

# TECNOLÓGICO NACIONAL DE MÉXICO

## Instituto Tecnológico de Tepic



---

## Tutoriales Unidad 4

Desarrollo de Aplicaciones Híbridas

---

M.C. Israel Arjona Vizcaíno

---





# Tabla de contenido

<b>Inserción de documentos en Firebase (01/11)</b>	<b>4</b>
Paso 1: Crear un proyecto en firebase	4
Paso 2: Añade una aplicación al proyecto creado	5
Paso 3: Crea una nueva base de datos	7
Paso 4: Creación de datos en Firebase	8
Paso 5: Crea una nueva aplicación (blank)	10
Paso 6: Instalar y añadir la configuración de firebase a la aplicación	10
Paso 7: Crear modelo para la recepción y envío de información a Firebase (Modelo)	11
Paso 8: Creación de un servicio para las operaciones con firebase (Controlador)	12
Paso 9: Creación de componente con formulario para captura de estudiantes (Vista)	13
Retos	15
Reto 1: Validaciones (01/11)	15
Reto 2: Trasladar a tabs (01/11)	16
Reto especial: Ponle un login (5 puntos) (03/11)	16
<b>Consulta de documentos en Firebase (03/11)</b>	<b>16</b>
Paso 1: Crear un nuevo proyecto	16
Paso 2: Configuración de firebase en el proyecto	16
Paso 3: Creación del modelo	16
Paso 4: Creación de servicio para consulta (Controlador)	17
Paso 5: Actualizar página de home (Controlador)	17
Paso 6: Construyendo la interfaz (Vista)	18
Retos	19
Reto 1: Trasladar a tabs (03/11)	19
Reto 2: Añadir 2 tabs más (03/11)	19
Reto especial: Detalle de estudiante (5 puntos) (05/11)	19
<b>Actualización de documentos en Firebase (04/11)</b>	<b>19</b>
Paso 1: Creación de método en el servicio para actualización (Controlador)	19
Paso 2: Creación de método para conexión con el Servicio (Controlador)	20
Paso 3: Actualización de evento para lanzar la activación/desactivación del estudiante (Vista)	20
<b>Consulta los detalles del estudiante de manera individual (6/11)</b>	<b>21</b>
Paso 1: Crear la página para mostrar la información del estudiante.	21
Paso 2: Agregar el evento tappable, el cual permite presionar un cierto tiempo el componente y traer la consulta desde firebase.	21
Paso 3: Importar dos clases de Angular.	21
Paso 4: Inyectar el servicio Router en el constructor.	22
Paso 5: Importar librerías en detail.page.ts para recibir los parámetros y mostrarlas en la página de detalle.	22



Paso 6: Creación de la vista detalle.	24
Retos	25
Reto 1: Cambiar card por formulario (09/11)	25
<b>Tutorial Formas de logeo con diferentes métodos de acceso (07/11)</b>	<b>25</b>
Paso 1: Habilitar los métodos de acceso de firebase	25
Paso 2: En métodos de acceso buscamos, correo electrónico/contraseña	25
Paso 3: Habilitamos método de acceso por google	26
Paso 4: Habilitar método de acceso de github	26
Paso 4: crear Nuevo proyecto	29
Paso 5: Módulo de firebase	29
Paso 6: ir al archivo app.module.ts dentro de app y añadir las siguientes librerías	30
Paso 7: Generamos los servicios de autenticación	30
paso 8: Generamos la clase	30
Paso 9: Generamos una nueva página de login y de registro	31
Paso 10: Primero hacemos el registro, nos vamos a registro/registro.ts	32
Paso 11: Ahora nos vamos a login.ts	33
Paso 12: Mandar login como primer pantalla	37
Paso 13. Crear un logout	37
Retos	38
Reto 1: Cambiar card por formulario (08/11)	38
Eliminar estudiante de manera individual (06/11)	39
Paso 1: Crear método de eliminación en el servicio	39
Paso 2: Crear método en details.page.ts	39
Paso 3: Modificar la vista	40
Retos	40
Reto 1: Convertir a Input y actualizar (09/11)	40



# Inserción de documentos en Firebase (01/11)

## Paso 1: Crear un proyecto en firebase

1. Da clic en añadir proyecto

The screenshot shows the 'Proyectos recientes' (Recent Projects) screen in the Firebase console. At the top left is a purple button labeled '+ Añadir proyecto'. Below it is a blue button labeled 'Ver proyecto de demostración'. To the right are several project cards: 'DAM20193U2EJE02' (dam20193u2eje02), 'DAM20193EJE02' (dam20193eje02), 'Graduacion' (graduacion-db), 'DAM2018' (dam2018-8d42a), and 'DAM2018U3P01-1' (dam2018u3p01-1). Each card has a small icon and a double slash symbol below it.

2. Ponle nombre al proyecto

The screenshot shows the 'Empieza por ponerle un nombre al proyecto' (Start by giving your project a name) step in the Firebase project creation wizard. The input field is labeled 'Nombre del proyecto' and contains the text 'DAMCrudFirebase'. Below the input field is a small edit icon. At the bottom is a large blue 'Continuar' (Continue) button.



### 3. Deshabilita Google Analytics

## Google Analytics para tu proyecto de Firebase

Google Analytics es una solución de analíticas gratuita e ilimitada que permite crear informes y realizar tareas de segmentación (entre otras acciones) en Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions y Cloud Functions.

Google Analytics incluye estos servicios:

- |  |  |
|--|--|
| ✗ Pruebas A/B <small>?</small>                                       | ✗ Usuarios sin fallos <small>?</small>                               |
| ✗ Segmentación de usuarios en productos de Firebase <small>?</small> | ✗ Activadores de Cloud Functions basados en eventos <small>?</small> |
| ✗ Predicción del comportamiento de los usuarios <small>?</small>     | ✗ Informes gratuitos ilimitados <small>?</small>                     |

Habilitar Google Analytics en este proyecto  
Recomendado

## Paso 2: Añade una aplicación al proyecto creado

### 1. Añade una aplicación de tipo Web





## 2. Ponle un apodo a la aplicación

1 Registrar la aplicación

Apodo de la aplicación (?)

Configura también **Firebase Hosting** en esta aplicación. [Más información](#)

Hosting también se puede configurar más tarde. Puedes empezar de forma gratuita en cualquier momento.

**Registrar aplicación**

2 Añadir SDK de Firebase

## 3. Guarda la configuración de `firebaseConfig`

2 Añadir SDK de Firebase

Antes de utilizar cualquier servicio de Firebase, copia y pega estas secuencias de comandos en la parte inferior de la etiqueta <body>:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.2.2.firebaseio-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#available-libraries -->

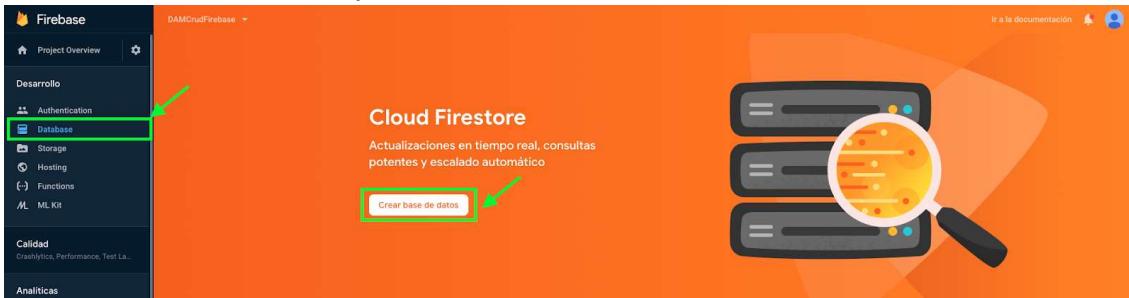
<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: [REDACTED],
    authDomain: "damcrudfirebase.firebaseio.com",
    databaseURL: "https://damcrudfirebase.firebaseio.com",
    projectId: "damcrudfirebase",
    storageBucket: "damcrudfirebase.appspot.com",
    messagingSenderId: [REDACTED],
    appId: [REDACTED]
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

Más información sobre Firebase para aplicaciones web: [Primeros pasos](#), [Referencia de APIs del SDK web](#) y [Muestras](#)

