

Nombre del instructor	Adolfo Centeno Tellez		
Área temática	Full stack		
Rol			
Competencia	Angular		
Subcompetencia	-		
Objetivo Nivel Aplicación	Crear un proyecto en Angular 18, que permita administrar cada una de las secciones de nuestro CV, realizando las operaciones básicas de lectura, creación y eliminación.		
Niveles Aplicación (2.5h)			
	Duración	Objetivos	Subtemas
	2.5 hrs	Programar las operaciones básicas de lectura, creación y eliminación para dar mantenimiento a cada una de las secciones del proyecto CV, la información está contenida en la base de datos Firestore.	<ul style="list-style-type: none"> - Creación del proyecto, crear componentes y configuración de routing - Creación de las operaciones de lectura, creación y eliminación.

Título de la actividad

Proyecto administrador de la información de nuestro cv

Objetivo

Crear un proyecto en Angular 18, que permita administrar cada una de las secciones de nuestro CV, realizando las operaciones básicas de lectura, creación y eliminación

Video inicial (Teaser)- 130-160 palabras máximo

Título de la práctica: Proyecto administrador de la información de nuestro cv

En las lecciones 1 a 5 realizamos el proyecto de nuestro CV, realizamos lecturas directas de la base de datos Firestore y las visualizamos en el template estilo Elon Musk. La información permanece estática a menos que se modifique manualmente en el portal de Firebase.

En esta práctica realizaremos un nuevo proyecto en Angular llamado admin-cv, crearemos los componentes y servicios para cada sección del CV, para cada sección realizaremos las operaciones básicas de lectura, creación y borrado.

3.1. Autoevaluación

Marca con una cruz los elementos de la siguiente lista que describan cómo te percibes al final del módulo de conocimiento:

- Creación de un proyecto en Angular 18
- Instalación de las dependencias para soportar Firestore
- Puedes crear un proyecto en Firebase, además de crear una base de datos en Firestore
- Creación de los archivos de ambiente en Angular
- Descargar las credenciales de Firebase
- Crear componentes
- Crear servicios
- Inyectar servicios en componentes

Si marcaste todos los elementos de la lista ¡Felicitaciones! Estás listo para llevar a la práctica estos conocimientos a través de un caso. Si dejaste algún elemento sin marcar, te invitamos a que revises de nuevo el tema correspondiente y luego regreses a este módulo.

Involucrados

Participante (tercera persona)

Protagonista

Testigo

Participante

Impersonal

Primera persona Cuenta su propia historia	Primera persona Espectador del suceso Rol menos subjetivo Guía del participante	Tercera persona No forma parte del relato Ve el suceso "desde afuera"	Tono impersonal Influye en el lector Es objetivo Conoce todo respecto a la historia
--	--	---	--

3. Planteamiento de la situación

Se plantea la necesidad actualizar los datos de nuestro CV cada que sea necesario y así sea funcional con el paso del tiempo.

La información se encuentra almacenada en Firestore por tanto es necesario crear un proyecto nuevo en Angular que permita realizar esas actualizaciones a las secciones de nuestro CV.

Desarrollo de la situación

Para iniciar tu trabajo debes asegurarte nuevamente de tener instalado **node js** en la versión **20.x**. Usando **nvm list**, verifica se encuentre instalado correctamente, seleccione la versión correcta usando **nvm use v20.x**

Además revisar que Angular en la versión 18 esté instalado de forma correcta con el comando **ng --version**.

Acción 1: Creación del proyecto, crear componentes y configuración de routing.

Debemos crear un proyecto nuevo en la terminal.

