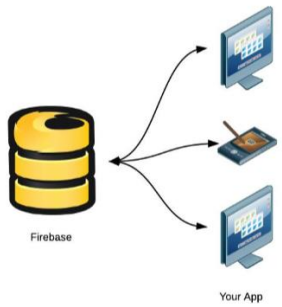
Bases de datos en tiempo real con Angular

La base de datos en tiempo real de Firebase (Firebase Realtime Database) es sin duda uno de los servicios más populares de la plataforma. Contar con la capacidad de almacenar datos "en la nube" es uno de los requerimientos de los que pocas aplicaciones actuales pueden escapar, y poder hacerlo sin necesidad de preocuparnos por toda la infraestructura de servidor necesaria es toda una ventaja.

Firebase nos proporciona un servicio de base de datos con la particularidad de ser en tiempo real. ¿Pero qué significa esto? La sincronización en tiempo real implica que cualquier cambio realizado en los datos por cualquier cliente (usuario, aplicación, dispositivo…) se sincronizarán automáticamente y de forma inmediata (siempre que la conexión lo permita) en el resto de clientes, sin necesidad de que éstos vuelvan a consultar los datos. ¿Y si se pierde temporalmente la conexión? No hay problema, Firebase también está preparado para permitir interactuar con la base de datos cuando el dispositivo no tiene conexión (siempre dentro de unos límites) mediante un sistema de cachés y colas de escritura locales. Cuando el dispositivo vuelve a tener conexión, los cambios locales serán sincronizados automáticamente con la base de datos y, si aplica, con el resto de clientes conectados a ella.



Ejemplo: Crear una app en angular y firebase que permita insertar, consultar, modificar y eliminar datos.

Para el desarrollo de este ejemplo, debe seguir los siguientes pasos:

Paso 1: Crear proyecto en angular CLI

Inicialmente se debe crear un proyecto en Angular CLI

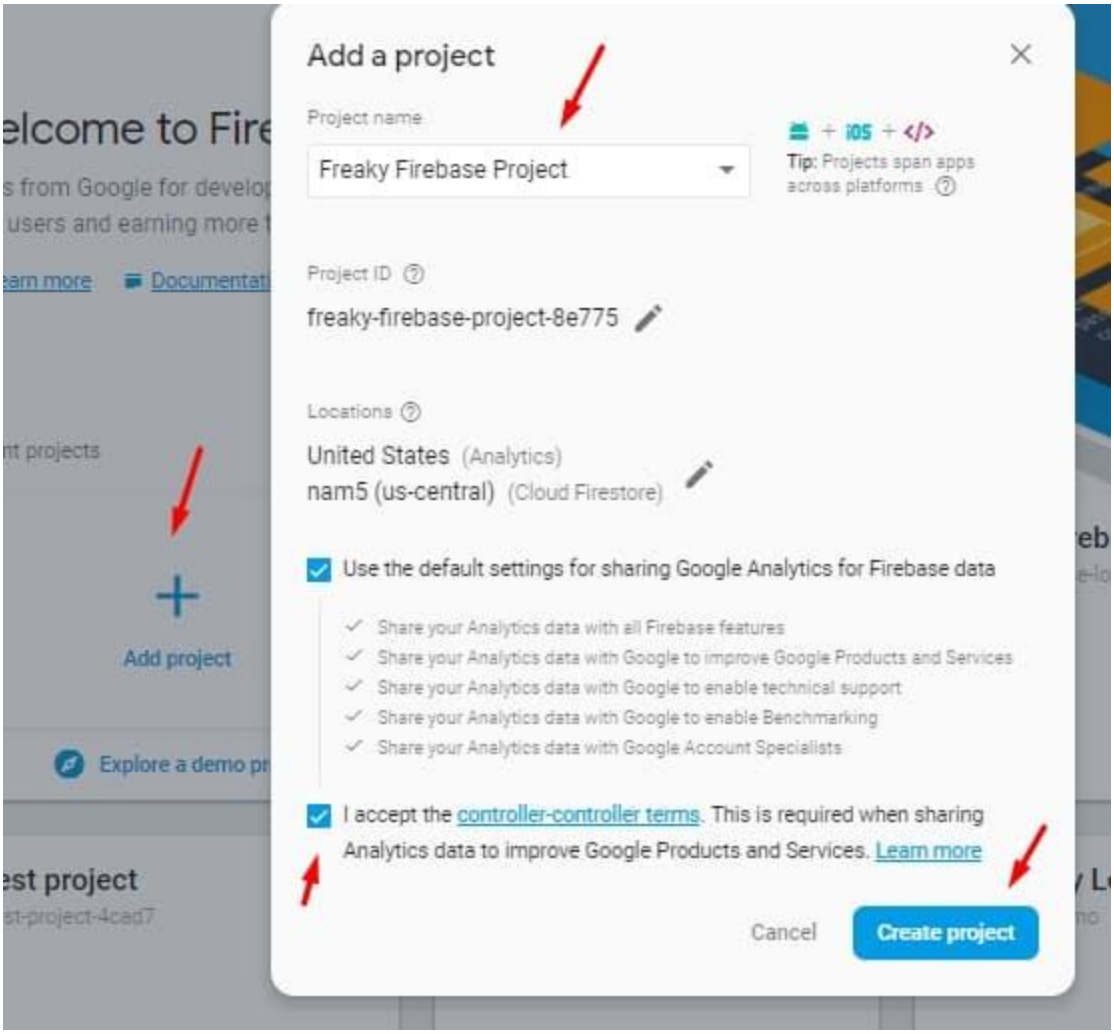
```
$ ng new Angular7Firestore
$ cd Angular7Firestore
```