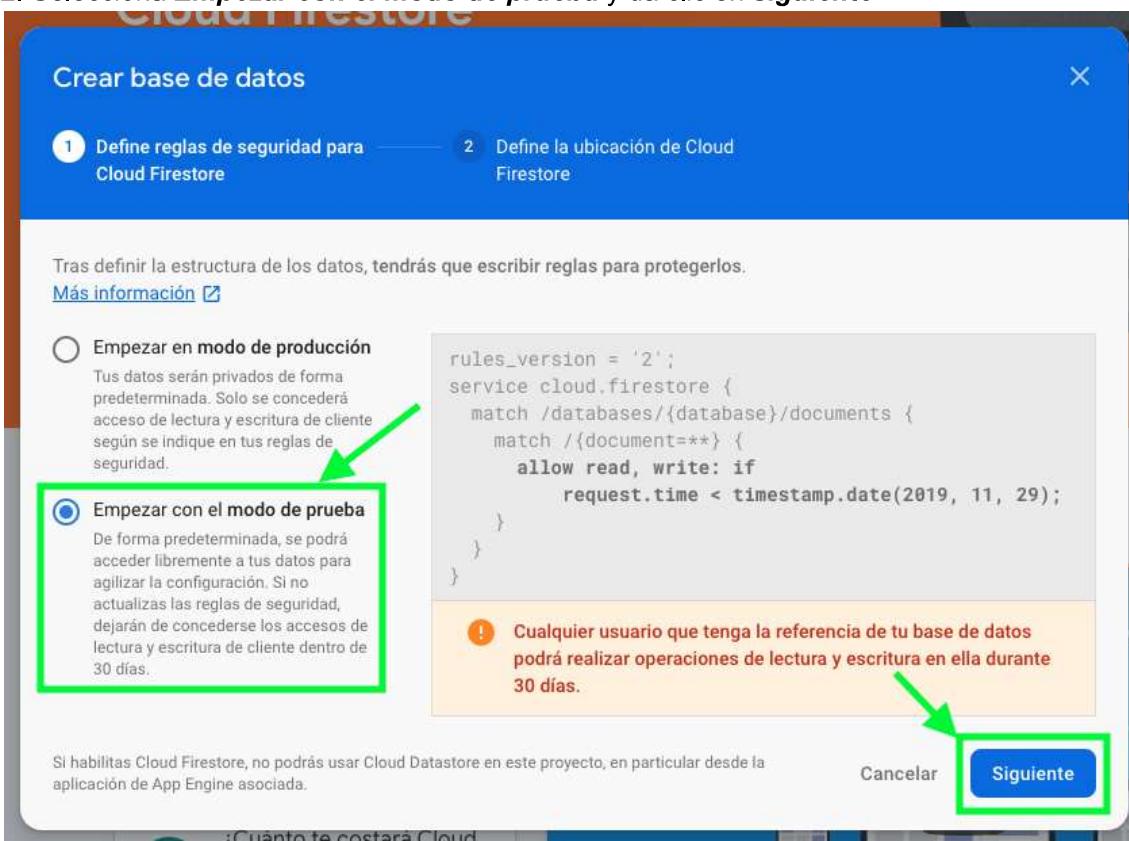


## Paso 3: Crea una nueva base de datos

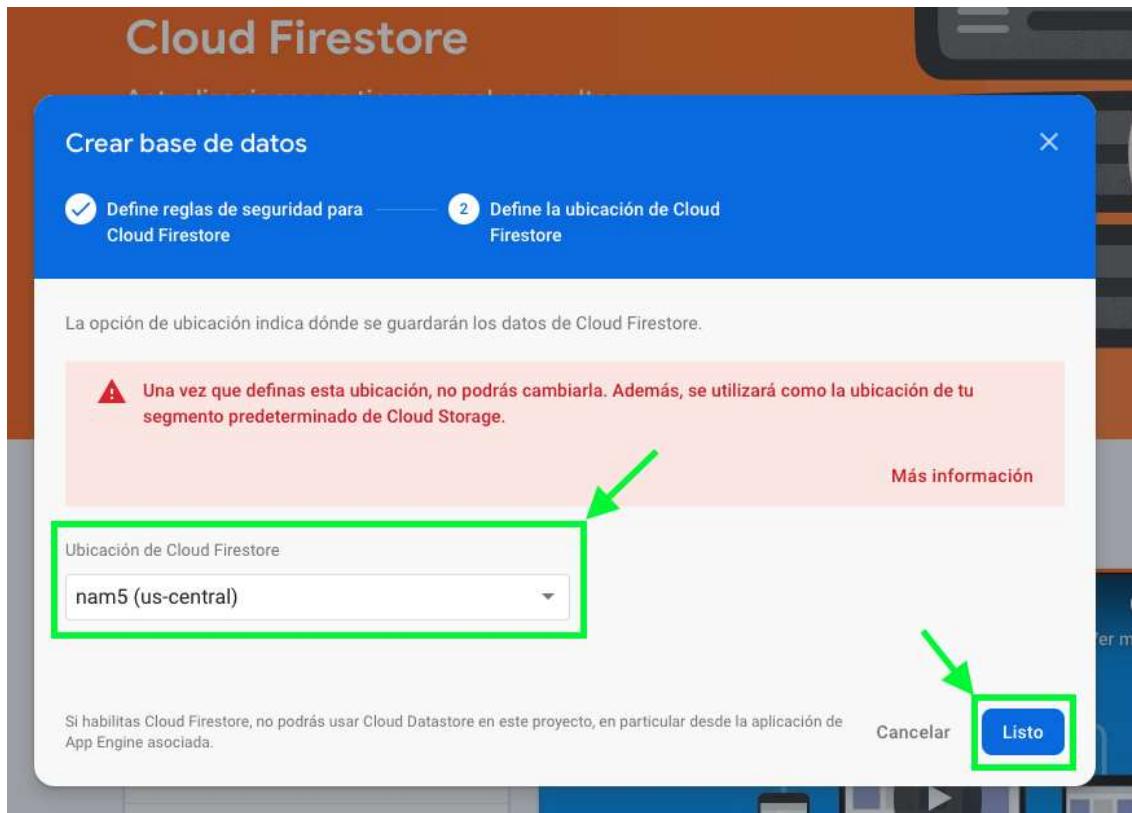
1. Ve a la sección **Database** y da clic en el botón **Crear base de datos**



2. Selecciona **Empezar con el modo de prueba** y da clic en **siguiente**

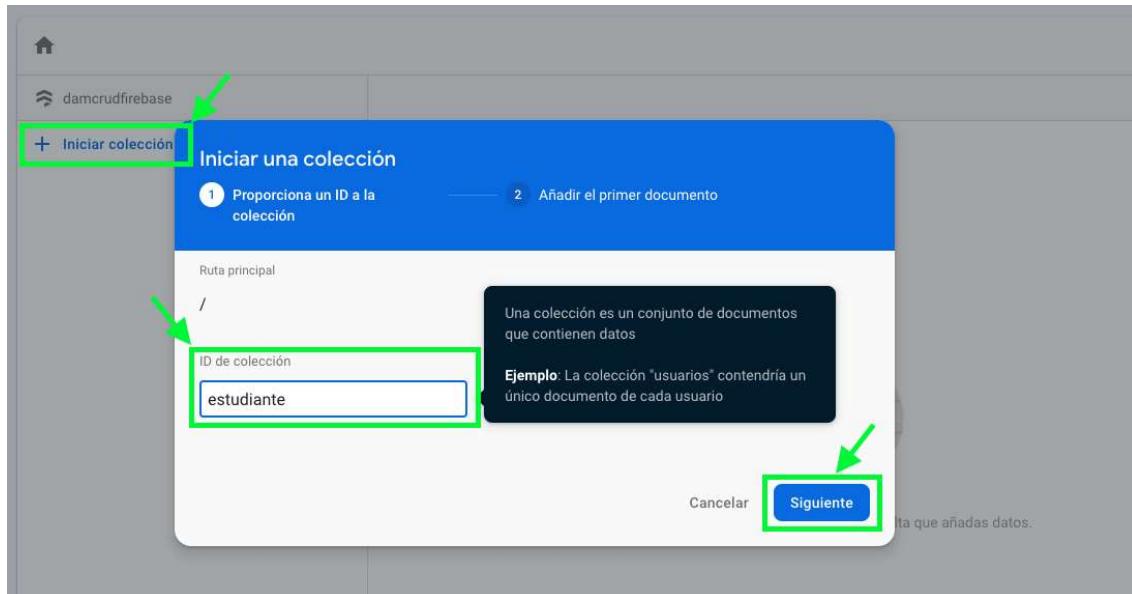


3. Elige una **ubicación** de los datos de **Cloud Firestore** (puedes elegir cualquiera), si eres extremista busca un lugar donde no haya tantos temblores, ni incendios, ni derrumbes, ni tsunamis,.....y da clic en **Listo** Yo elegiré **nam5**



## Paso 4: Creación de datos en Firebase

### 1. Crea una colección denominada **estudiante**



### 2. Crea un documento con las siguientes propiedades: name: string, controlnumber: string, curp: string, age: number, active: boolean



Iniciar una colección

1 Proporciona un ID a la colección      2 Añadir el primer documento

Ruta principal del documento  
/estudiante

ID del documento

ID automático

Campo	Tipo	Valor
name	= string	Israel Arjona Cas
controlnumber	= string	28400391
curp	= string	AOVI840917HNT
age	= number	9
active	= boolean	true

+      Cancelar      Guardar



## Paso 5: Crea una nueva aplicación (blank)

```
iMac-de-Israel:.Trash israelarjonavizcainos$ ionic start eje05
Pick a framework! 😊
Please select the JavaScript framework to use for your new app. To bypass this prompt next time, supply a value for the --type option.
? Framework: Angular
Let's pick the perfect starter template! 💖
Starter templates are ready-to-go Ionic apps that come packed with everything you need to build your app. To bypass this prompt next time, supply template, the second argument to ionic start.
? Starter template: blank
```