\$ ng new admin-cv --no-standalone

```
→ 2025 ng new admin-cv --no-standalone
? Which stylesheet format would you like to use? CSS [https://developer.mozilla.org/docs/Web/CSS]
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? (y/N) N
```

Posteriormente probamos nuestro proyecto nuevo:

\$ cd admin-cv

\$ ng serve

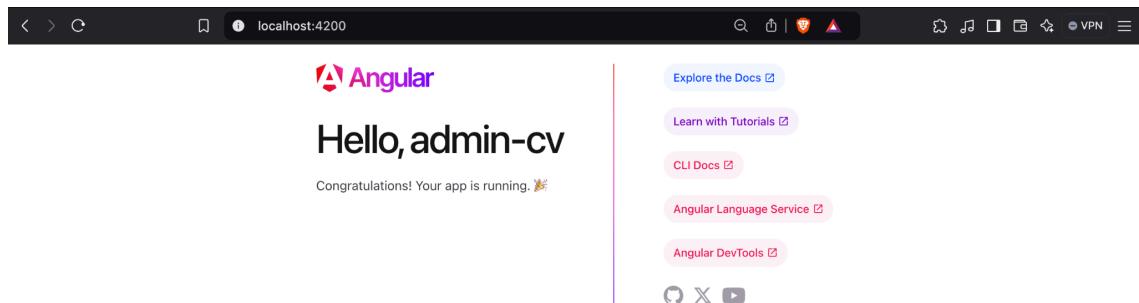
```
→ 2025 cd admin-cv
→ admin-cv git:(master) ng serve
Initial chunk files | Names           | Raw size
polyfills.js         | polyfills        | 90.20 kB
main.js              | main             | 23.85 kB
styles.css           | styles           | 95 bytes

| Initial total | 114.15 kB

Application bundle generation complete. [3.733 seconds]

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
→ Local: http://localhost:4200/
→ press h + enter to show help
```

Visualizar en el navegador en localhost:4200



El siguiente paso es actualizar el componente principal **app.component.html**.

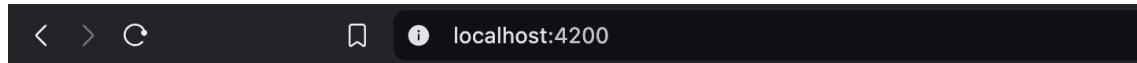
\$ nano src/app/app.component.html

```

1  <h3> <b> admin-Cv App </b> </h3>
2  <nav>
3      <ul>
4          <li><a routerLink="/header"> header </a>
5          </li>
6          <li><a routerLink="/workexperience"> work-experience </a>
7          </li>
8          <li><a routerLink="/education"> education </a>
9          </li>
10         <li><a routerLink="/certificates"> certificates </a>
11         </li>
12         <li><a routerLink="/skills"> skills </a>
13         </li>
14         <li><a routerLink="/languages"> languages </a>
15         </li>
16         <li><a routerLink="/interests"> interests </a>
17         </li>
18     </ul>
19 </nav>
20
21     <router-outlet></router-outlet>

```

Nuestro menú principal luce así:



admin-Cv App

- [header](#)
- [work-experience](#)
- [education](#)
- [certificates](#)
- [skills](#)
- [languages](#)
- [interests](#)

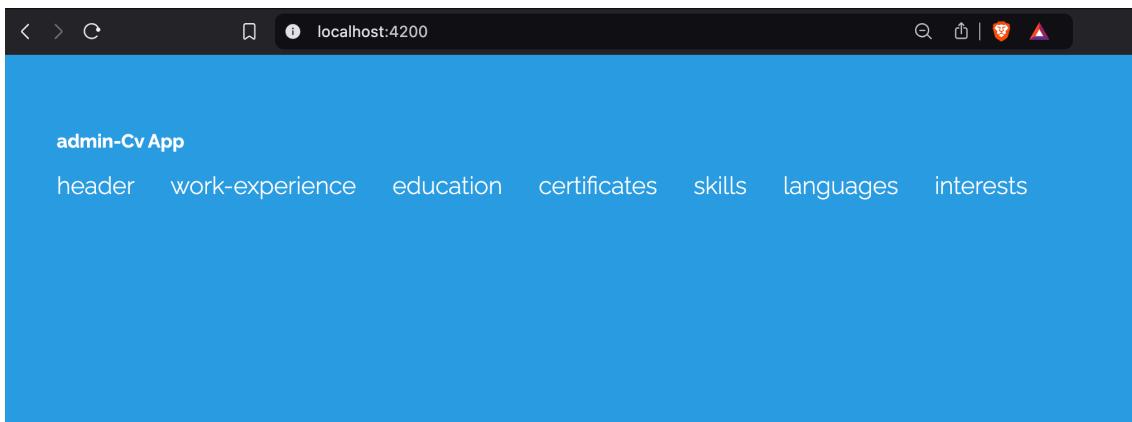
Ahora es turno de mejorar la apariencia del proyecto, agregaremos hojas de estilos globales, aplicables a todos los componentes. Actualizamos el archivo **src/styles.css**

