IONIC

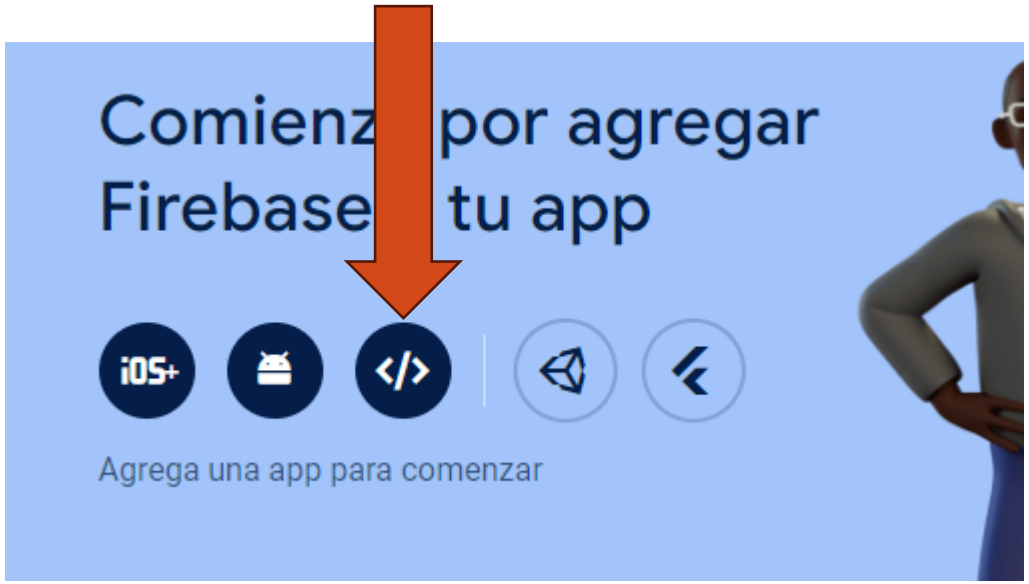
Firebase – Realtime Database

IONIC

- Instalación (Ionic con Angular):
  - npm install firebase @angular/fire
  - Crear proyecto Firebase
  - Crear Realtime Database Firebase (modo prueba)
  - Agregar autenticación con correo electrónico/contraseña

IONIC

- Instalación (Ionic con Angular):
  - Agregar Firebase a la aplicación



IONIC

- Configuración app.module.ts (o el servicio en modo standalone):

```
import { AngularFireModule } from "@angular/fire/compat";

const firebaseConfig = {
  apiKey: "xxx",
  authDomain: "xxx",
  databaseURL: "xxx",
  projectId: "xxx",
  storageBucket: "xxx",
  messagingSenderId: "xxx",
  appId: "xxx"
}

imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule,
    AngularFireModule.initializeApp(firebaseConfig),
```

IONIC

- Instalación (Ionic con Angular):
  - Crear Realtime Database Firebase (modo prueba)
  - ~~Agregar autenticación con correo electrónico/contraseña~~

IONIC

- Ficheros environment.ts y environment.prod.ts:

```
export const environment = {
  production: false,
  firebaseConfig: {
    apiKey: “----------------------",
    authDomain: “-----------",
    projectId: “-----------",
    storageBucket: “--------",
    messagingSenderId: “-----------------------",
    appId: “--------------------",
    databaseURL: "https://----------.europe-west1.firebasedatabase.app"
  }
};
```

IONIC

- Imports e inicialización:

```
import { Injectable } from '@angular/core';
import { initializeApp } from 'firebase/app';
import { Database, DatabaseReference, getDatabase, onValue, ref, remove, set, push } from
"firebase/database";
import { environment } from 'src/environments/environment';

@Injectable({
  providedIn: 'root',
})
export class FirebaseService {
  private db: Database;
  private dbRefTareas: DatabaseReference;
  private dbRefCategorias: DatabaseReference;
  public tareas: any[] = [];
  public categorias: any[] = [];
```

IONIC

- Inicialización y suscripción:

```
constructor() {
   initializeApp(environment.firebaseConfig);
   this.db = getDatabase();
   this.dbRefTareas = ref(this.db, 'tareas/');
   this.dbRefCategorias = ref(this.db, 'categorias/');

   onValue(this.dbRefCategorias, (snapshot) => {
     this.categorias = [];
     snapshot.forEach((childSnapshot) => {
       this.categorias.push((childSnapshot));
     });
   }, {
     onlyOnce: false
   });
   onValue(this.dbRefTareas, (snapshot) => {
     this.tareas = [];
     snapshot.forEach((childSnapshot) => {
       this.tareas.push((childSnapshot));
     });
   }, {
     onlyOnce: false
   });
 }
```

IONIC

- Creación y borrado:

```
public crearCategoria(categoria: string) {
    const nuevaCategoria = push(this.dbRefCategorias);
    set(nuevaCategoria, { categoria: categoria });
  }
  public borrarCategoria(key: string) {
    remove(ref(this.db, 'categorias/' + key));
  }
  public crearTarea(tarea: string, categoria: string) {
    const dbRefNuevaTarea = ref(this.db, `tareas/`);
    const nuevaTarea = push(dbRefNuevaTarea);
    set(nuevaTarea, { categoria: categoria, tarea: tarea });
  }
  public borrarTarea(key: string) {
    remove(ref(this.db, 'tareas/' + key));
  }
```

IONIC

- Uso:
  - Lectura (por clave).

```
leerUno() {
    const db = getDatabase();
    const usuario = ref(db, 'usuarios/fpaniagua');
    onValue(usuario, (snapshot) => {
      const data = snapshot.val();
      console.log(data);
    });
  }
```

IONIC

- Uso:
  - Lectura rama.

```
leerTodos() {
    const db = getDatabase();
    onValue(ref(db, '/usuarios/'), (snapshot) => {
      console.log(snapshot.val());;
      console.log(snapshot.child("fpaniagua").val());
      snapshot.forEach((child) => {
        console.log(child.key, child.val());
      });
    }, {
      onlyOnce: false //Para actualizar en cada momento
    });
  }
```

IONIC

- Uso:
  - Borrado

```
borrar() {
    console.log("Borrando...");
    const db = getDatabase();
    remove(ref(db, 'usuarios/' + "omartin"));
}
```