



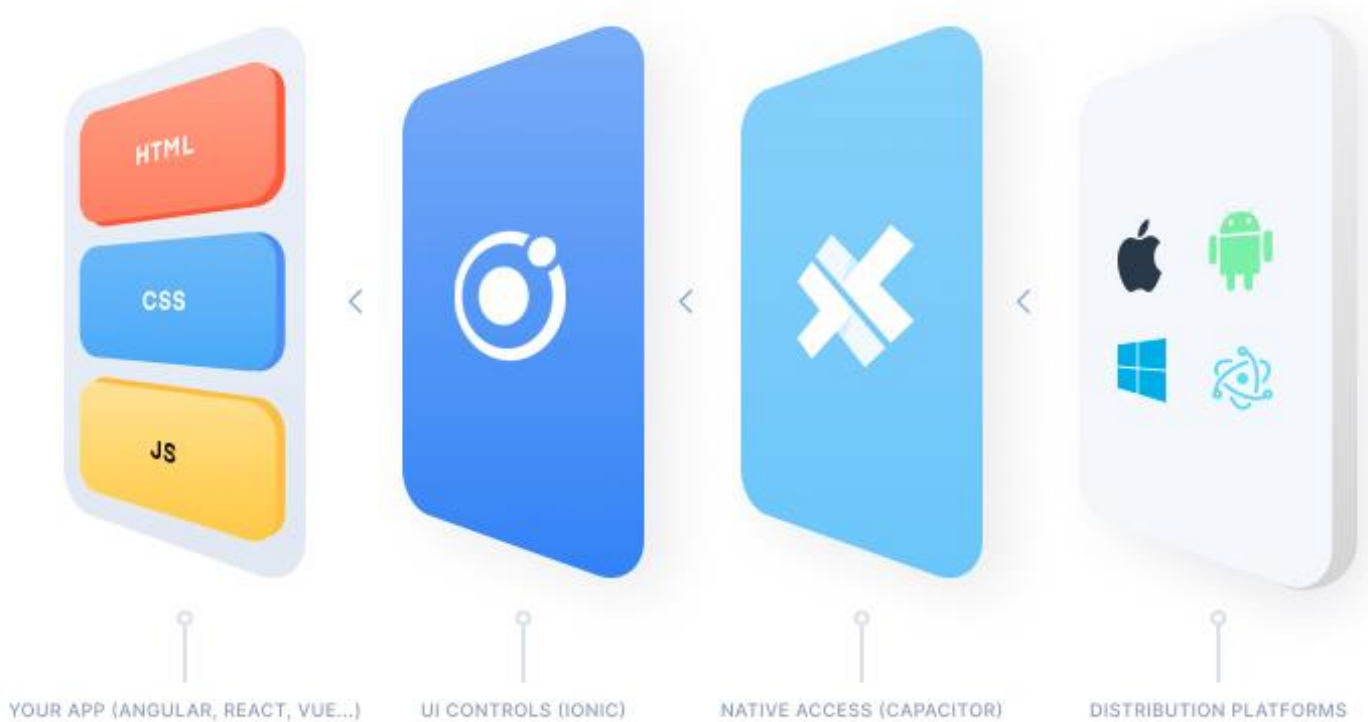
---

# PROGRAMACIÓN DE DISPOSITIVOS MOVILES

---

## SQLITE IONIC

JOAN MANUEL GREGORIO PÉREZ, M.A.



**JMGREP DEVELOPER SCHOOL**

## Resultados de Aprendizaje

### Que el estudiante:

Diseñe aplicaciones Híbridas utilizando el framework IONIC.

Diseñe aplicaciones con navegación

Pueda crear formulario de login y validación en IONIC

## Introducción

A lo largo de esta guía paso a paso, descubrirá cómo crear la operación SQLite CRUD en la aplicación Ionic Angular con la ayuda del paquete SQLite.

Idealmente, SQLite es una base de datos robusta que se utiliza exclusivamente para crear un sistema de base de datos en dispositivos más pequeños que nos permiten realizar consultas SQL y construir una aplicación simple con operaciones CRUD.

De manera genérica, en esta guía rápida, demostraremos cómo crear una base de datos SQLite, crear una tabla e insertarla en la base de datos del dispositivo, agregar o insertar datos en la tabla, buscar, obtener o leer datos de la tabla, eliminar una fila de la tabla y actualice la fila de la tabla en la aplicación móvil sin conexión ionic sqlite.

### ¿Que es IONIC?

Ionic Framework es un SDK de frontend de código abierto para desarrollar aplicaciones híbridas basado en tecnologías web (HTML, CSS y JS). Es decir, un framework que nos permite desarrollar aplicaciones para iOS nativo, Android y la web, desde una única base de código. Su compatibilidad y, gracias a la implementación de Cordova e Ionic Native, hacen posible trabajar con componentes híbridos. Se integra con los principales frameworks de frontend, como Angular, React y Vue, aunque también se puede usar Vanilla JavaScript. Este framework fue creado en 2013 por Drifty Co. y hasta la llegada de React Native ha sido una de las tecnologías líderes para el desarrollo de aplicaciones móviles híbridas.