\$ nano src/styles.css

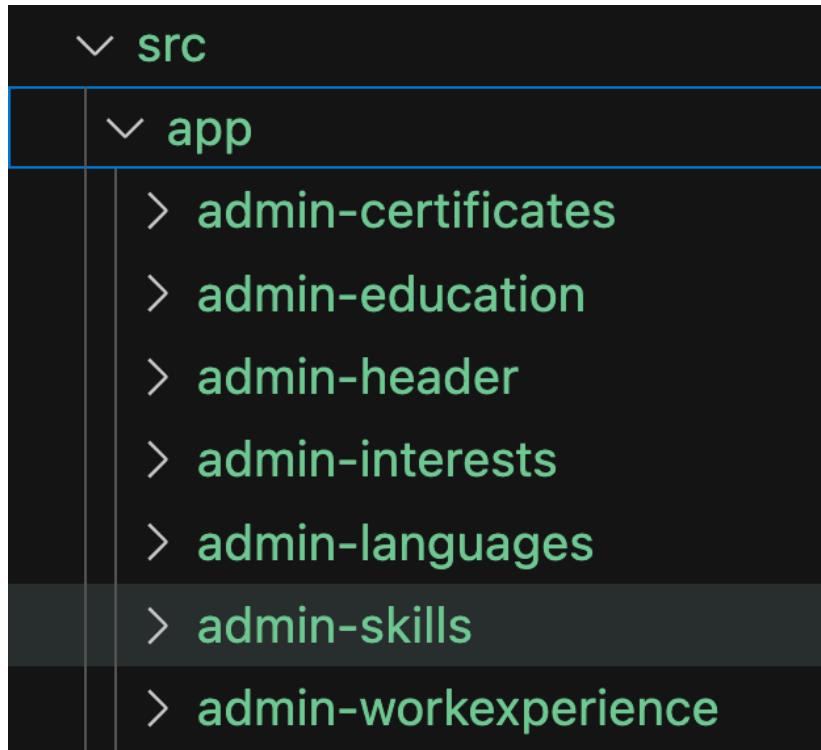
```

1  /* You can add global styles to this file, and also import other style files */
2  @import url('https://fonts.googleapis.com/css?family=Raleway:300,700');
3  body{
4      background: #2E9ce6;
5      padding: 3em;
6      font-family: 'Raleway', 'Arial';
7      color: #fff; }
8  ul {
9      list-style-type: none;
10     margin: 0 0 2em 0;
11     padding: 0; }
12    ul li {
13        display:inline;
14        margin-right: 30px; }
15    ul li a {
16        font-size: 1.5em; }
17    a{
18        color:#fff;
19        text-decoration: none; }
```

Nuestro proyecto debe verse así.



Después de crear la hoja de estilos globales, ahora crearemos los componentes necesarios para el proyecto:



\$ ng g c admin-certificates

```

→ admin-cv git:(master) ✘ ng g c admin-certificates

CREATE src/app/admin-certificates/admin-certificates.component.css (0 bytes)
CREATE src/app/admin-certificates/admin-certificates.component.html (33 bytes)
CREATE src/app/admin-certificates/admin-certificates.component.spec.ts (675 bytes)
CREATE src/app/admin-certificates/admin-certificates.component.ts (246 bytes)
UPDATE src/app/app.module.ts (521 bytes)
  
```

\$ ng g c admin-education

```

→ admin-cv git:(master) ✘ ng g c admin-education

CREATE src/app/admin-education/admin-education.component.css (0 bytes)
CREATE src/app/admin-education/admin-education.component.html (30 bytes)
CREATE src/app/admin-education/admin-education.component.spec.ts (654 bytes)
CREATE src/app/admin-education/admin-education.component.ts (234 bytes)
UPDATE src/app/app.module.ts (637 bytes)
  