## Paso 6: Instalar y añadir la configuración de firebase a la aplicación

1. Estando en la carpeta del proyecto creado en el paso anterior. Instalar **firebase @angular/fire**

```
iMac-de-Israel:eje05 israelarjonavizcainos$ npm i --save firebase @angular/fire
((██████████)) " fetchMetadata: sill resolveWithNewModule firebase@7.2.2 checking instal
```

2. Abrir el archivo **environments/environment.ts** en el proyecto de ionic y añadir al objeto environment la propiedad **firebaseConfig**, generada en el paso 2 (recuerda que es un objeto de javascript, debes quitar la palabra **var** y cambiar el signo de “=” por “：“)

```
export const environment = {
  production: false,
  firebaseConfig: {
    apiKey: [REDACTED],
    authDomain: "damcrudfirebase.firebaseio.com",
    databaseURL: "https://damcrudfirebase.firebaseio.com",
    projectId: "damcrudfirebase",
    storageBucket: "damcrudfirebase.appspot.com",
    messagingSenderId: [REDACTED],
    appId: [REDACTED]
  }
};
```

3. Añade los módulos **AngularFireModule** y **AngularFireDatabaseModule**, y el archivo de configuración **src/environments/environment.ts** a **src/app/app.module.ts**



```

import { AngularFireModule } from '@angular/fire';
import { AngularFireDatabaseModule } from '@angular/fire/database';
import { environment } from '../environments/environment';

@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule,
    AngularFireModule.initializeApp(environment.firebaseioConfig),
    AngularFireDatabaseModule],
  providers: [
    StatusBar,
    SplashScreen,
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

4. Importa el componente **AngularFirestore** como servicio (**provider**) en el archivo **src/app/app.module.ts**

```

import { AngularFirestore } from "@angular/fire/firestore";

@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule,
    AngularFireModule.initializeApp(environment.firebaseioConfig),
    AngularFireDatabaseModule],
  providers: [
    StatusBar,
    SplashScreen,
    AngularFirestore,
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }
  ],
  bootstrap: [AppComponent]
})
```

## Paso 7: Crear modelo para la recepción y envío de información a Firebase (Modelo)

1. Ejecuta en la consola el comando para la creación del modelo que denominaremos **estudiante**. Esto creará una carpeta denominada **models** y dentro dos archivos typescript

```

models
  TS estudiante.spec.ts
  TS estudiante.ts
iMac-de-Israel:eje05 israelarjonavizcaino$ ionic g class models/estudiante --type=model
> ng generate class models/estudiante
CREATE src/app/models/estudiante.spec.ts (170 bytes)
CREATE src/app/models/estudiante.ts (28 bytes)
[OK] Generated class!
```



2. Abre el archivo ***src/app/models/estudiante.ts*** y actualiza la estructura, tomando en cuenta las propiedades de la colección creada en ***firebase***

```
export class Estudiante {
  name: string;
  controlnumber: string;
  curp: string;
  age: number;
  active: boolean;
}
```

## Paso 8: Creación de un servicio para las operaciones con ***firebase*** (Controlador)

Este servicio podrá ser consumido desde cualquier componente de la aplicación y servirá para hacer las operaciones ***CRUD*** que se requieren en ***firebase*** para la colección ***estudiante***.

1. Ejecute el comando para la creación del servicio

```
iMac-de-Israel:eje05 israelarjonavizcaino$ ionic g service services/estudiante
> ng generate service services/estudiante
CREATE src/app/services/estudiante.service.spec.ts (353 bytes)
CREATE src/app/services/estudiante.service.ts (139 bytes)
[OK] Generated service!
```

2. Abre el archivo ***src/app/services/estudiante.service.ts*** e importa el componente ***AngularFirestore*** y la clase ***Estudiante***

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

import { AngularFirestore } from '@angular/fire/firestore';
import { Estudiante } from 'src/app/models/estudiante';

@Injectable({
```

3. Inyecta el componente ***AngularFirestore*** en el constructor

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

import { AngularFirestore } from '@angular/fire/firestore';
import { Estudiante } from 'src/app/models/estudiante';

@Injectable({
  providedIn: 'root'
})
export class EstudianteService {

  constructor(private firestore:AngularFirestore) { }
```



4. Agrega el método `createStudent()` que se conectará con **Firebase** y añadirá un documento en la colección **estudiante**

```
createStudent(student: Estudiante) {  
  return this.firestore.collection('estudiante').add(student);  
}
```

## Paso 9: Creación de componente con formulario para captura de estudiantes (Vista)

1. Ejecuta el comando necesario para la creación de la página que denominaremos **newStudent** dentro de una carpeta denominada **pages**,

```
└─ pages  
  └─ new-student  
    TS new-student.module.ts U  
    <> new-student.page.html U  
    ⚡ new-student.page.scss U  
    TS new-student.page.spec.ts U  
    TS new-student.page.ts U  
    iMac-de-Israel:eje05 israelarjonavizcaino$ ionic g page pages/newStudent  
> ng generate page pages/newStudent  
CREATE src/app/pages/new-student/new-student.module.ts (564 bytes)  
CREATE src/app/pages/new-student/new-student.page.scss (0 bytes)  
CREATE src/app/pages/new-student/new-student.page.html (129 bytes)  
CREATE src/app/pages/new-student/new-student.page.spec.ts (720 bytes)  
CREATE src/app/pages/new-student/new-student.page.ts (275 bytes)  
UPDATE src/app/app-routing.module.ts (578 bytes)  
[OK] Generated page!  
iMac-de-Israel:eje05 israelarjonavizcaino$
```

2. Importa la clase **ReactiveFormsModule** en el archivo **src/app/pages/new-student/new-student.module.ts**

```
import { NgModule } from '@angular/core';  
import { CommonModule } from '@angular/common';  
import { FormsModule, ReactiveFormsModule } from '@angular/forms';  
import { Routes, RouterModule } from '@angular/router';  
  
import { IonicModule } from '@ionic/angular';  
  
import { NewStudentPage } from './new-student.page';  
  
const routes: Routes = [  
  {  
    path: '',  
    component: NewStudentPage  
  }  
];  
  
@NgModule({  
  imports: [  
    CommonModule,  
    FormsModule,  
    ReactiveFormsModule,  
    IonicModule,  
    RouterModule.forChild(routes)  
  ]  
}
```



3. Importa los componentes **FormGroup** y **FormBuilder** en el archivo **src/app/pages/new-student/new-student.page.ts**, los cuales son necesarios para la creación y control de formularios

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormBuilder } from '@angular/forms';
```

4. Importa la clase **Estudiante** y el componente **EstudianteService** al archivo **src/app/pages/new-student/new-student.page.ts**

```
import { Estudiante } from '../../../../../models/estudiante';
import { EstudianteService } from '../../../../../services/estudiante.service';
```

5. Declara un **atributo** que represente un objeto de tipo **Estudiante** e **inyecta el servicio EstudianteService** a través del **constructor**. Asimismo, crea un atributo para controlar el formulario denominado **myForm** e inyecta en el **constructor** un atributo derivado de **FormBuilder** que denominaremos **fb**.

```
export class NewStudentPage implements OnInit {
  public myForm:FormGroup;
  public student:Estudiante;
  constructor(private studentService:EstudianteService, private fb:FormBuilder) { }

  ngOnInit() {}
}
```

6. Inicializa el formulario **myForm** en el método **ngOnInit()**

```
ngOnInit() {
  this.myForm = this.fb.group( {
    name:[""],
    controlnumber:[""],
    curp:[""],
    age:[0],
    active:[false]
  });
}
```

7. Actualiza el archivo **src/app/pages/new-student/new-student.page.html**



```
<ion-content color="primary" padding>
  <ion-grid>
    <form [formGroup]="myForm">
      <ion-row color="primary" justify-content-center>
        <ion-col align-self-center size-md="6" size-lg="5" size-xs="12">
          <div text-center>
            <h3>Login</h3>
          </div>
          <div padding>
            <ion-item>
              <ion-input formControlName="name" type="text"></ion-input>
            </ion-item>
            <ion-item>
              <ion-input formControlName="curp" type="text"></ion-input>
            </ion-item>
            <ion-item>
              <ion-input formControlName="controlnumber" type="text"></ion-input>
            </ion-item>
            <ion-item>
              <ion-input formControlName="age" type="number"></ion-input>
            </ion-item>
            <ion-item>
              <ion-label>Activo</ion-label>
              <ion-toggle formControlName="active" color="secondary"></ion-toggle>
            </ion-item>
          </div>
          <div padding>
            <ion-button size="large" type="button" [disabled]="!myForm.valid"
              (click)="create()" expand="block">Guardar</ion-button>
          </div>
        </ion-col>
      </ion-row>
    </form>
  </ion-grid>
</ion-content>
```

8. Crea el método ***create()*** que consumirá el servicio para la creación de un documento en **Firebase**

```
create() {
  this.student = {
    name: this.myForm.controls.name.value,
    controlnumber: this.myForm.controls.controlnumber.value,
    age: this.myForm.controls.age.value,
    curp: this.myForm.controls.curp.value,
    active: this.myForm.controls.active.value,
  }

  this.studentService.createStudent(this.student);
}
```