luego, para el enrutamiento, seleccione no, ya que en esta aplicación no usaremos crear ningún componente nuevo, para el estilo elegimos SCSS.

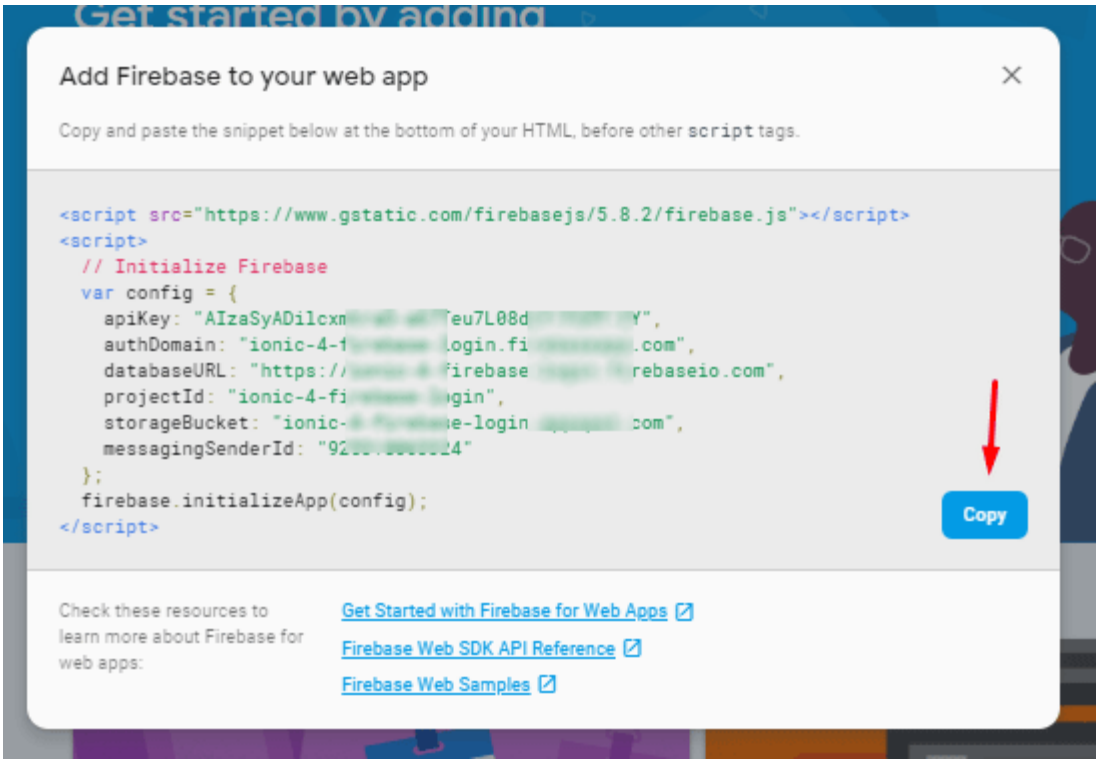
```
D:\JollysLAB\ng>ng new Angular7Firestore
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? Sass    [ http://sass-lang.com
```