```

\$ ng g c admin-header

```

→ admin-cv git:(master) ✘ ng g c admin-header

CREATE src/app/admin-header/admin-header.component.css (0 bytes)
CREATE src/app/admin-header/admin-header.component.html (27 bytes)
CREATE src/app/admin-header/admin-header.component.spec.ts (633 bytes)
CREATE src/app/admin-header/admin-header.component.ts (222 bytes)
UPDATE src/app/app.module.ts (741 bytes)
  
```

\$ ng g c admin-interests

```
→ admin-cv git:(master) ✘ ng g c admin-interests
CREATE src/app/admin-interests/admin-interests.component.css (0 bytes)
CREATE src/app/admin-interests/admin-interests.component.html (30 bytes)
CREATE src/app/admin-interests/admin-interests.component.spec.ts (654 bytes)
CREATE src/app/admin-interests/admin-interests.component.ts (234 bytes)
UPDATE src/app/app.module.ts (857 bytes)
```

\$ **ng g c admin-languages**

```
→ admin-cv git:(master) ✘ ng g c admin-languages
CREATE src/app/admin-languages/admin-languages.component.css (0 bytes)
CREATE src/app/admin-languages/admin-languages.component.html (30 bytes)
CREATE src/app/admin-languages/admin-languages.component.spec.ts (654 bytes)
CREATE src/app/admin-languages/admin-languages.component.ts (234 bytes)
UPDATE src/app/app.module.ts (973 bytes)
```

\$ **ng g c admin-skills**

```
→ admin-cv git:(master) ✘ ng g c admin-skills
CREATE src/app/admin-skills/admin-skills.component.css (0 bytes)
CREATE src/app/admin-skills/admin-skills.component.html (27 bytes)
CREATE src/app/admin-skills/admin-skills.component.spec.ts (633 bytes)
CREATE src/app/admin-skills/admin-skills.component.ts (222 bytes)
UPDATE src/app/app.module.ts (1077 bytes)
```

\$ **ng g c admin-workexperience**

```
→ admin-cv git:(master) ✘ ng g c admin-workexperience
CREATE src/app/admin-workexperience/admin-workexperience.component.css (0 bytes)
CREATE src/app/admin-workexperience/admin-workexperience.component.html (35 bytes)
CREATE src/app/admin-workexperience/admin-workexperience.component.spec.ts (689 bytes)
CREATE src/app/admin-workexperience/admin-workexperience.component.ts (254 bytes)
UPDATE src/app/app.module.ts (1213 bytes)
```

Debemos configurar el ruteo de nuestro proyecto, es decir habilitar la posibilidad de cambiar de componente a petición del usuario.

Modificamos el archivo de configuración **src/app/app-routing.module.ts**

\$ **nano src/app/app-routing.module.ts**

```

1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { AdminHeaderComponent } from './admin-header/admin-header.component';
4 import { AdminWorkexperienceComponent } from './admin-workexperience/admin-workexperience.component'
5 import { AdminEducationComponent } from module "/Users/adsoft/admin-cv/src/app/admin-certificates/admin-certificates.component"
6 import { AdminCertificatesComponent } from './admin-certificates/admin-certificates.component';
7 import { AdminSkillsComponent } from './admin-skills/admin-skills.component';
8 import { AdminLanguagesComponent } from './admin-languages/admin-languages.component';
9 import { AdminInterestsComponent } from './admin-interests/admin-interests.component';
10
11 const routes: Routes = [
12   { path: 'header', component: AdminHeaderComponent },
13   { path: 'workexperience', component: AdminWorkexperienceComponent },
14   { path: 'education', component: AdminEducationComponent },
15   { path: 'certificates', component: AdminCertificatesComponent },
16   { path: 'skills', component: AdminSkillsComponent },
17   { path: 'languages', component: AdminLanguagesComponent },
18   { path: 'interests', component: AdminInterestsComponent },
19 ];
20
21 @NgModule({
22   imports: [RouterModule.forRoot(routes)],
23   exports: [RouterModule]
24 })
25 export class AppRoutingModule {}
```

Ahora nuestro proyecto responde al Clic del usuario y permite cambiar el componente a visualizar:

Recursos:

- **Tema 1**
- **Tema 2**

Resultado esperado:

1.- Cuál es el archivo de configuración para administrar las rutas de un proyecto angular.