9. Establecer la ruta que atenderá a la nueva página en ***src/app/app-routing.module.ts*** y redirecciona la ruta raíz a la ruta de ***newStudent***

```
const routes: Routes = [
  { path: '', redirectTo: 'newstudent', pathMatch: 'full' },
  { path: 'home', loadChildren: () => import('./home/home.module').then( m => m.HomePageModule)},
  { path: 'newstudent', loadChildren: () => import('./pages/new-student/new-student.module').then( m => m.NewStudentPageModule)},
];
```

## Retos

### Reto 1: Validaciones (22/04)

Pon las validaciones siguientes al formulario:



1. name: requerido, 3 caracteres mínimo y 150 máximo
2. controlnumber: requerido, 10 caracteres exactamente
3. age: requerido
4. curp: formato adecuado, requerido
5. active: requerido

**Fecha de entrega: 22 de abril de 2019**

### **Reto 2: Trasladar a tabs (23/04) - Tarea 02**

Crea una nueva aplicación de tipo **tabs** y en el primer tab establece la funcionalidad de toda la práctica anterior

**Fecha de entrega: 23 de abril de 2020**

### **Reto especial: Ponle un login (5 puntos) (24/04) - Tarea E01**

Al proyecto de tabs añade un login en donde pregunte el email y el password, los cuales deben ser iguales a algún elemento en una colección de usuarios en firebase, si existe dejalo entrar a los tabs, sino pon un mensaje de error.

**Fecha de entrega: 24 de abril de 2020**

## **Consulta de documentos en Firebase (23/04) - Tarea 03**

En esta práctica se utilizó el mismo proyecto creado en Firebase en la práctica anterior. Asimismo, se obviaron algunas cosas, pensando en que la práctica anterior fue realizada a conciencia y fueron expuestas y resueltas todas las dudas.

La práctica consistirá en la creación de una pantalla que mostrará los nombres, curps y estatus de los documentos de la colección **estudiante** de **Firebase**.

### **Paso 1: Crear un nuevo proyecto**

El proyecto será tipo blank y se denominará **eje02**

### **Paso 2: Configuración de firebase en el proyecto**

Instale **firebase @angular/fire**, configure la variable de entorno de **firebaseConfig** e importe los módulos necesarios en el archivo **src/app/app.module.ts**

### **Paso 3: Creación del modelo**

Genere una **clase** dentro de una carpeta que se llame **models** denominada **estudiante** que represente el modelo de la colección **estudiante** de **firebase**



## Paso 4: Creación de servicio para consulta (Controlador)

1. Genere un **servicio** denominado **estudiante**, dentro de una carpeta llamada **services**, para la conexión con firebase y la realización de las consultas.
2. Importe las librerías necesarias (**AngularFirestore**) y el modelo (**src/app/models/estudiante.ts**) necesarios para la conexión con firebase
3. Inyecte en el constructor un atributo que derive de **AngularFirestore**, al que llamaremos **firestore**
4. Genere un método denominado **getStudents()**, el cual realizará la consulta en la colección **estudiante** en firebase y regresará los documentos como resultado.

```
export class EstudianteService {  
  
    constructor(private firestore: AngularFirestore) { }  
  
    getStudents() {  
        return this.firestore.collection('estudiante').snapshotChanges();  
    }  
}
```

## Paso 5: Actualizar página de home (Controlador)

1. Actualiza el código de **src/app/home/home.page.ts** importando la clase del Servicio **EstudianteService**

```
import { EstudianteService } from "../services/estudiante.service";
```

2. Inyecta un atributo de tipo **EstudianteService** denominado **service**, en el constructor de **src/app/home/home.page.ts**

3. Crea un **atributo** denominado **students**, que represente un arreglo de elementos de tipo **Estudiante** y que sea de ámbito público (para que pueda ser accedido desde el html).

```
export class HomePage {  
  
    public students: Estudiante[];  
  
    constructor(private service:EstudianteService) {}  
  
}
```

4. Dentro del constructor ejecuta la consulta del servicio para obtener todos los estudiantes y almacenarlos en el atributo **students**.



```

export class HomePage {
  public students: Estudiante[];

  constructor(private service:EstudianteService) {
    this.service.getStudents().subscribe(data => {
      this.students = data.map(e => {
        return {
          id: e.payload.doc.id,
          ...e.payload.doc.data()
        } as Estudiante;
      })
    });
  }
}

```

## Paso 6: Construyendo la interfaz (Vista)

- Actualice el archivo **src/app/home/home.page.html** con el siguiente código

```

<ion-header>
  <ion-toolbar color="success">
    <ion-title>
      Estudiantes
    </ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-list>
    <ion-item-sliding *ngFor="let student of students">
      <ion-item-options side="start">
        <ion-item-option color="secondary">
          <ion-icon name="checkmark"></ion-icon>
        </ion-item-option>
      </ion-item-options>
      <ion-item>
        <ion-label>
          <h2>
            <strong>{{student.name}}</strong>
            <ion-chip color="secondary" *ngIf="student.active" class="ion-float-right">
              <ion-label>Activo</ion-label>
            </ion-chip>
          </h2>
          <h3>{{student.controlnumber}}</h3>
        </ion-label>
      </ion-item>
      <ion-item-options side="end">
        <ion-item-option color="danger">
          <ion-icon name="close"></ion-icon>
        </ion-item-option>
      </ion-item-options>
    </ion-item-sliding>
  </ion-list>
</ion-content>

```

La aplicación debería verse así





<https://github.com/iarjonavizcaino/dam20193t06.git>

## Retos

### Reto 1: Trasladar a tabs (24/04) - Tarea 04

En la aplicación de tabs creada en la práctica anterior poner en el segundo tab la funcionalidad de la página creada en esta práctica

**Fecha de entrega: 24 de abril de 2020**

### Reto 2: Añadir 2 tabs más (26/04) - Tarea 05

En la aplicación de tabs creada en la práctica anterior añada 2 tabs más:

1. El tercer tab será una lista de los estudiantes activos
2. El cuarto tab será una lista de los estudiantes inactivos

**Fecha de entrega: 26 de abril de 2020**

### Reto especial: Detalle de estudiante (5 puntos) (27/04) - E02

Añade un botón al ítem sliding en el que al presionarlo se pueda visualizar la demás información del estudiante (CURP y Edad).

**Fecha de entrega: 27 de abril de 2020**

## Actualización de documentos en Firebase (30/04)

En esta práctica aprenderás a realizar actualizaciones a documentos almacenados en Firebase.

Utilizaremos el código de la práctica anterior y serán actualizaciones mínimas que se realizarán en algunos componentes.

### Paso 1: Creación de método en el servicio para actualización (Controlador)

1. Añada un método al archivo `src/app/services/estudiante.service.ts` denominado `updateStudent()` y coloque el siguiente código

```
export class EstudianteService {  
  
    constructor(private firestore: AngularFirestore) { }  
  
    getStudents() {  
        return this.firestore.collection('estudiante').snapshotChanges();  
    }  
  
    updateStudent(student: Estudiante, id: string) {  
        this.firestore.doc('estudiante/'+id).update(student);  
    }  
}
```



## Paso 2: Creación de método para conexión con el Servicio (Controlador)

Actualiza el archivo `src/app/home/home.page.ts` añadiendo un método denominado `update()` el cual se conectará con el servicio para actualizar el documento en firebase

```
export class HomePage {
    (property) HomePage.students: Estudiante[]
    public students: Estudiante[];

    constructor(private service:EstudianteService) {
        this.service.getStudents().subscribe(data => {
            this.students = data.map(e => {
                return {
                    id: e.payload.doc.id,
                    ...e.payload.doc.data()
                } as Estudiante;
            })
        });
    }

    update(student:Estudiante, active:boolean) {
        student.active = active;
        this.service.updateStudent(student, student.id);
    }
}
```

## Paso 3: Actualización de evento para lanzar la activación/desactivación del estudiante (Vista)

Actualiza el archivo `src/app/home/home.page.html` sobreescribiendo el método clic para enlazar con el typescript

```
<ion-content>
<ion-list>
    <ion-item-sliding *ngFor="let student of students">
        <ion-item-options side="start">
            <ion-item-option color="secondary" (click)="update(student,true)">
                <ion-icon name="checkmark"></ion-icon>
            </ion-item-option>
        </ion-item-options>
        <ion-item>
            <ion-label>
                <h2>
                    <strong>{{student.name}}</strong>
                    <ion-chip color="secondary" *ngIf="student.active" class="ion-float-right">
                        <ion-label>Activo</ion-label>
                    </ion-chip>
                    <ion-chip color="danger" *ngIf="!student.active" class="ion-float-right">
                        <ion-label>Inactivo</ion-label>
                    </ion-chip>
                </h2>
                <h3>{{student.controlnumber}}</h3>
            </ion-label>
        </ion-item>
        <ion-item-options side="end" >
            <ion-item-option color="danger" (click)="update(student,false)">
                <ion-icon name="close"></ion-icon>
            </ion-item-option>
        </ion-item-options>
    </ion-item-sliding>
</ion-list>
</ion-content>
```