Paso 2: Creación de proyecto Firebase

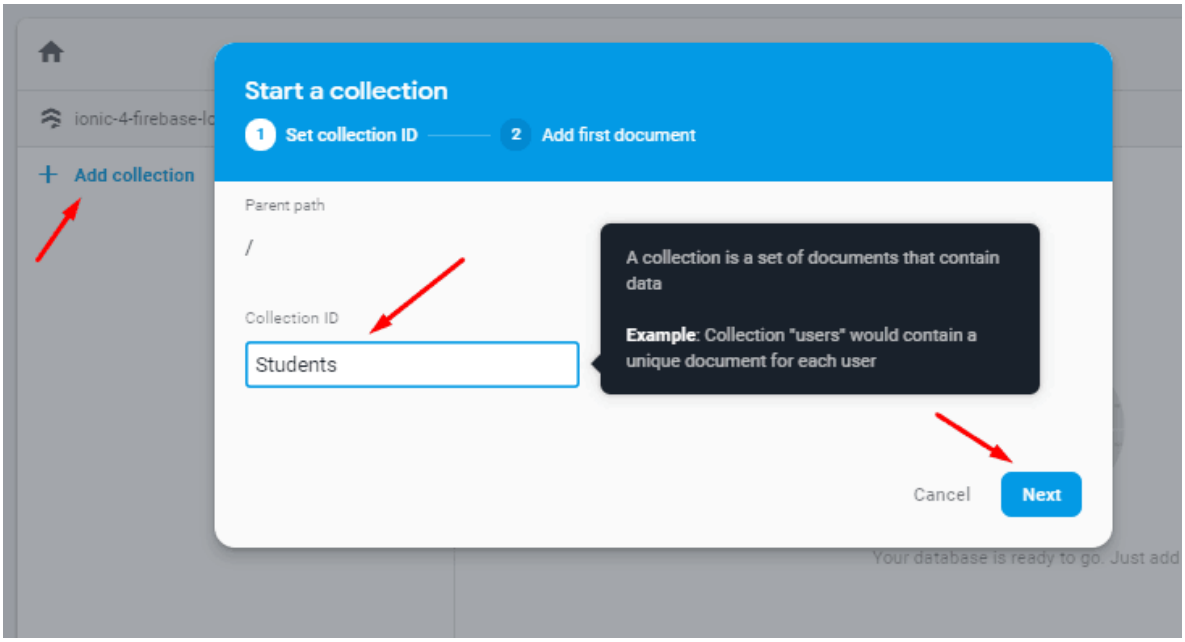
Haga clic en "Agregar proyecto" y luego ingrese la información relacionada con la aplicación, haga clic en "crear"