Opción	Retroalimentación
src/app-routing.module.ts	La opción correcta es: src/app/app-routing.module.ts
src/app/app-routing.module.ts	La opción correcta es: src/app/app-routing.module.ts
src/app/app.module.ts	La opción correcta es: src/app/app-routing.module.ts
src/app/routing.module.ts	La opción correcta es: src/app/app-routing.module.ts

2.- Archivo de hojas de estilo aplicables a todos los componentes del proyecto

Opción	Retroalimentación
src/app/styles.css	El archivo correcto es: src/styles.css
src/app/global.css	El archivo correcto es: src/styles.css
src/global.css	El archivo correcto es: src/styles.css
src/styles.css	El archivo correcto es: src/styles.css

Acción 2: - Creación de las operaciones de lectura, creación y eliminación

En esta sección lograremos la conectividad con Firestore para realizar las operaciones de lectura, escritura y eliminación de colecciones.

Empezaremos instalando los componentes para soporte Firestore en nuestro proyecto

\$ npm install --save @angular/fire@18.0.1

```
→ admin-cv git:(master) ✘ npm install --save @angular/fire@18.0.1
added 118 packages, removed 12 packages, changed 2 packages, and audited 1066 packages in 3m
159 packages are looking for funding
  run `npm fund` for details

10 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
→ admin-cv git:(master) ✘
```

Posteriormente creamos los ambientes de desarrollo, donde colocaremos los keys de Firestore

\$ ng generate environments

```
→ admin-cv git:(master) ✘ ng generate environments
CREATE src/environments/environment.ts (31 bytes)
CREATE src/environments/environment.development.ts (31 bytes)
UPDATE angular.json (3076 bytes)
```

Modificar **environment.ts** y **environment.development.ts** con las llaves del proyecto firebase, el mismo del CV de los temas 1 a 5, ya que trabajaremos con la misma base de datos:

\$ nano src/environments/environment.ts

```

1 export const environment = {
2   production: false,
3   firebaseConfig: {
4     apiKey: "AIzaSyDVb_5lgLQJP_30MAGJ-FL4BBCSqkLV50c",
5     authDomain: "my-cv-e292f.firebaseio.com",
6     projectId: "my-cv-e292f",
7     storageBucket: "my-cv-e292f.firebaseiostorage.app",
8     messagingSenderId: "18704728562",
9     appId: "1:18704728562:web:a61c4eba2ffc95a2deeb6b"
10   }
11 };

```

\$ nano src/environments/environment.development.ts

```

1 export const environment = {
2   production: false,
3   firebaseConfig: {
4     apiKey: "AIzaSyDVb_5lgLQJP_30MAGJ-FL4BBCSqkLV50c",
5     authDomain: "my-cv-e292f.firebaseio.com",
6     projectId: "my-cv-e292f",
7     storageBucket: "my-cv-e292f.firebaseiostorage.app",
8     messagingSenderId: "18704728562",
9     appId: "1:18704728562:web:a61c4eba2ffc95a2deeb6b"
10   }
11 };

```

Configurar el archivo **src/app/app.module.ts** para soportar firebase en nuestro proyecto, agregar en imports:

```

import { FormsModule } from '@angular/forms';
import { AngularFireModule } from '@angular/fire/compat';
import { environment } from '../environments/environment';

```

En la sección de imports agregar **FormsModule** para habilitar al proyecto la captura de datos en formularios HTML de entrada y el binding con variables en TypeScript.