<https://github.com/iarjonavizcaino/dam20193t06.git> - 2do commit



## Consulta los detalles del estudiante de manera individual (6/11)

**Paso 1: Crear la página para mostrar la información del estudiante.**

```
je05 - Tutorial\u2eje06: ionic g page detail
```

**Paso 2: Agregar el evento tappable, el cual permite presionar un cierto tiempo el componente y traer la consulta desde firebase.**

1. Actualiza el archivo *home.page.html*.

```
</ion-item-option>
</ion-item-options>

<ion-item tappable (click)="detail(student)">
  <ion-label>
    <h2>
      <strong>{{student.name}}</strong>
```

## **Paso 3: Importar dos clases de Angular.**

1. La clase Router es un servicio que se utiliza para cargar otras páginas.
2. La clase NavigationExtras nos permite enviar parametros.
3. Edita el documento *home.page.ts*.

```
import {Router, NavigationExtras} from "@angular/router";

//Router Direccionar
//NavigationExtrar enviar parametros a otra pagina.
```



## Paso 4: Inyectar el servicio Router en el constructor.

1. Actualiza el archivo **home.page.ts**. Al inyectar el servicio nos permitirá desplazarnos de páginas.

```
19  public students: Estudiante[];  
20  
21  constructor(private service: EstudianteService, private router: Router) { //Inyectar router al constructor  
22    this.service.getStudents().subscribe(data => {  
23      this.students = data.map(e => {  
24        return {
```

2. Creación del método detail() se hace uso de NavigationExtras para el envío de parámetros.

```
    this.service.updateStudents(student, student.  
  }  
  
  detail(student: Estudiante){  
    let navext: NavigationExtras = {  
      queryParams:{  
        special: JSON.stringify([student])  
      }  
    };  
    this.router.navigate(['/detail'], navext);  
  }  
}
```

## Paso 5: Importar librerías en detail.page.ts para recibir los parámetros y mostrarlas en la página de detalle.

1. Actualiza el archivo **detail.page.ts**.

```
import {ActivatedRoute, Router} from "@angular/router";  
import { Estudiante } from "../models/estudiante";  
import {EstudianteService} from "../services/estudiante.service";  
import {ToastController} from "@ionic/angular";
```



2. Crear un objeto de la clase Estudiante:  
**student: Estudiante;**

```
export class DetailPage implements OnInit {  
  student: Estudiante;
```

3. Inyectar los servicios en el constructor:  
**constructor(private service: EstudianteService,  
 private actroute: ActivatedRoute,  
 private router: Router,  
 private toast: ToastController).**

```
styleUrls: ['./detail.page.scss'],  
}  
export class DetailPage implements OnInit {  
  
  student: Estudiante;  
  
  constructor(private service: EstudianteService, //Inyectar al constructor  
             private actroute: ActivatedRoute,  
             private router: Router,  
             private toast: ToastController) { //Recibir parametros.
```

4. Crear la función arrow, para recibir los parámetros con el servicio ActivatedRoute, la cual afecta el ámbito de las variables. Coloque el siguiente código.

```
styleUrls: ['./detail.page.scss'],  
}  
export class DetailPage implements OnInit {  
  
  student: Estudiante;  
  
  constructor(private service: EstudianteService, //Inyectar al constructor  
             private actroute: ActivatedRoute,  
             private router: Router,  
             private toast: ToastController) { //Recibir parametros.  
  
    this.actroute.queryParams.subscribe( //Funcion arbol, afectan el ambito de las variables  
      params => {  
        if(params && params.special){  
          this.student = JSON.parse(params.special) as Estudiante;  
          console.log(this.student);  
        }  
      }  
    );  
  }  
}
```

5. Finalmente para comprobar que los datos se traen exitosamente se ha colocado la última línea de código para revisar el JSON en consola.



## Paso 6: Creación de la vista detalle.

- Actualiza el archivo **detail.page.html**, colocando el siguiente código html.

```

2   <ion-toolbar>
3     <ion-buttons slot="start">
4       <ion-back-button defaultHref="/"></ion-back-button>
5     </ion-buttons>
6     <ion-title>{{student.name}}</ion-title>
7   </ion-toolbar>
8 </ion-header>
9 <ion-content>
10 <ion-fab horizontal="end" vertical="top" slot="fixed" edge>
11 <ion-fab-button>
12   <ion-icon name="add"></ion-icon>
13 </ion-fab-button>
14 <ion-fab-list>
15   <ion-fab-button color="success">
16     <ion-icon name="checkmark"></ion-icon>
17   </ion-fab-button>
18   <ion-fab-button color="warning">
19     <ion-icon name="close"></ion-icon>
20   </ion-fab-button>
21   <ion-fab-button color="danger">
22     <ion-icon name="trash"></ion-icon>
23   </ion-fab-button>
24 </ion-fab-list>
25 </ion-fab>
26 <ion-card text-center>
27   <ion-card-header>
28     <ion-card-subtitle>{{student.name}}</ion-card-subtitle>
29     <ion-card-title>{{student.controlnumber}}</ion-card-title>
30   </ion-card-header>
31   <ion-card-content>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

31 <ion-card-content>
32   <p>CURP:<strong>{{student.curp}}</strong></p>
33   <p>EDAD:<strong>{{student.age}}</strong></p>
34   <ion-chip color="secondary" *ngIf="student.active" class="ion-float-center">
35     <ion-label>Activo</ion-label>
36   </ion-chip>
37   <ion-chip color="danger" *ngIf="!student.active" class="ion-float-center">
38     <ion-label>Inactivo</ion-label>
39   </ion-chip>
40 </ion-card-content>
41 </ion-card>
42 </ion-content>
43
```

- Finalmente, la aplicación deberá verse así:

← Arleth Karina Acosta Torres

Arleth Karina Acosta Torres  
15400771  
CURP:AOTA970903MNTCRR09  
EDAD:22  
Inactivo

X
✓
✗
✖



## Retos

### Reto 1: Cambiar card por formulario (09/11)

Cambiar el Card con la información del estudiante a un formulario y añadir un botón para editar cualquier propiedad.

## Tutorial Formas de logeo con diferentes métodos de acceso (07/11)

### Paso 1: Habilitar los métodos de acceso de firebase

1. En firebase nos vamos a autenticación. posteriormente a métodos de acceso

The screenshot shows the Firebase console interface. On the left, there's a sidebar with 'Ionic-App' and 'Desarrolla'. The main area is titled 'DAMCrudFirebase - Authentication'. Below it, there are tabs: 'Usuarios', 'Método de acceso' (which is highlighted in blue), 'Plantillas', and 'Usos'. Under the 'Método de acceso' tab, there's a section titled 'Proveedores de acceso' with two tabs: 'Proveedor' and 'Estado'.

### Paso 2: En métodos de acceso buscamos, correo electrónico/contraseña

1. le daremos click a las opciones de google, github



The screenshot shows the Firebase console's authentication settings. Under the 'Método de acceso' tab, the 'Correo electrónico/contraeña' provider is listed with its status set to 'Habilitado'. Below the table, there is a note explaining that this provider allows users to register with an email and password, includes two-step verification, password recovery, and link account features. There is also a note about linking accounts. At the bottom right, there are 'Cancelar' and 'Guardar' buttons.

## Paso 3: Habilitamos método de acceso por google

Solo habilitamos y posteriormente introducimos nuestro correo institucional en la parte de correo electrónico el nombre del proyecto lo genera solo. cuando habilitamos y después le damos guardar.

The screenshot shows the same Firebase authentication settings page after enabling the Google provider. The 'Google' provider is now listed with its status set to 'Habilitado'. A note below explains that Google sign-in is configured and linked to the app's iOS and Android clients. It also provides instructions for linking Google sign-in to an Android app by adding the SHA-1 key to the project's configuration. A modal window is open at the bottom, prompting the user to update the project configuration with their public name and GitHub email address. The modal fields are: 'Nombre público del proyecto' (project.395421403211) and 'Correo electrónico de administración del proyecto' (eliopev@itepic.edu.mx). There is a 'Actualizar' button at the bottom of the modal.