### Características de Ionic Framework

Creado por Joan Manuel Gregorio Pérez, M.A.  
Curso de IONIC Desarrollo de Aplicaciones móviles

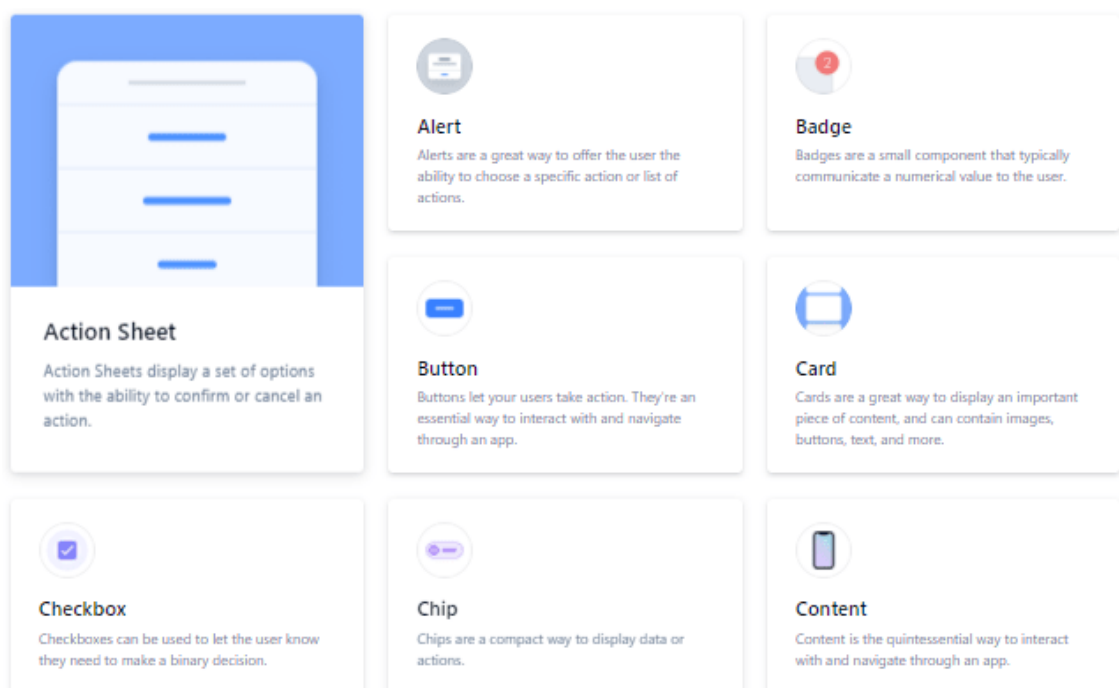
### **Ionic se caracteriza por ser un Frameworks que:**

- Permite desarrollar y desplegar aplicaciones híbridas, que funcionan en múltiples plataformas, como iOS nativo, Android, escritorio y la web (como una aplicación web progresiva), todo ello con una única base de código.
- Ofrece un **diseño limpio, sencillo y funcional**.
- Emplea **Capacitor (o Cordova)** para implementar de forma nativa o se ejecuta en el navegador como una aplicación web progresiva.
- Está construido sobre tecnologías web: **HTML, CSS y JavaScript**.
- Se puede usar con los Frameworks FrontEnd más populares, como **Angular, React y Vue**.

### **Componentes UI de Ionic**

La facilidad que Ionic ofrece para el diseño de interfaces es uno de sus puntos fuertes y lo consigue gracias a sus componentes. Los Componentes de Ionic son bloques de construcción de alto nivel que nos ayudan a construir de forma rápida la interfaz de usuario de nuestra aplicación. Algunos de sus componentes UI principales son:

- **Tarjetas (ion-cards).** Uno de los componentes estándar de la interfaz de usuario. Sirve como punto de entrada a información más detallada. Suele formarse por encabezado, título, subtítulo y contenido.
- **Listas (ion-lists).** Compuestas de varias filas de elementos, que pueden incluir texto, botones, iconos y miniaturas, entre otros. Las listas de Ionic admiten interacciones diversas como deslizar para revelar opciones o arrastrar para reordenar o eliminar elementos.
- **Pestañas (ion-tabs).** Normalmente se utilizan junto a barras de pestañas (ion-tab-bars) para implementar una navegación basada en pestañas que se comporte de forma similar a una aplicación nativa.



No.	Requerimientos	Cantidad
1	Memoria USB	1
2	PC , Node, Npm, Visual Studio Code y navegador web, SQLite, Android Studio	1
3	Guía de laboratorio	1

### Parte A: Instalación de Ionic

En esta sección vamos a ver en detalle todo el proceso de instalación de IONIC y de todas las dependencias necesarias para que funcione. Estas son:

- NodeJs
- IONIC
- Otras dependencias

### Instalar NodeJS

En primer lugar tenemos que instalar el gestor de paquetes de NodeJs (npm) para poder instalar el propio Ionic y algunas otras dependencias que nos harán falta. Para instalarlo simplemente podemos descargarlo e instalarlo desde su Web:

<https://nodejs.org/es/download/>



### Descargas

Versión actual: **14.17.5** (includes npm 6.14.14)

Descargue el código fuente de Node.js o un instalador pre-compilado para su plataforma, y comience a desarrollar hoy.