Análogamente importar **AngularFireModule** para soportar Firestore.

```

imports: [
  BrowserModule,
  AppRoutingModule,
  FormsModule,

```

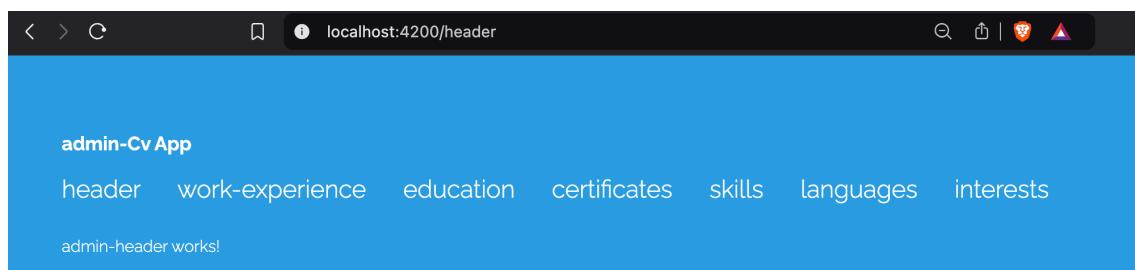
```
    AngularFireModule.initializeApp(environment.firebaseioConfig),
],
```

\$ nano src/app/app.module.ts

```
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { FormsModule } from '@angular/forms';
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { AdminHeaderComponent } from './admin-header/admin-header.component';
7 import { AdminWorkexperienceComponent } from './admin-workexperience/admin-workexperience.component';
8 import { AdminEducationComponent } from './admin-education/admin-education.component';
9 import { AdminCertificatesComponent } from './admin-certificates/admin-certificates.component';
10 import { AdminSkillsComponent } from './admin-skills/admin-skills.component';
11 import { AdminLanguagesComponent } from './admin-languages/admin-languages.component';
12 import { AdminInterestsComponent } from './admin-interests/admin-interests.component';
13 import { AngularFireModule } from '@angular/fire/compat';
14 import { environment } from '../environments/environment';
15
16 @NgModule({
17   declarations: [
18     AppComponent,
19     AdminHeaderComponent,
20     AdminWorkexperienceComponent,
21     AdminEducationComponent,
22     AdminCertificatesComponent,
23     AdminSkillsComponent,
24     AdminLanguagesComponent,
25     AdminInterestsComponent
26   ],
27   imports: [
28     BrowserModule,
29     FormsModule,
30     AppRoutingModule,
31     AngularFireModule.initializeApp(environment.firebaseioConfig),
32   ],
33   providers: [],
34   bootstrap: [AppComponent]
35 })
36 export class AppModule { }
```

Asegurarse que el proyecto siga funcionando:

\$ ng serve



En este momento, creamos una clase **model** para almacenar los documentos de la colección work-experience de Firestore.

Nos movemos a la carpeta **src/app**, creamos la carpeta **models**, dentro de models creamos la carpeta **work-experience**. Finalmente dentro de la carpeta work-experience creamos el modelo **work-experience.model.ts**.

```
$ cd src/app
```

```
$ mkdir models
```

```
$ cd models
```

```
$ mkdir work-experience
```

```
● → admin-cv git:(master) ✘ cd src/app
● → app git:(master) ✘ mkdir models
● → app git:(master) ✘ cd models
● → models git:(master) ✘ mkdir work-experience
● → models git:(master) ✘ nano work-experience/work-experience.model.ts
```

```
$ nano work-experience/work-experience.model.ts
```

```
1 export class WorkExperience {
2     id?: string;
3     startDate?: string;
4     endDate?: string = '';
5     location?: string = '';
6     position?: string = '';
7     company?: string = '';
8     accomplishments?: string = '';
9 }
10
```

Ahora, creamos el **service** para work-experience. Nos movemos a la carpeta **src/app**, creamos la carpeta services, dentro de services creamos la carpeta work-experience-service.

```
$ cd src/app
```

```
$ mkdir services
```

```
$ cd services
```

\$ mkdir work-experience-service

```
● → admin-cv git:(master) ✘ cd src/app
● → app git:(master) ✘ mkdir services
● → app git:(master) ✘ cd services
● → services git:(master) ✘ mkdir work-experience-service
```

Creamos el servicio **work-experience-service**.

\$ ng generate service work-experience-service/work-experience

```
● → services git:(master) ✘ ng generate service work-experience-service/work-experience
CREATE src/app/services/work-experience-service/work-experience.service.spec.ts (398 bytes)
CREATE src/app/services/work-experience-service/work-experience.service.ts (143 bytes)
● → services git:(master) ✘ ls work-experience-service
work-experience.service.spec.ts work-experience.service.ts
```

Insertamos el código para realizar las operaciones de lectura, creación y eliminación de un documento de la colección work-experience de Firestore.