## Paso 4: Habilitar método de acceso de github

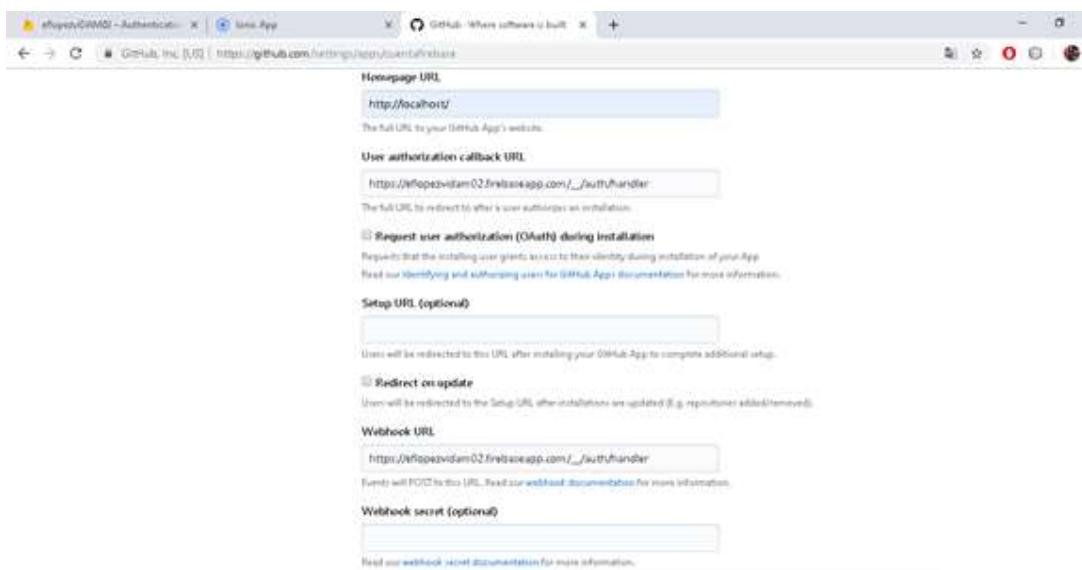
1. Método de acceso por github, primeros tendríamos que irnos a githut y crear una api para conectarnos a ella. En



2. Los campos azules pondremos el id y el secreto del cliente que nos arroja nuestra api una vez creada. Nos vamos al siguiente enlace para crear un app en git hub
3. Copiamos la url, que aparece en pantalla, la ocuparemos después



4. <https://github.com/apps/new>, y al siguiente enlace



### Los campos que interesan

**Homepage URL** = el nombre que se muestra en pantalla

**user authorization callback URL** = el enlace que copiamos anteriormente

**webhook URL** = el enlace que copiamos anteriormente



---

## 5. Desactiva los certificados de seguridad

---

### SSL verification

By default, we verify SSL certificates when delivering payloads.

**Enable SSL verification**     **Disable (not recommended)**

---

## 6. Crear la APP

Where can this GitHub App be installed?

**Only on this account**  
Only allow this GitHub App to be installed on the account's account.

**Any account**  
Allow this GitHub App to be installed by any user or organization.

[Create GitHub App](#)   [Cancel](#)

---

The screenshot shows the GitHub App settings page. At the top, there is a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, a yellow banner displays the message 'Registration successful. You must generate a private key in order to install your GitHub App.' Underneath the banner, there are tabs for 'Settings', 'Developer settings', 'GitHub App', and 'Connect Firebase'. The 'GitHub App' tab is currently selected. On the left, there is a sidebar with options: 'General' (selected), 'Permissions & events', 'Install App', 'Advanced', and 'Public page'. The main content area is titled 'About' and contains the following information:

- Owned by: @diferencia
- App ID: 45472
- Client ID: 1v1\_1Rq929a4B6ac65bb
- Client secret: b2565aa48b57e14c7467e73e5bd5e4e50870f1df1d

At the bottom of the 'About' section, there are two buttons: 'Revoke all user tokens' and 'Reset client secret'.

## 7. Nos generaría los campos que ocupamos QUE SON EL id y El secret. ahora nos vamos a firebase copiamos lo que nos arrojo nuestra app de git hub y en : el método de acceso de github en firebase y le damos guardar

---



The screenshot shows the Firebase console interface. On the left, there's a sidebar with 'Ionic App' at the top, followed by 'Desarrolla' (Authentication, Database, Storage, Hosting, Functions, ML Kit), 'Configuración' (Diagnósticos, Rendimiento, Redes), 'Estadísticas' (Extensión), and 'Spark' (Create UID o correo). The main area is titled 'Authenticación' and shows a 'Nuevo' button. A modal window is open for a new client ID, with fields for 'ID del cliente' (set to 'M.de75512e18056c2f') and 'Secreto del cliente' (set to 'd3ef8852b1ff08b0b0509d725de90e9b552cba520'). Below these fields is a note: 'Para completar la configuración, agrega esta URL de devolución de llamada de autorización a la configuración de tu app de GitHub.' with a link 'Más información'. At the bottom of the modal are 'Cancelar' and 'Guardar' buttons. The status bar at the bottom shows 'Windows Escríbe aquí para buscar' and a taskbar with icons for File Explorer, Task View, Edge, and others.

## Paso 4: crear Nuevo proyecto

1. Crear un nuevo proyecto con el comando `ionic start login` y lo guardamos en una parte específica de tu computador

```
PROBLEMA: OUTPUT DEBE SER COMPLETO TERMINAL  
Windows PowerShell  
Copyright (c) Microsoft Corporation. Todos los derechos reservados.  
Proueba la nueva tecnologia PowerShell multiplataforma: https://aka.ms/powershell  
PS C:\Users\Ronaldo> ionic start login
```

2. Escoger la plantilla tabs o blank, que es la que se siguen utilizando. Al momento de hacer login por el método tradicional, gmail, Facebook o github. Te mande a la pantalla principal de tabs o blank

## Paso 5: Módulo de firebase

1. descargar un módulo `npm i firebase @angular/fire`



## Paso 6: ir al archivo app.module.ts dentro de app y añadir las siguientes librerías

```

10 app.module.ts • 10 environments •
src > app > app.module.ts > AppModule
11 import { NgModule } from '@angular/core';
12 import { BrowserModule } from '@angular/platform-browser';
13
14 import { AppComponent } from './app.component';
15
16 import { AppRoutingModule } from './app-routing.module';
17
18 import { AngularFireModule } from '@angular/fire';
19 import { AngularFirestoreModule } from '@angular/fire/database';
20 import { AngularFirestore } from '@angular/fire/firestore';
21
22 //enlace de configuración de firebase
23 import { environment } from './environments/environment';
24
25 import { AngularAuthModule } from '@angular/fire/auth';
26
27 
```

1. Mas abajo dentro de app.module.ts, se añade angularFireAuthModule, que es la que permite crear distintos logeos con diferentes plataformas

```

10 app.module.ts • 10 environments •
src > app > app.module.ts > AppModule
11 import { NgModule } from '@angular/core';
12 import { BrowserModule } from '@angular/platform-browser';
13
14 import { AppComponent } from './app.component';
15 import { AppRoutingModule } from './app-routing.module';
16
17 import { AngularFireModule } from '@angular/fire';
18 import { AngularFirestoreModule } from '@angular/fire/database';
19 import { AngularFirestore } from '@angular/fire/firestore';
20
21 import { AngularAuthModule } from '@angular/fire/auth';
22
23 @NgModule({
24   declarations: [AppComponent],
25   entryComponents: [],
26   imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule,
27             AngularFireModule.initializeApp(environment.firebaseioConfig), // enlazar firebase se dice que config
28             AngularFirestoreModule,
29             AngularAuthModule],
30   providers: [
31     StatusBar,
32     SplashScreen,
33     { provide: RouteReuseStrategy, useClass: IonicRouteStrategy },
34     AngularFirestore
35   ],
36   bootstrap: [AppComponent]
37 })
38 export class AppModule {}
```

## Paso 7: Generamos los servicios de autenticación

```

PS C:\Users\Ronaldo\Desktop\ionic\Unidad2\loginPrue>
PS C:\Users\Ronaldo\Desktop\ionic\Unidad2\loginPrue>
PS C:\Users\Ronaldo\Desktop\ionic\Unidad2\loginPrue>
PS C:\Users\Ronaldo\Desktop\ionic\Unidad2\loginPrue>
PS C:\Users\Ronaldo\Desktop\ionic\Unidad2\loginPrue>
PS C:\Users\Ronaldo\Desktop\ionic\Unidad2\loginPrue> ionic g service services/auth
```

## paso 8: Generamos la clase

```

CREATE src/app/services/auth.service.ts (133 bytes)
[OK] Generated service!
PS C:\Users\Ronaldo\Desktop\ionic\Unidad2\loginPrue> ionic g class models/user --type=model
> ng.cmd generate class models/user
```



## Paso 9: Generamos una nueva página de login y de registro

```
CREATE src/app/models/user.spec.ts (146 bytes)
CREATE src/app/models/user.ts (22 bytes)
[OK] Generated class!
PS C:\Users\Ronaldo\Desktop\ionic\Unidad2\loginPrue> ionic generate page login
```

```
[OK] Generated page!
PS C:\Users\Ronaldo\Desktop\ionic\Unidad2\loginPrue> ionic generate page register
```

1. Añadir atributos a la clase user.ts, models/user.ts

```
src > app > models > user.ts > User
1  export class User{
2      email: string;
3      password: string;
4  }
5
```

2. Añadir métodos de logeo en services/authservices, Añadimos librerías

```
src > app > services > auth.service.ts > AuthService > onLogin
1  import { Injectable } from '@angular/core';
2  import { AngularFireAuth } from '@angular/fire/auth';
3  import {User} from '../models/user';
4
5  import { auth } from 'firebase/app';
```

3. Añadimos la constructor los siguientes parámetros y creamos el método registrar



```

  TS app.module.ts      TS auth.service.ts ●    TS users.ts      TS register.page.ts      ○ register.page.html
src > app > services > TS auth.service.ts > AuthService > onLogin
  8     providedIn: 'root'
  9   })
10   export class AuthService {
11
12     public isLoggedIn: any = false;
13     constructor(public afAuth: AngularFireAuth, private afsAuth: AngularFireAuth) {
14       afAuth.authState.subscribe(user => (this.isLoggedIn = user));
15     }
16
17     //register
18     async onRegister(user:User) {
19       try {
20         return await this.afAuth.auth.createUserWithEmailAndPassword(
21           user.email,
22           user.password
23         )
24       } catch (error) {
25         console.log('error on register', error)
26       }
27     }
28   }