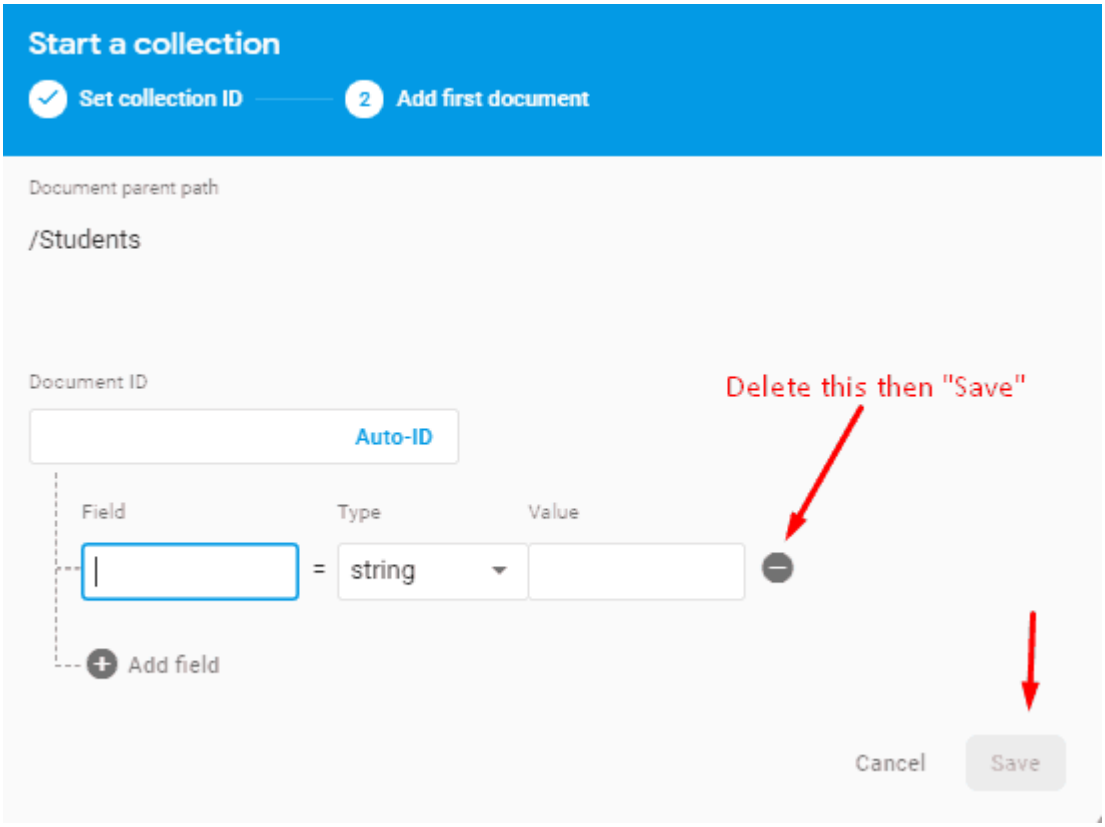
A continuación, haga clic en el icono " **Aplicación web** " para obtener el texto de configuración con información secreta de la aplicación que agregaremos en nuestra aplicación para comunicarnos con los servicios relacionados con este proyecto de Firebase.

