul class="top-menu disable-link-styles">
      <li>
        <a>Home</a>
      </li>
      <li>
        <a>Profesores</a>
      </li>
    </ul>
</header>
```

Las reglas **css**

```
.top-menu li:first-child {
    margin-left: 40px;
}

header {
    height: 55px;
}
```

En el fichero ts quitar el **app-** del selector.

```
@Component({
  selector: 'top-menu',
  templateUrl: './top-menu.component.html',
  styleUrls: ['./top-menu.component.css']
})
```

Agregar el selector al html del componente **app**

```
<top-menu></top-menu>
<main class="l-main l-sample-app">
    <div class="main-container">
        <div class="list centrar">
            <router-outlet></router-outlet>
        </div>
    </div>
</main>
```

Lo siguiente es crear un nuevo componente para **profesores.** Ahora solo para tener una página mas y probar la navegación.

En el **router config** agregar la nueva regla

```
{
    path:'profesores',
    component: ProfesoresComponent
},
```

Y en el componente **top-menu** crear los enlaces, además de dar estilos al botón activo

```
<ul class="top-menu disable-link-styles">
  <li>
    <a routerLink="home" routerLinkActive="menu-active" [routerLinkActiveOptions]="{exact:true}">Home</a>
  </li>
  <li>
    <a routerLink="profesores" routerLinkActive="menu-active">Profesores</a>
  </li>
</ul>
</header>
```

En uno de los componentes, el que muestra la lista de maestros, hay que crear la función para buscar. Esa función se llama después con un evento en el campo de búsqueda.

```
maestroBtn$: Observable<MaestroBtn[]>;

constructor(private maestrosBtnService: MaestrosBtnService) { }

ngOnInit() {
  this.maestrosBtnService.findAllMaestrosBtn()
      .do(console.log)
      .subscribe(
          maestros => this.todosLosMaestros = this.filtrado = maestros
      );
  this.maestroBtn$ = this.maestrosBtnService.findAllMaestrosBtn();
}

buscar(buscar:string){
  this.filtrado = this.todosLosMaestros.filter(maestro => maestro.nombre.includes(buscar));
}
```

Fichero html del componente.

```
<h2>Lista de profesores</h2>
<input class="search-bar" placeholder="Buscar"(keyup)="buscar(input.value)" #input>

<lista-maestros [maestros]="filtrado"></lista-maestros>
```

Lo siguiente es hacer los detalles para cada profesor. Para ello, hay que obtener la url del maestro que es lo que se usa como clave primaria y foránea.

En el servicio maestrosBtService se crea una función que hará un JOIN de las dos tablas a partir de la url que se pasa por parámetro.

```
constructor(
    private route: ActivatedRoute,
    private maestrosBtService:MaestrosBtnService) { }

ngOnInit() {
     const maestroUrl = this.route.snapshot.params['id'];
    this.maestrosBtService.findDetallesForMaestro(maestroUrl)
}
```

Servicio de maestrosBt

```
findDetallesForMaestro(maestroUrl:string){
    return this.af.database.list('detalle', {
        query: {
            orderByChild:'url',
            equalTo:'maestroUrl'
        }
    });
}
```

Ahora hay que crear los formularios para añadir Profesores y Detalles.

Un componente solo para el formulario y sus validaciones

```
<form [formGroup]="form" autocomplete="off" novalidate class="lesson-form">

  <fieldset>
    <legend>Maestro</legend>
    <div class="form-field">
      <label>Nombre:</label>
      <input name="title" formControlName="nombre" placeholder="Nombre">
      <div class="field-error-message"
           *ngIf="isErrorVisible('nombre','required')">Este campo es necesario</div>
    </div>
    <div class="form-field">
      <label>Apellidos:</label>
      <input name="title" formControlName="apellidos" placeholder="Apellidos">
      <div class="field-error-message"
           *ngIf="isErrorVisible('apellidos','required')">Este campo es necesario</div>
    </div>
```

```
form:FormGroup;

constructor(private fb:FormBuilder) { }

ngOnInit() {
  this.form = this.fb.group({
    nombre: ['', Validators.required],
    apellidos: ['', Validators.required],
    dni: ['',Validators.required],
    asignatura: ['',Validators.required],
    url: ['',Validators.required]
  });

}

isErrorVisible(field:string, error:string){

  return this.form.controls[field].dirty
      && this.form.controls[field].errors &&
      this.form.controls[field].errors[error];
}
```

Y otro componente que será el que tiene el botón de enviar y las funciones para guardar los datos en la base de datos.

```
  constructor(private route:ActivatedRoute, private detallesListadoService: DetallesListadoService) { }

  ngOnInit() {
  }

  save(form){
    this.detallesListadoService.createNewDetalle(form.value)
        .subscribe(
            () => {
              alert("Se ha creado correctamente");
              form.reset();
            },
            err => alert(`error al crear el detalle ${err}`)
        );
  }

}
```