```

4. Dentro del archivo services, crearemos el logeo normal, Facebook, google y github y después los mandamos a llamar

```

Debug Terminal Help          • auth.service.ts - loginPrue - Visual Studio Code
home.page.ts      ○ home.page.html      TS auth.service.ts ●
c > app > services > TS auth.service.ts > AuthService
27   }
28
29   //login
30   async onLogin(user:User) {
31     try {
32       return await this.afAuth.auth.signInWithEmailAndPassword(user.email, user.password)
33     } catch (error) {
34       console.log('error en login', error);
35     }
36   }
37
38   loginGitUser(){
39     return this.afsAuth.auth.signInWithPopup(new auth.GithubAuthProvider());
40   }
41   loginGoogleUser(){
42     return this.afsAuth.auth.signInWithPopup(new auth.GoogleAuthProvider());
43   }
44 }
45
46

```

**Paso 10: Primero hacemos el registro, nos vamos a registro/registrar.ts  
Añadimos las siguientes librerías**



```
TS login.page.ts      TS register.page.ts ×    TS auth.service.ts

src > app > register > TS register.page.ts > RegisterPage
1   import { Component, OnInit } from '@angular/core';
2   import {Router} from '@angular/router';
3   import {AuthService} from '../services/auth.service';
4   import {User} from '../models/user';
5
```

1. añadimos lo siguiente para crear el registro

```
export class RegisterPage implements OnInit {

  user: User = new User();
  constructor(private authSvc: AuthService, private router: Router) { }

  ngOnInit() {
  }

  async onRegister(){
    const user = await this.authSvc.onRegister(this.user);
    if(user){
      console.log('successfully created user');
      this.router.navigateByUrl('/home');
    }
  }
}
```

2. Ahora nos vamos a Register.html, y ponemos el siguiente código, podemos probar que ya nos registra usuarios en firebase manera normal



```
7  ion-content color="secondary">
8    <form>
9      <ion-grid>
10        <ion-row justify-content-center>
11          <ion-col align-self-center size-md="6" size-lg="5" size-xlg="4">
12            <div text-align="center">
13              <h3>Registro!</h3>
14            </div>
15            <div padding>
16              <ion-item>
17                <ion-input name="email" type="email" [(ngModel)]="user.email" placeholder="correo" required></ion-input>
18              </ion-item>
19              <ion-item>
20                <ion-input name="password" type="password" [(ngModel)]="user.password" placeholder="password" required></ion-input>
21              </ion-item>
22            </div>
23            <div padding>
24              <ion-button color="primary" size="large" [click]="_onregister()" expand="block" >Registrar</ion-button>
25            </div>
26          </div>
27          <a href="/login">Ya tienes cuenta!</a>
28        </ion-col>
29      </ion-row>
30      <ion-grid>
31        <ion-row>
32      </ion-grid>
33    </form>
34  </ion-content>
```

## Paso 11: Ahora nos vamos a login.ts

1. copiamos las siguientes librerías

```
TS login.page.ts X TS auth.service.ts
src > app > login > TS login.page.ts
1  import { Component, OnInit } from '@angular/core';
2  import { Router } from '@angular/router';
3  import { AuthService } from '../services/auth.service';
4  import { User } from '../models/user';
5  import { AlertController } from '@ionic/angular';
6
```

2. Añadimos el siguiente código en el login.ts para crear el login normal, google, github



```
register.page.ts    16 login.page.ts ●  0 login.page.html      0 register.page.html
c > app > login > 16 login.page.ts > LoginPage > @ onLogin
13  export class LoginPage implements OnInit {
14    user: User = new User();
15    constructor(private router: Router, private authSvc: AuthService, private alertController: AlertController,
16      private authService: AuthService) {}
17
18    ngOnInit() {
19    }
20    async onLogin() {
21      const user = await this.authSvc.onLogin(this.user);
22      if (user) {
23        console.log('successfully logged user');
24        this.router.navigateByUrl('/home');
25      } else {
26
27        const alert = await this.alertCtrl.create({
28          header: 'Datos incorrectos',
29          message: 'Los datos son incorrectos',
30          buttons: [
31            {
32              text: 'salir',
33            }
34          ]
35        });
36        await alert.present();
37      }
38    }
39  
```

3. y más abajo del método ponemos los siguientes métodos, los cuales se encargarán de que hagamos login con gmail y git hub



```
p > login > TS LoginPage > LoginPage > onLogin
  ↴
onLoginGoogle(): void {
  this.authService.loginGoogleUser()
    .then((res) => {
      this.onLoginRedirect();
    })
    .catch(err => console.log('err', err.message));
}

onLoginGit(): void {
  this.authService.loginGitUser()
    .then((res) => {
      this.onLoginRedirect();
    })
    .catch(err => console.log('err', err.message));
}

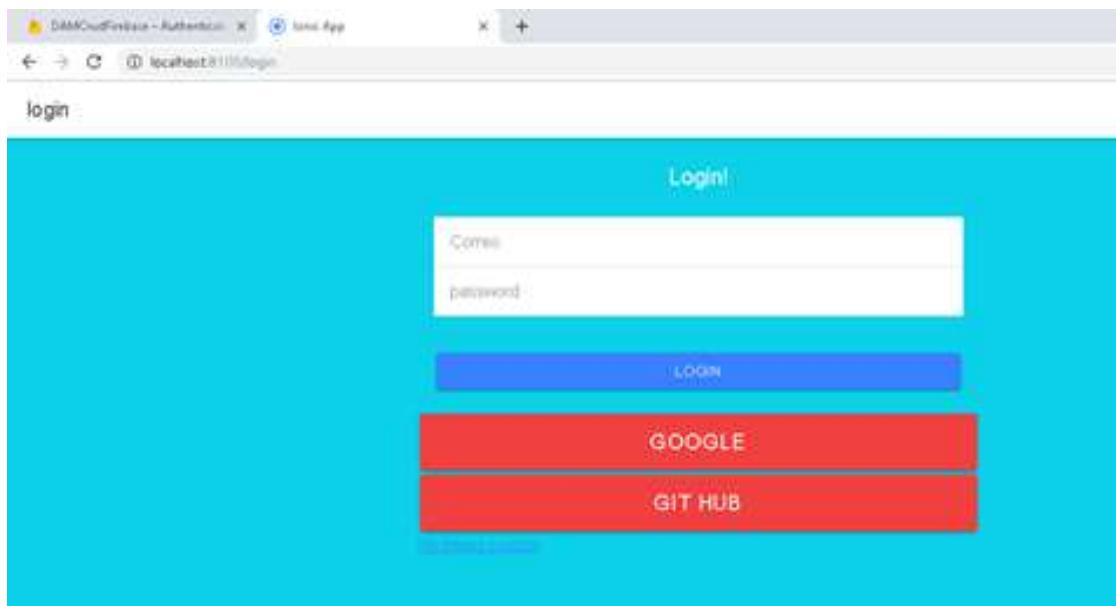
}
onLoginRedirect(): void {
  this.router.navigate(['/home']);
}
```

4. Agregamos lo siguiente en login.html. donde mandaremos a llamar los métodos



```
register.page.ts    16 login.page.ts •  0 login.page.html •  0 register.page.html
> app > login > 0 login.page.html > 0 ion-content > 0 form > 0 ion-grid > 0 ion-row > 0 ion-col > 0 div > 0 ion-button
  ion-content color="secondary">
    form
      ion-grid
        ion-row color="primary" justify-content-center>
          ion-col align-self-center size-md="6" size-lg="5" size-xs="12"
            div text-center
              h3>Login
            div padding
              ion-item
                ion-input name="email" type="email" [(ngModel)]="user.email" placeholder="Correo" required
              ion-item
              ion-item
                ion-input name="password" type="password" [(ngModel)]="user.password" placeholder="password"
              ion-item
            div padding
              ion-button (click)="onLogin()" expand="block">Login
                GOOGLE
              ion-button
              ion-button expand="block" size="large" color="danger" (click)="onLoginGit()">
                Git Hub
              ion-button
            
```

5. Hasta este punto, puedes poner ionic serve. y <http://localhost:8100/login>, ya podría hacer login normal, gmail y github



## Paso 12: Mandar login como primer pantalla

- en app.routing.module.ts, añadimos los siguientes comandos para que nuestra primer página sea el login

```
const routes: Routes = [
  { path: '', redirectTo: 'login', pathMatch: 'full' },
  { path: 'home', loadChildren: './home/home.module#HomePageModule' },
  { path: 'login', loadChildren: () => import('./login/login.module').then( m => m.LoginPageModule) },
  { path: 'register', loadChildren: './register/register.module#RegisterPageModule' },
];
```