	LTS Recomendado para la mayoría	Actual Últimas características
 <b>Instalador Windows</b> <small>node-v14.17.5-x64.msi</small>	 <b>Instalador macOS</b> <small>node-v14.17.5.pkg</small>	 <b>Código Fuente</b> <small>node-v14.17.5.tar.gz</small>

Instalador Windows (.msi)	32-bit	64-bit
Binario Windows (.zip)	32-bit	64-bit
Instalador macOS (.pkg)	64-bit	
Binario macOS (.tar.gz)	64-bit	
Binario Linux (x64)	64-bit	
Binario Linux (ARM)	ARMv7	ARMv8
Código Fuente	node-v14.17.5.tar.gz	

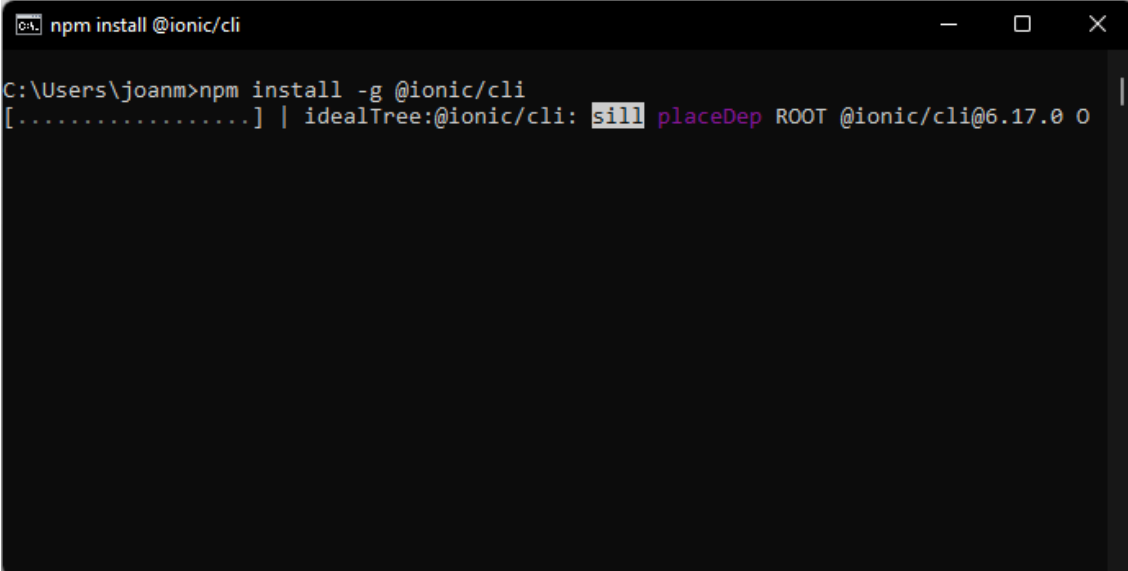
**O en Linux instalarlo usando el gestor de paquetes:**

```
sudo apt-get install nodejs  
sudo apt-get install npm
```

## Instalar IONIC

La CLI de Ionic se puede instalar globalmente con npm:

- 1- Abra la consola o terminal y ejecute el siguiente comando:  
*`npm install -g @ionic/cli`*



```
C:\> npm install @ionic/cli  
C:\Users\joanm> npm install -g @ionic/cli  
[.....] | idealTree:@ionic/cli: sill placeDep ROOT @ionic/cli@6.17.0 0
```

Validar la versión instalada con el siguiente comando:

```
ionic -v
```

```
C:\WINDOWS\system32\cmd.exe

ionic CLI 6.17.0

Usage:
  $ ionic <command> [<args>] [--help] [--verbose] [--quiet] [--no-interactive] [--no-color] [--confirm] [options]

Global Commands:
  completion ..... (experimental) Enables tab-completion for Ionic CLI commands.
  config <subcommand> ..... Manage CLI and project config values (subcommands: get, set, unset)
  deploy <subcommand> ..... (paid) Appflow Deploy functionality (subcommands: manifest)
  docs ..... Open the Ionic documentation website
  info ..... Print project, system, and environment information
  init ..... (beta) Initialize existing projects with Ionic
  login ..... Log in to Ionic
  logout ..... Log out of Ionic
  signup ..... Create an Ionic account
  ssh <subcommand> ..... Commands for configuring SSH keys (subcommands: add, delete, generate, list,
  setup, use)
  start ..... Create a new project

Project Commands:
  You are not in a project directory.
```

**SQLite** es un sistema de gestión de base de datos relacional, contenida en una biblioteca muy pequeña, escrita en C.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, **SQLite** no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo.

Esta es una muy buena opción cuando necesitamos programar aplicaciones pequeñas y que manejen base de datos.

Entre sus principales ventajas están:

En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

- Es de libre distribución
- No requiere configuración
- No se requiere uso del servidor
- Multiplataforma (Windows, Linux, Android, Mac, etc.)

Se maneja como un único archivo y puede estar situado en cualquier parte; esto permite poder usarlo incluso desde una memoria USB.

- Es transaccional
- Puede ser utilizado por casi todos los lenguajes de programación (C, .NET, Visual Basic 6, Python, Delphi, PHP, Java, etc.)
- Utiliza registros de longitud variable, lo cual permite ahorrar espacio, permitiendo que la base de datos sea más pequeña.

Debido a sus múltiples ventajas, este sistema de base de datos ya es usado por diversas empresas y productos, como: Mozilla Firefox, Adobe Photoshop Elements, varias aplicaciones de Apple, Skype y más.

Por experiencia, puedo recomendarte esta base de datos, siempre y cuando tu aplicación no realice muchos accesos simultáneos a la base de datos, de lo contrario, sería mejor utilizar las bases de datos tradicionales de tipo cliente-servidor.