\$ nano work-experience-service/work-experience.service.ts

```
1 import { Injectable } from '@angular/core';
2 import { AngularFirestore, AngularFirestoreCollection } from '@angular/fire/compat/firestore';
3 import { WorkExperience } from '../../../../../models/work-experience/work-experience.model';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8
9 export class WorkExperienceService {
10
11   private dbPath = '/work-experience';
12   workExperienceRef: AngularFirestoreCollection<WorkExperience>;
13
14   constructor(private db: AngularFirestore) {
15     this.workExperienceRef = db.collection(this.dbPath);
16   }
17
18   getWorkExperience(): AngularFirestoreCollection<WorkExperience> {
19     return this.workExperienceRef;
20   }
21
22   createWorkExperience(myJob: WorkExperience): any {
23     return this.workExperienceRef.add({ ...myJob });
24   }
25
26   deleteWorkExperience(id? : string): Promise<void> {
27     return this.workExperienceRef.doc(id).delete();
28   }
29 }
```

Modificamos el **typescript** del **componente admin-workexperience** para injectar el service. Agregamos los **imports** del servicio, el modelo y **map**, además actualizamos el constructor para injectar el servicio y leer los documentos de la colección work-experience.

Nos aseguramos de que estemos en **src/app**.

\$ nano admin-workexperience/admin-workexperience.component.ts

```

1 import { Component } from '@angular/core';
2 import { WorkExperienceService } from '../services/work-experience-service/work-experience.service';
3 import { WorkExperience } from '../models/work-experience/work-experience.model';
4 import { map } from 'rxjs/operators';
5
6 @Component({
7   selector: 'app-admin-workexperience',
8   templateUrl: './admin-workexperience.component.html',
9   styleUrls: ['./admin-workexperience.component.css']
10 })
11
12 export class AdminWorkexperienceComponent {
13   itemCount: number = 0;
14   btntxt: string ="Aregar";
15   goalText: string ="";
16   workExperience: WorkExperience[] = [];
17   myWorkExperience: WorkExperience = new WorkExperience();
18
19   constructor(public workExperienceService: WorkExperienceService)
20   {
21     console.log(this.workExperienceService);
22     this.workExperienceService.getWorkExperience().snapshotChanges().pipe(
23       map(changes =>
24         changes.map(c =>
25           ({ id: c.payload.doc.id, ...c.payload.doc.data() })
26         )
27       )
28     ).subscribe(data => {
29       this.workExperience = data;
30       console.log(this.workExperience);
31     });
32   }
33 }
```

Agregamos las operaciones de agregar y eliminar un documento.

```

34   AgregarJob(){
35     console.log(this.myWorkExperience);
36     this.workExperienceService.createWorkExperience(this.myWorkExperience).then(() => {
37       console.log('Created new item successfully!');
38     });
39   }
40
41   deleteJob(id? :string){
42     this.workExperienceService.deleteWorkExperience(id).then(() => {
43       console.log('delete item successfully!');
44     });
45     console.log(id);
46   }
47 }
```

También modificamos la interfaz en **html** para desplegar los documentos, y para permitir la captura y eliminación de nuevo registro.

NOTA: En cada etiqueta input de tipo texto para capturar valores, se asocia a una variable definida en el archivo Typescript usando la directiva [(ngModel)]="variable"