## Paso 13. Crear un logout

- Ir finalmente en home.ts
- Añadimos lo siguiente

```
import {Router} from '@angular/router';
import {AngularFireAuth} from '@angular/fire/auth';
import {AuthService} from './services/auth.service'
```

```
})
export class HomePage {
  constructor(private authSvc: AuthService, private router: Router, private afAuth: AngularFireAuth) {}

  onLogout(){
    console.log('Logout');
    this.afAuth.auth.signOut();
    this.router.navigateByUrl('/login')
  }
}
```



3. Añadimos lo siguiente en el home. Html

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Ionic Blank
    </ion-title>
  </ion-navbar>
</ion-header>

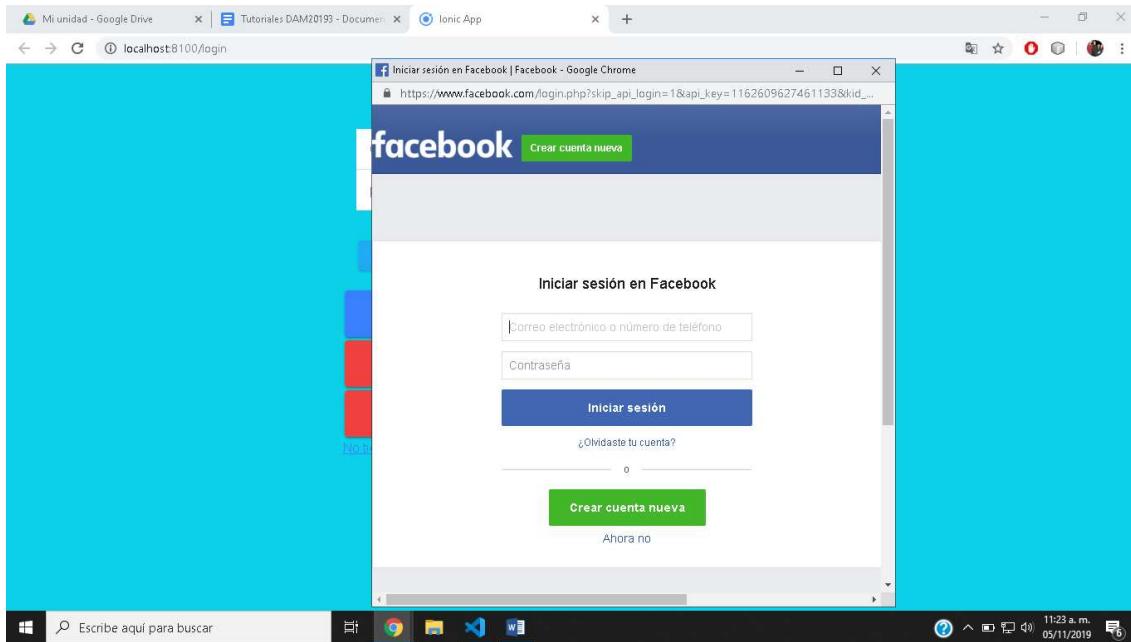
<ion-content>
  <ion-button size="large" (click)="mninguit()">> salir</ion-button>
</ion-content>
```

## Retos

### Reto 1: Login Twitter y Facebook (08/11)

Añadir ahora un login de Twitter y Facebook. Crear un botón o un ícono en login.html y que al hacer click puedas hacer session en Twitter y otro para Facebook.

**Fecha de entrega: 8 de noviembre de 2019**





# Eliminar estudiante de manera individual (06/11)

Continuaremos con el tutorial de consulta

## Paso 1: Crear método de eliminación en el servicio

En **estudiante.service.ts**, se declara un nuevo método para realizar la eliminación del usuario por ID

```

File Edit Selection View Go Debug Terminal Help estudiante.service.ts - detail.page.ts - detail.page.html
eje06 > src > app > services > estudiante.service.ts > ...
1 import { Injectable } from '@angular/core';
2 import { AngularFirestore, AngularFirestoreDocument } from '@angular/fire/firestore';
3
4 declare interface Estudiante {
5   nombre: string;
6   apellido: string;
7   edad: number;
8   email: string;
9 }
10
11 @Injectable()
12 export class EstudianteService {
13   constructor(private firestore: AngularFirestore) { }
14
15   getStudents(): Observable<AngularFirestoreCollection<Estudiante>> {
16     return this.firestore.collection('estudiantes').snapshotChanges();
17   }
18
19   updateStudent(student: Estudiante, id: string){
20     this.firestore.doc(`estudiantes/${id}`).update(student);
21   }
22
23   deleteStudent(id: string){
24     this.firestore.doc(`estudiantes/${id}`).delete();
25   }
26 }

```

PROMPTS OUTPUT DEBUG CONSOLE TERMINAL

```

[ng] i ｢wsl｣: Compiling...
[ng] Date: 2019-11-06T15:17:55-0922 - Hash: 8c518597be6dbf197ae8
[ng] 100 unchanged chunks
[ng] chunk {detail-detail-module} detail-detail-module.js, detail-detail-module.js.map (detail-detail-module) 7.49 KB [rendered]
[ng] Time: 67ms
[ng] i ｢wsl｣: Compiled successfully.

```

File Explorer: File, Edit, Selection, View, Go, Debug, Terminal, Help, estudiante.service.ts - detail.page.ts - detail.page.html

Terminal: node

Status Bar: 1a25, Ctrl F, Spacebar, UTF-8, LF, TypeScript 3.6.3, 08:18 a.m., 06/11/2019

## Paso 2: Crear método en details.page.ts

1. En **detail.page.ts**, crearemos un método de borrado que invocará al método creado anteriormente en **estudiante.service.ts**, el cual recibirá el parámetro ID.
2. Agregaremos un método asíncrono, el cual creará un toast que mostrará el mensaje que muestre al usuario que se ha eliminado el estudiante.
3. Modificaremos el método de borrado para agregar el método que mostrará el toast

```

File Edit Selection View Go Debug Terminal Help detail.page.ts - Programa06 - Visual Studio Code
eje06 > src > app > detail > detail.page.ts > DetailPage
1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute, Router } from '@angular/router';
3 import { EstudianteService } from '../services/estudiante.service';
4 import { ToastModule } from 'primeng/toast';
5
6 declare interface Estudiante {
7   nombre: string;
8   apellido: string;
9   edad: number;
10  email: string;
11 }
12
13 @Component({
14   selector: 'app-detail-page',
15   templateUrl: './detail.page.html',
16   styleUrls: ['./detail.page.css']
17 })
18
19 export class DetailPage implements OnInit {
20   estudiante: Estudiante;
21
22   constructor(private route: ActivatedRoute, private router: Router, private service: EstudianteService, private toast: ToastModule) { }
23
24   ngOnInit() {
25   }
26
27   delete(id: string){
28     this.service.deleteStudent(id);
29     this.presentToast();
30     this.router.navigate(['']);
31   }
32
33   async presentToast(){
34     const t = await this.toast.create({
35       message: 'Estudiante eliminado',
36       duration: 2000
37     });
38     t.present();
39   }
40 }

```



## Paso 3: Modificar la vista

modificaremos la página **detail.page.html**, en el fab button que contiene el icono trash, agregaremos un evento para mandar a llamar al método de borrado, mandando el ID del estudiante clickeado.

The screenshot shows the Visual Studio Code interface with the file "detail.page.html" open. The code is an Ionic template for a student detail page. It includes an ion-fab-list with four buttons: success (checkmark), warning (close), danger (trash), and a neutral one (add). The trash button has a click event that calls the delete function with the student's ID. The page also contains an ion-card with student information like name, control number, and a card content area.

```
File | Edit | Selection | View | Go | Debug | Terminal | Help | detail.page.html - ProgramasU2 | Visual Studio Code
EXPLORER: detail.page.html
OPEN EDITORS: detail.page.html
PROGRAMASU2: ej03, ej04, ej05, ej06, eje, node_modules, .git, app, detail, detail.module.js, detail.page.html, detail.page.ts, detail.page.ts, detail.page.ts, detail.page.ts, detail.page.ts, home, home.module.ts
12 <ion-fab-button>
13   <ion-icon name="add"></ion-icon>
14 </ion-fab-button>
15 <ion-fab-list>
16   <ion-fab-button color="success">
17     <ion-icon name="checkmark"></ion-icon>
18   </ion-fab-button>
19   <ion-fab-button color="warning">
20     <ion-icon name="close"></ion-icon>
21   </ion-fab-button>
22   <ion-fab-button color="danger" (click)="delete(student.id)">
23     <ion-icon name="trash"></ion-icon>
24   </ion-fab-button>
25 </ion-fab-list>
26 </ion-fab>
27 <ion-card text-center>
28   <ion-card-header>
29     <ion-card-subtitle>{{student.name}}</ion-card-subtitle>
30     <ion-card-title>{{student.controlNumber}}</ion-card-title>
31   <ion-card-header>
32   <ion-card-content>
33     <ion-item>{{student.name}}</ion-item>
```

## Retos

### Reto 1: Convertir a Input y actualizar (09/11)

En la aplicación, dentro de la página de detalles, se sustituirán los label con inputs, con el fin de poder modificarlos, y también agregaremos la funcionalidad para poder actualizar los datos

**Fecha de entrega: 9 de noviembre de 2019**