En los navegadores web, dependemos principalmente de los objetos de almacenamiento local para almacenar los datos de forma local, mientras que, en los dispositivos móviles, Cordova ofrece una opción impecable para crear una aplicación fuera de línea. El paquete SQLite le permite acceder a las bases de datos SQLite en los dispositivos y eso con demasiada facilidad y rapidez. Independientemente de la plataforma en la que esté creando una aplicación, no se preocupe; SQLite cubre todas las plataformas principales, como Android, iOS, macOS, no solo sino también Windows.

### Ejemplo de aplicación móvil CRUD fuera de línea Ionic 5 SQLite

Paso 1: configurar el entorno iónico

Paso 2: Actualizar rutas

Paso 3: Instale el complemento SQLite

Paso 4: Actualizar la clase del módulo de la aplicación

Paso 6: Cree el servicio CRUD de SQLite

Paso 7: Implementar Crear y Eliminar

Paso 8: Obtenga la recopilación de datos y la lista de visualización

Paso 9: implementar la actualización o editar

Paso 10: Pruebe la aplicación Ionic Crud

Para poder crear un nuevo proyecto debemos de tener en cuenta en donde lo queremos almacenar, para esto es muy importante movernos dentro de nuestra terminal o CMD, usaremos comandos básicos así que no te preocupes, te dejo algunos comandos que debes de tener en cuenta.

Comandos	Descripción	Sistema Operativo
ls	Listar Elementos	MacOS / Distros GNU/Linux
dir	Listar Elementos	Windows
cd	Cambio de directorio	Todos
clear	Limpiar terminal	MacOS / Distros GNU/Linux
cls	Limpiar terminal	Windows
pwd	Ruta absoluta actual	MacOS / Distros GNU/Linux
chdir	Ruta absoluta actual	Windows

Una vez hayamos pensado en donde queremos almacenar nuestro proyecto, debemos de saber donde nos encontramos dentro de nuestra terminal, regularmente siempre nos ubica en nuestra carpeta personal o la del usuario que inició sesión, si queremos asegurarnos en donde nos encontramos debemos de usar el comando **pwd** o en su defecto **chdir** dependiendo el sistema operativo en donde nos encontremos, si después de presionar **enter** nos muestra algunas carpetas conocidas como Escritorio o Desktop o Documentos o Descargas entonces nos encontramos en nuestra carpeta personal, y ahí quedará de nosotros elegir en donde queremos almacenar el proyecto.

Para este ejemplo elegiré crear mi proyecto en el Escritorio, para esto debo de usar el comando para cambiar de directorio "cd" seguido del nombre de la carpeta a donde nos dirigimos "Escritorio" quedaría así:

```
C:\WINDOWS\system32\cmd.exe

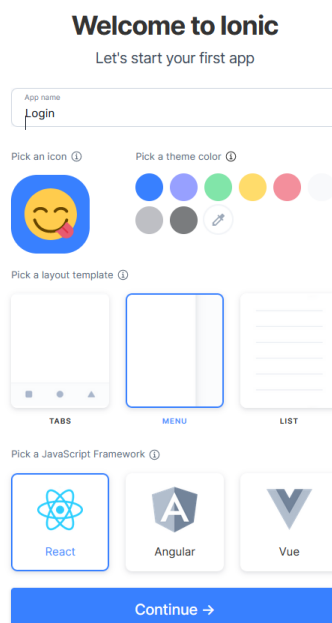
C:\Users\joanm>cd Escritorio
```

**NOTA:** debemos de cuidar que el nombre de la carpeta esté bien escrito, respetando mayúsculas, o de igual manera podemos apoyarnos del Tabulador para autocompletar lo que estemos escribiendo.

```
Windows PowerShell

C:\Users\joanm>ionic start
? Use the app creation wizard? Yes
```

*Si usamos únicamente este comando el CLI nos brindará los pasos que debemos de seguir, como ingresar el nombre de nuestro proyecto, si queremos usar un template al momento de iniciar nuestro proyecto, y también el Framework que deseamos usar con Ionic.*





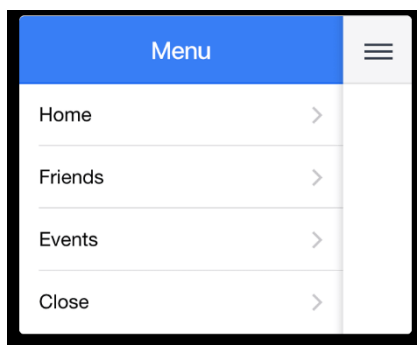
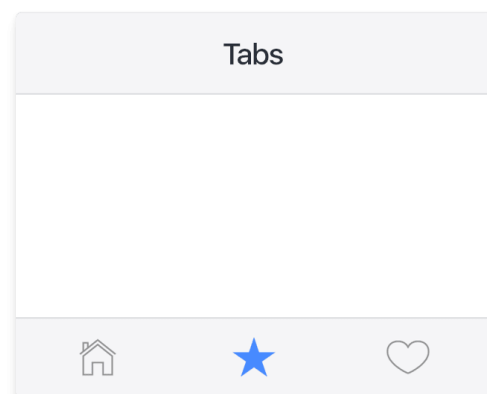
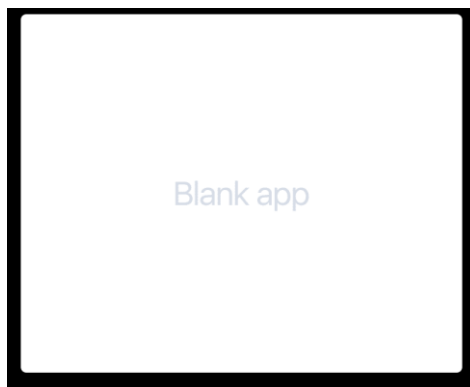
Elegimos el tipo de proyecto que queremos generar (Tbas, Menu, List) en el recuadro App Name, escribiremos el nombre del proyecto, luego procedemos a dar clic en continue y nos presentara el login de nuestra cuenta de github para guardar el proyecto en nuestro repositorio.