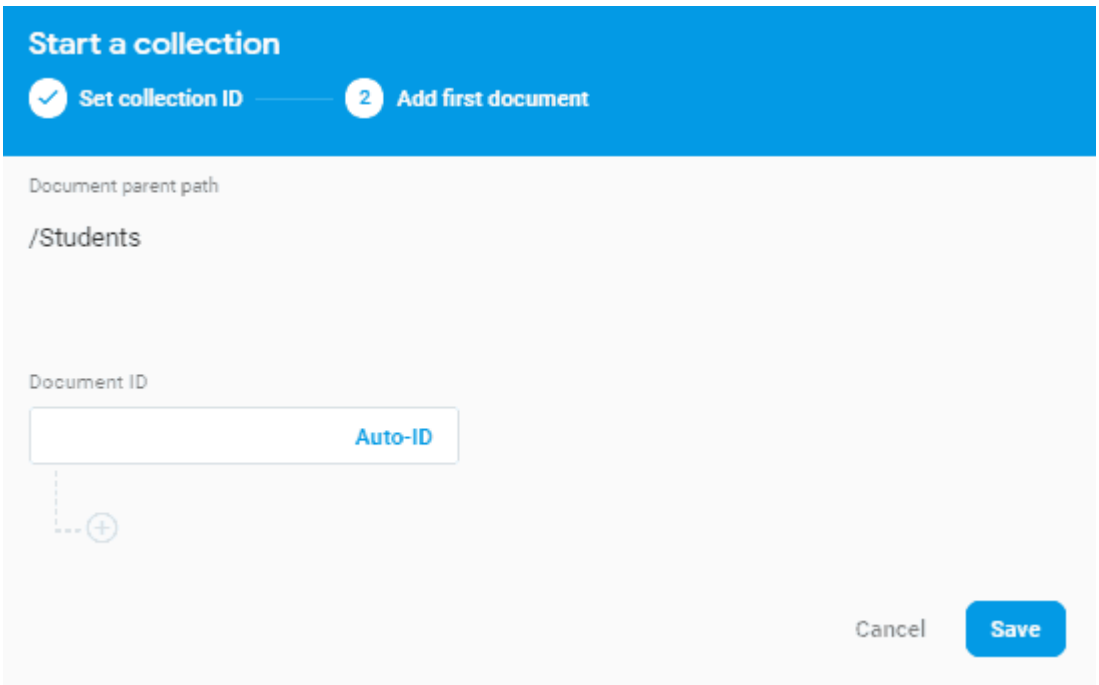


Luego habilite el servicio Firestore Database en su proyecto Firebase.
Haga clic en " **Base de datos** " en la barra lateral izquierda, luego haga clic en " **+ Agregar colección** " ingrese el nombre "Estudiantes", puede agregar lo que desee. Aquí estamos creando datos de ejemplo para estudiantes. Luego presione " **Siguiente** "



En la siguiente pantalla, elimine el registro en foco para este ejemplo o puede ingresar cualquier dato de fila ficticia. Agregaremos directamente desde la aplicación. Después de eliminar, presione **Guardar**





Ahora tenemos el proyecto Firebase y la base de datos de Firestore con una colección " *Students* " está lista, el siguiente paso es conectar nuestra aplicación con Firebase y la base de datos de Firestore.

Paso 3: Agregar Firebase en una aplicación angular

Ahora abra el archivo de entorno en la ubicación " **~ Angular7Firestore / src / assets / environment.ts** " y luego agregue las credenciales del proyecto Firebase como se muestra a continuación

```
export const environment = {
  production: false,
  firebase: {
    apiKey: "YOUR_apiKey",
    authDomain: "YOUR_authDomain",
    databaseURL: "YOUR_databaseURL",
    projectId: "YOUR_projectId",
    storageBucket: "YOUR_storageBucket",
    messagingSenderId: "YOUR_messagingSenderId"
  }
 };
```

Paso 4: Instalar Firebase en la aplicación

Instale Firebase SDK y @ angular / fire paquete ejecutando debajo del comando CLI

```
$ npm install --save firebase @angular/fire
```

En el archivo del módulo principal de la aplicación, importaremos Firebase e inicializaremos Firebase con credenciales de entorno.

Paso 6: Configurar el app.module.ts

Reemplace el código siguiente en su archivo **app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';


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


@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,

    AngularFireModule.initializeApp(environment.firebase),
    AngularFirestoreModule,
    AngularFireDatabaseModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Modulo Firebase Angular

Modulo Firebase Bases de datos en tiempo real

Modulo Firestore Angular

Paso 7: Crear un servicio con los métodos CRUD de Firestore

Cree un servicio con métodos de operación CRUD (ng generate service). Los métodos de servicio se pueden usar en cualquier lugar sin tener que volver a escribir una y otra vez. La estructura de rutas donde se crea el servicio es la siguiente:

"~ Angular7Firestore / src / app / service / **crud.service.ts** ",

Los métodos implementados en el servicio se describen a continuación:

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

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

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

  constructor(private firestore: AngularFirestore ) { }


  create_NewStudent(record) {
    return this.firestore.collection('Students').add(record);
  }

  read_Students() {
    return this.firestore.collection('Students').snapshotChanges();
  }

  update_Student(recordID,record){
    this.firestore.doc('Students/' + recordID).update(record);
  }

  delete_Student(record_id) {
    this.firestore.doc('Students/' + record_id).delete();
  }
}
```

crea un nuevo registro en la colección especificada usando el método *add*

read_Students : llame al método *snapshotChanges* que obtendrá registros y también lo suscribirá para obtener actualizaciones

update_Student : actualice el registro tomando la identificación y luego llamando al método de *actualización*

delete_Student : llama al método de *eliminación* tomando la identificación del registro

El archivo app.component.html tiene la siguiente estructura

```
<div class="container">

  <h1 style="margin-top: 25px;">
    {{title}}
  </h1>
  <div class="row" style="margin-top: 25px;">
    <div class="col-md-3">
      <input type="text" class="form-control" [(ngModel)]="studentName" placeholder="Name">
    </div>
    <div class="col-md-3">
      <input type="text" class="form-control" [(ngModel)]="studentAge" placeholder="Age">
    </div>
    <div class="col-md-3">
      <input type="text" class="form-control" [(ngModel)]="studentAddress" placeholder="Address">
    </div>
    <div class="col-md-3">
      <button type="button" (click)="CreateRecord()" [disabled]="!studentName || !studentAge || !studentAddress" class="btn
      btn-primary">+
        Create Student</button>
    </div>
  </div>
```

Datos de entrada

Funcion para crear registros en el controller