\$ nano admin-workexperience/admin-workexperience.component.html

```

1 <div class ="container color-dark">
2   <div class = "col">
3     <p>work-experience items</p>
4   </div>
5   </div>
6   <div class ="container color-ligh">
7     <div class = "col">
8       <form>
9         <input type="text" class="txt" name="startDate" placeholder="start date (mmm/yyyy)" [(ngModel)]= "myWorkExperience.startDate" > <br>
10        <input type="text" class="txt" name="endDate" placeholder="end date (mm/yyyy)" [(ngModel)]= "myWorkExperience.endDate" ><br>
11        <input type="text" class="txt" name="location" placeholder="location (city, country)" [(ngModel)]= "myWorkExperience.location" ><br>
12        <input type="text" class="txt" name="position" placeholder="position" [(ngModel)]= "myWorkExperience.position" ><br>
13        <input type="text" class="txt" name="company" placeholder="company" [(ngModel)]= "myWorkExperience.company" ><br>
14        <input type="text" class="txt" name="accomplishments" placeholder="accomplishments" [(ngModel)]= "myWorkExperience.accomplishments" ><br>
15        <input type="submit" class ="btn" value="{{ btntxt }}" (click)="AgregaJob()">
16      </form>
17    </div>
18    <div class = "col">
19      <p class="life-container" *ngFor="let job of workExperience;">
20        <input type="submit" class ="btn" value="delete" (click)='deleteJob(job.id)'>
21        {{job.company}} - {{job.position}}
22      </p>
23    </div>
24  </div>
25 </div>
26 </div>
27 </div>
28 </div>
29 </div>
30 </div>
31 </div>
32 </div>
33 </div>
34 </div>
35 </div>
```

Podemos visualizar nuestro componente terminado.

Probamos agregar registros nuevos

Deben actualizarse la lista de registros con el nuevo documento capturado

admin-Cv App

header work-experience education certificates skills languages interests

work-experience items

ene-2023
dic-2024
Orizaba, Mexico
full stack developer
adsoft Corp
cloud Erp, blockchain
Agregar

[delete](#) | kubeet SA de CV - Backend Developer
[delete](#) | adsoft Corp - full stack developer
[delete](#) | Waves Lab - Frontend Developer

También probemos eliminar el primer documento.

admin-Cv App

header work-experience education certificates skills languages interests

work-experience items

dic-2024
Orizaba, Mexico
full stack developer
adsoft Corp
cloud Erp, blockchain
Agregar

[delete](#) | adsoft Corp - full stack developer
[delete](#) | Waves Lab - Frontend Developer

NOTA: Visualizar su proyecto en producción del cv de los temas 1 a 5, desplegado en render.com, los cambios deben verse reflejados de forma automática.

El reto de la práctica es implementar el resto de las secciones del CV, siguiendo los procedimientos presentados en este documento.

Recursos:

- **Tema 3**
- **Tema 4**
- **Tema 5**

Resultado esperado:

1.- Módulo que debe importarse para habilitar el binding de HTML con variables en TypeScript.

Opción	Retroalimentación
BrowserModule	El módulo correcto es: FormsModule
FormsModule	El módulo correcto es: FormsModule
AngularFireModule	El módulo correcto es: FormsModule
map	El módulo correcto es: FormsModule

1.- Directiva en angular para ligar una etiqueta HTML con una variable en Typescript

Opción	Retroalimentación
[(ngmodel)="variable"]	La directiva angular correcta es: [(ngModel)="variable"]
[(Model)="variable"]	La directiva angular correcta es: [(ngModel)="variable"]
[(ngModel)=variable]	La directiva angular correcta es: [(ngModel)="variable"]
[(ngModel)="variable"]	La directiva angular correcta es: [(ngModel)="variable"]

--	--

Integración de las acciones

Has resuelto de manera satisfactoria cinco acciones relevantes:

- Crear un proyecto en Angular
- Configurar un proyecto para soporte routing
- Creación de componentes y servicios
- Conectar un Proyecto Angular con Firestore
- Implementación de las operaciones de lectura, escritura y eliminacion de documentos de una colección de Firestore.

Reflexión de la simulación final

Muchos de los problemas comunes actuales en el desarrollo web con conexiones a base de datos, es el análisis de información, así como la conectividad, pero que pueden resolverse de manera eficiente siguiendo procesos bien definidos de configuracion, creacion de componentes y servicios que implementan operaciones básicas de lectura, creacion y eliminacion.

Ahora, toma un momento para reflexionar acerca del impacto de estos conocimientos en tu vida laboral. ¿Qué beneficios consideras que te otorgan los conceptos y las funciones comentadas en este desarrollo?

A continuación, realizarás un reto, como una actividad evaluable en la que tendrás que aplicar lo aprendido en este curso y posteriormente deberás entregar en plataforma la evidencia solicitada.