Si ya tenemos en mente el nombre de nuestro proyecto podemos hacer la ejecución en una sola línea simplemente agregando otros parámetros como lo indico a continuación: Muy bien ya nos encontramos en la carpeta, es momento de crear el proyecto, para esto usaremos el comando:

```
C:\WINDOWS\system32\cmd.exe
C:\Users\joanm>ionic start login blank
```

Los Templates disponibles son:

Templates	Descripción
blank	Crea un Proyecto sin Template.
tabs	Crea un proyecto con tres vistas donde puedes navegar entre ellas muy fácil.
sidemenu	Crea un proyecto con un menú lateral



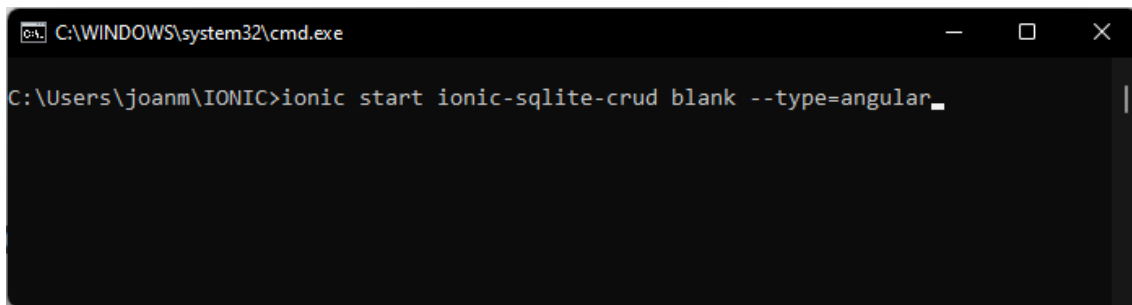
## Crear aplicación en blanco

- 1- Crear un proyecto en blanco con el siguiente comando en IONIC

*Ionic start "nombre del proyecto" "tipo de proyecto"*

En nuestro caso como crearemos un login este será el nombre del proyecto y el tipo del proyecto es en blanco

*ionic start ionic-sqlite-crud blank --type=angular*



```
C:\WINDOWS\system32\cmd.exe

C:\Users\joanm\IONIC>ionic start ionic-sqlite-crud blank --type=angular_
```

Seleccionamos **angular**

**Nos dirigimos a la carpeta de nuestro proyecto:**

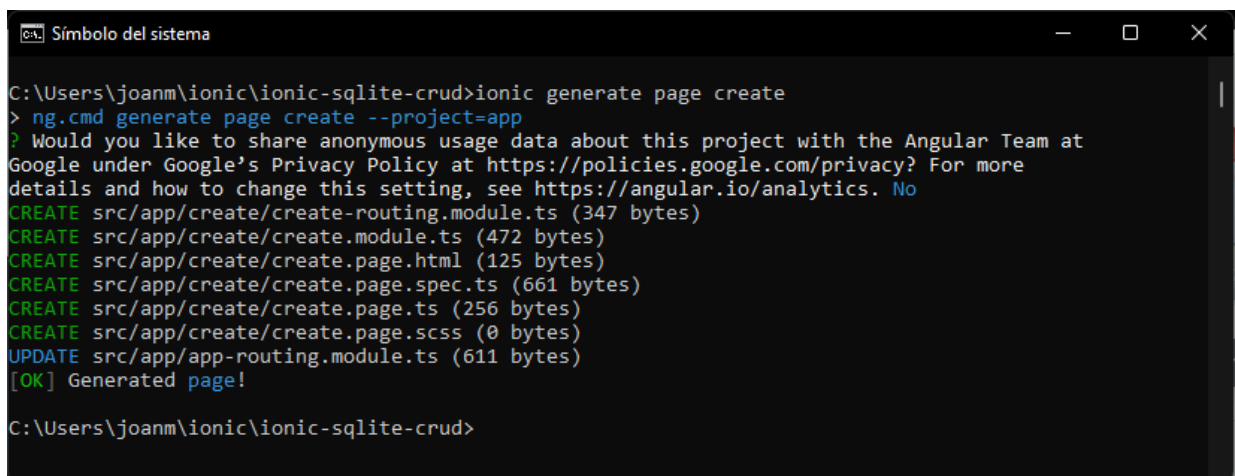
*cd ionic-sqlite-crud* - **presionamos la tecla enter**

A continuación, genere páginas o componentes de iónic con la ayuda del comando sugerido.

Asegúrese de eliminar la página de inicio porque no la necesitamos.