```
  <div class="row" style="margin-top: 25px;">
    <div class="col-md-3">
      <div class="card" style="width: 18rem;" *ngFor="let item of students">
        <div class="card-body" *ngIf="!item.isEdit; else elseBlock">
          <h5 class="card-title">{{item.Name}}</h5>
          <h6 class="card-subtitle mb-2 text-muted">Age: {{item.Age}} Years</h6>
          <p class="card-text">Address: {{item.Address}}</p>
          <a href="#" class="card-link" (click)="EditRecord(item)">Edit</a>
          <a href="#" class="card-link" (click)="RemoveRecord(item.id)">Delete</a>
        </div>
        <ng-template #elseBlock>
          <div class="card-body">
            <h5 class="card-title">Edit</h5>
            <div class="form-group">
              <div class="row">
                <div class="col-md-12">
                  <input type="text" class="form-control" [(ngModel)]="item.EditName" placeholder="Edit Name">
                </div>
                <div class="col-md-12">
                  <input type="text" class="form-control" [(ngModel)]="item.EditAge" placeholder="Edit Age">
                </div>
                <div class="col-md-12">
                  <input type="text" class="form-control" [(ngModel)]="item.EditAddress" placeholder="Edit Address">
                </div>
              </div>
            </div>
            <a href="#" class="card-link" (click)="item.isEdit = false">Cancel</a>
            <a href="#" class="card-link" (click)="UpdateRecord(item)">Update</a>
          </div>
        </ng-template>
      </div>
    </div>
  </div>
</div>
```

ngFor para recorrer datos que vienen de Firebase

Se imprime nombre

Se imprime edad

Se imprime dir

Boton editar

Boton Borrar

Aplica para editar datos

El archivo app.component.ts tiene la siguiente estructura

```typescript
import { Component, OnInit } from '@angular/core';
import { CrudService } from './service/crud.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent  implements OnInit {
  title = 'Firestore CRUD Operations Students App';

  students: any;
  studentName: string;
  studentAge: number;
  studentAddress: string;

  constructor(private crudService: CrudService) { }

  ngOnInit() {
    this.crudService.read_Students().subscribe(data => {

      this.students = data.map(e => {
        return {
          id: e.payload.doc.id,
          isEdit: false,
          Name: e.payload.doc.data()['Name'],
          Age: e.payload.doc.data()['Age'],
          Address: e.payload.doc.data()['Address'],
        };
      })
      console.log(this.students);

    });
  }
```

Datos de entrada

Refrescar datos del servicio Fire

Datos del servicio

```
  CreateRecord() {
    let record = {};
    record['Name'] = this.studentName;
    record['Age'] = this.studentAge;
    record['Address'] = this.studentAddress;
    this.crudService.create_NewStudent(record).then(resp => {
      this.studentName = "";
      this.studentAge = undefined;
      this.studentAddress = "";
      console.log(resp);
    })
      .catch(error => {
        console.log(error);
      });
  }
  RemoveRecord(rowID) {
    this.crudService.delete_Student(rowID);
  }
  EditRecord(record) {
    record.isEdit = true;
    record.EditName = record.Name;
    record.EditAge = record.Age;
    record.EditAddress = record.Address;
  }
  UpdateRecord(recordRow) {
    let record = {};
    record['Name'] = recordRow.EditName;
    record['Age'] = recordRow.EditAge;
    record['Address'] = recordRow.EditAddress;
    this.crudService.update_Student(recordRow.id, record);
    recordRow.isEdit = false;
  }

}
```

Datos de entrada definidos por el usuario

servicio para crear estudiantes

Reinicio de datos al insertar

Error al insertar

Borrar a partir del id

Datos editados

Actualizar datos

servicio de actualizacion por id

Al ejecutar la aplicación, podemos enviar y recibir datos desde FireCloud