*ionic generate page create*

*ionic generate page edit*



```
Símbolo del sistema

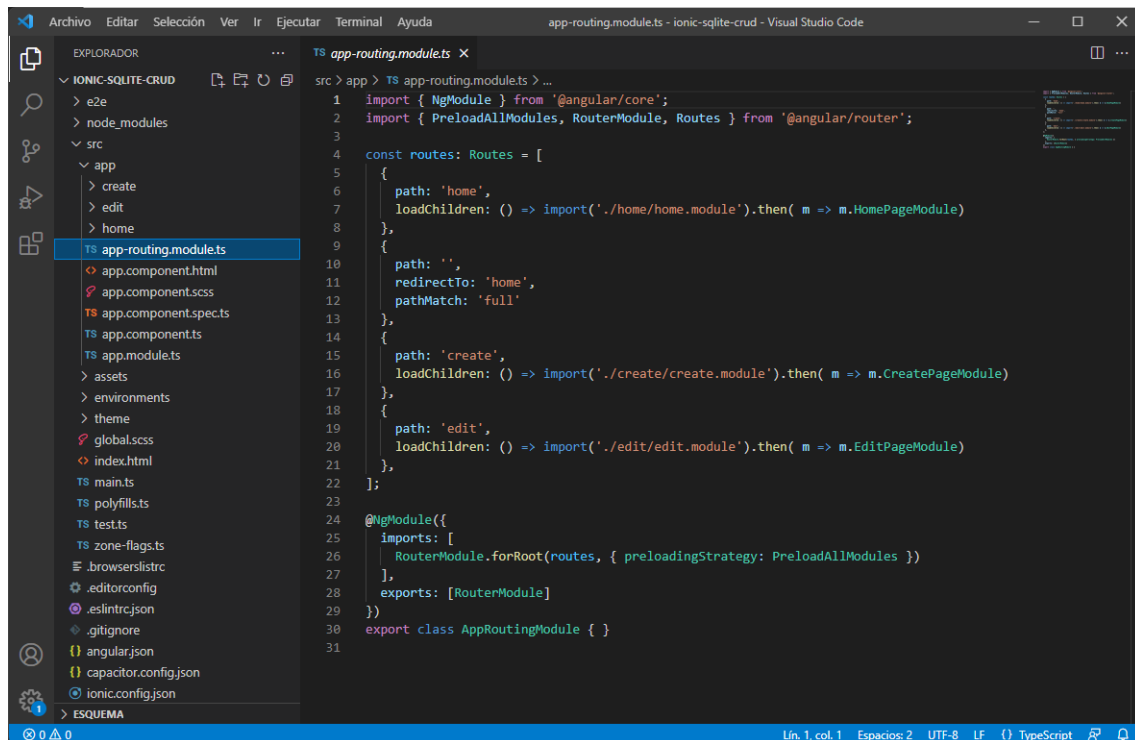
C:\Users\joanm\ionic\ionic-sqlite-crud>ionic generate page create
> ng.cmd generate page create --project=app
? Would you like to share anonymous usage data about this project with the Angular Team at
Google under Google's Privacy Policy at https://policies.google.com/privacy? For more
details and how to change this setting, see https://angular.io/analytics. No
CREATE src/app/create/create-routing.module.ts (347 bytes)
CREATE src/app/create/create.module.ts (472 bytes)
CREATE src/app/create/create.page.html (125 bytes)
CREATE src/app/create/create.page.spec.ts (661 bytes)
CREATE src/app/create/create.page.ts (256 bytes)
CREATE src/app/create/create.page.scss (0 bytes)
UPDATE src/app/app-routing.module.ts (611 bytes)
[OK] Generated page!

C:\Users\joanm\ionic\ionic-sqlite-crud>
```

## Actualizando las rutas

A continuación, diríjase al archivo de enrutamiento de la aplicación (app routing file); agregue la propiedad `id` en la ruta de edición y configure considerablemente la creación como ruta predeterminada.

**Update `app-routing.module.ts` file:**



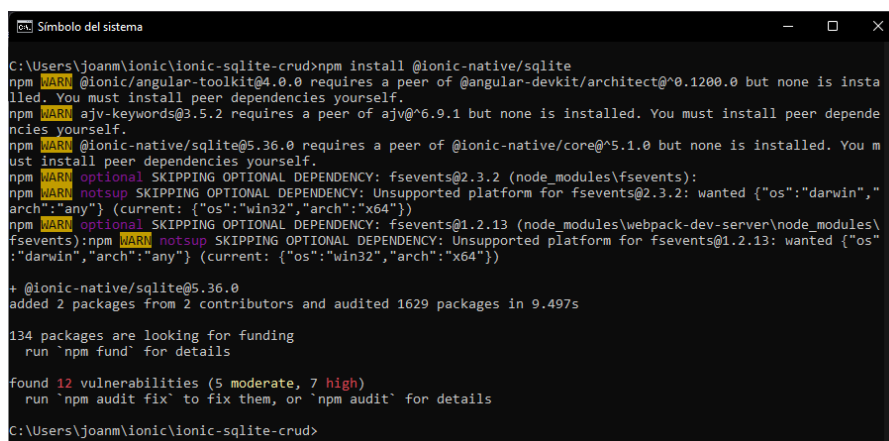
```
1 import { NgModule } from '@angular/core';
2 import { PreloadAllModules, RouterModule, Routes } from '@angular/router';
3
4 const routes: Routes = [
5   {
6     path: 'home',
7     loadChildren: () => import('./home/home.module').then( m => m.HomePageModule)
8   },
9   {
10    path: '',
11    redirectTo: 'home',
12    pathMatch: 'full'
13  },
14  {
15    path: 'create',
16    loadChildren: () => import('./create/create.module').then( m => m.CreatePageModule)
17  },
18  {
19    path: 'edit',
20    loadChildren: () => import('./edit/edit.module').then( m => m.EditPageModule)
21  },
22 ];
23
24 @NgModule({
25   imports: [
26     RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
27   ],
28   exports: [RouterModule]
29 })
30 export class AppRoutingModule { }
31
```

## Instalar SQLite Storage Plugin

Este es el paso más esencial; según las instrucciones, necesitamos instalar los complementos de núcleo nativo SQLite más Ionic.

Abra la terminal, escriba comando y luego ejecute.

```
npm install @ionic-native/sqlite
ionic cordova plugin add cordova-sqlite-storage
npm i @ionic-native/core
```



```
C:\Users\joanm\ionic\ionic-sqlite-crud>npm install @ionic-native/sqlite
npm WARN @ionic/angular-toolkit@4.0.0 requires a peer of @angular-devkit/architect@^0.1200.0 but none is installed. You must install peer dependencies yourself.
npm WARN ajv-keywords@3.5.2 requires a peer of ajv@^6.9.1 but none is installed. You must install peer dependencies yourself.
npm WARN @ionic-native/sqlite@5.36.0 requires a peer of @ionic-native/core@^5.1.0 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\webpack-dev-server\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
+ @ionic-native/sqlite@5.36.0
added 2 packages from 2 contributors and audited 1629 packages in 9.497s

134 packages are looking for funding
  run `npm fund` for details

found 12 vulnerabilities (5 moderate, 7 high)
  run `npm audit fix` to fix them, or `npm audit` for details
C:\Users\joanm\ionic\ionic-sqlite-crud>
```

```
Símbolo del sistema

C:\Users\joanm\ionic\ionic-sqlite-crud>npm i @ionic-native/core
npm WARN @ionic/angular-toolkit@4.0.0 requires a peer of @angular-devkit/architect@^0.1200.0 but none is installed. You must install peer dependencies yourself.
npm WARN ajv-keywords@3.5.2 requires a peer of ajv@^6.9.1 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\webpack-dev-server\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ @ionic-native/core@5.36.0
added 1 package from 1 contributor and audited 1630 packages in 6.207s

134 packages are looking for funding
  run `npm fund` for details

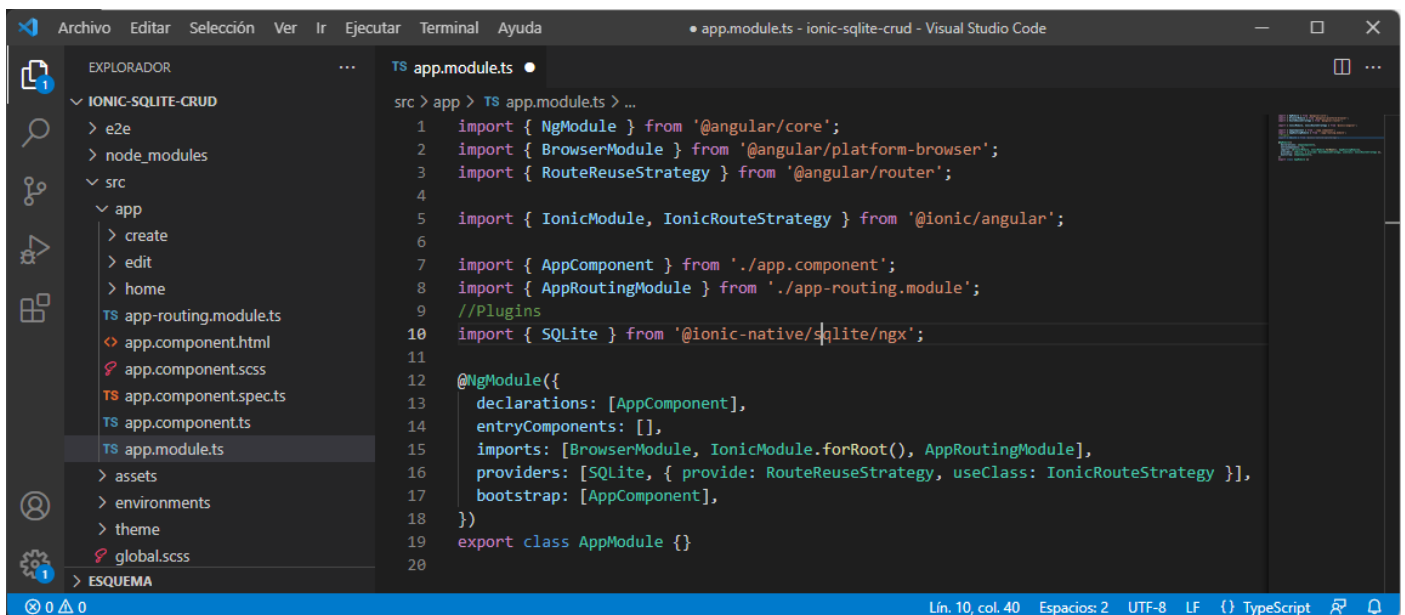
found 12 vulnerabilities (5 moderate, 7 high)
  run `npm audit fix` to fix them, or `npm audit` for details

C:\Users\joanm\ionic\ionic-sqlite-crud>
```

## Actualizar App Module Class

Además, necesitamos importar el complemento SQLite a la clase del módulo de la aplicación, y nos da acceso a sus métodos y funciones para usar en toda la aplicación.

Actualizar app.module.ts :



```
src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { RouteReuseStrategy } from '@angular/router';
4
5  import { IonicModule, IonicRouteStrategy } from '@ionic/angular';
6
7  import { AppComponent } from './app.component';
8  import { AppRoutingModule } from './app-routing.module';
9  //Plugins
10 import { SQLite } from '@ionic-native/sqlite/ngx';
11
12 @NgModule({
13   declarations: [AppComponent],
14   entryComponents: [],
15   imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule],
16   providers: [SQLite, { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }],
17   bootstrap: [AppComponent],
18 })
19 export class AppModule {}
20
```

## Crear SQLite CRUD Service

El servicio angular es mejor para el paradigma de capacidad de administración y reutilización del código. En consecuencia, genere el servicio para la aplicación móvil crud sin conexión iónica con el siguiente comando.

```
ionic generate service crud
C:\Users\joanm\ionic\ionic-sqlite-crud>ionic generate service crud
> ng.cmd generate service crud --project=app
CREATE src/app/crud.service.spec.ts (347 bytes)
CREATE src/app/crud.service.ts (133 bytes)
[OK] Generated service!
```

En el siguiente paso, tenemos que crear y agregar, leer, actualizar y eliminar métodos. Lo más importante es que tenemos que invocar la conexión de la base de datos SQLite en la aplicación Ionic desde el archivo de servicio angular ionic.

Sqlite ofrece el método de creación, que toma el nombre y la ubicación de la base de datos e inicializa la conexión de la base de datos SQLite.

El comando anterior manifestó el archivo **crud.service.ts**, ábralo y actualice el código sugerido dentro del archivo.

```
import { Injectable } from '@angular/core';
import { Platform } from '@ionic/angular';
import { SQLite, SQLiteObject } from '@ionic-native/sqlite/ngx';

@Injectable({
  providedIn: 'root'
})

export class CrudService {

  private dbInstance: SQLiteObject;
  readonly db_name: string = "remotestack.db";
  readonly db_table: string = "userTable";
  USERS: Array <any> ;

  constructor(
    private platform: Platform,
    private sqlite: SQLite
  ) {
    this.databaseConn();
  }

  // Create SQLite database
  databaseConn() {
    this.platform.ready().then(() => {
      this.sqlite.create({
        name: this.db_name,
        location: 'default'
      }).then((sqlite: SQLiteObject) => {
        this.dbInstance = sqlite;
      });
    });
  }
}
```

```

        sqlite.executeSql(`
            CREATE TABLE IF NOT EXISTS ${this.db_table} (
                user_id INTEGER PRIMARY KEY,
                name varchar(255),
                email varchar(255)
            ), [])
        .then((res) => {
            // alert(JSON.stringify(res));
        })
        .catch((error) => alert(JSON.stringify(error)));
    })
    .catch((error) => alert(JSON.stringify(error)));
});
}

// Crud
public addItem(n, e) {
    // validation
    if (!n.length || !e.length) {
        alert('Provide both email & name');
        return;
    }
    this.dbInstance.executeSql(`
    INSERT INTO ${this.db_table} (name, email) VALUES ('${n}', '${e}')`, [])
    .then(() => {
        alert("Success");
        this.getAllUsers();
    }, (e) => {
        alert(JSON.stringify(e.err));
    });
}

getAllUsers() {
    return this.dbInstance.executeSql(`SELECT * FROM ${this.db_table}`, []).then((res) => {
        this.USERS = [];
        if (res.rows.length > 0) {
            for (var i = 0; i < res.rows.length; i++) {
                this.USERS.push(res.rows.item(i));
            }
            return this.USERS;
        }
    }, (e) => {
        alert(JSON.stringify(e));
    });
}

// Get user
getUser(id): Promise<any> {

```

```

        return this.dbInstance.executeSql(`SELECT * FROM ${this.db_table} WHERE user_id = ?`,
[id])
        .then((res) => {
            return {
                user_id: res.rows.item(0).user_id,
                name: res.rows.item(0).name,
                email: res.rows.item(0).email
            }
        });
    }

    // Update
    updateUser(id, name, email) {
        let data = [name, email];
        return this.dbInstance.executeSql(`UPDATE ${this.db_table} SET name = ?, email = ? WHERE
user_id = ${id}`, data)
    }

    // Delete
    deleteUser(user) {
        this.dbInstance.executeSql(`
DELETE FROM ${this.db_table} WHERE user_id = ${user}`, [])
        .then(() => {
            alert("User deleted!");
            this.getAllUsers();
        })
        .catch(e => {
            alert(JSON.stringify(e))
        });
    }
}

```

Tenemos que realizar múltiples tareas utilizando el archivo de servicio, primero la plataforma de importación, los módulos SQLite y SQLiteObject. Estos son esenciales para crear y configurar la conexión de la base de datos SQLite cuando la plataforma está lista.

Para manejar las operaciones CRUD estamos usando SQLite, que permite escribir consultas MySQL con el método `executeSql()`. Además, creamos una conexión de base de datos SQLite, creamos una base de datos, creamos e insertamos una tabla en la base de datos. Además, `addItem()`, `getAllUsers()`, `getUser()`, `updateUser()` y `deleteUser()` para manejar eventos CRUD.

## Implementando crear y eliminar: Create and Delete

En este paso, aprenderemos a implementar cómo crear y mostrar la función de lista en una página ionic usando SQLite.

En este paso, agregaremos un formulario básico para permitir que los usuarios ingresen datos como el nombre y el correo electrónico usando ngModel. Use CrudService importando y agregando en el método constructor, métodos de acceso creando usuarios, obteniendo la lista de usuarios de la base de datos SQLite y eliminando el objeto de usuario de la base de datos SQLite.

Agrega el siguiente código en **create.page.ts**:

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

@Component({
  selector: 'app-create',
  templateUrl: './create.page.html',
  styleUrls: ['./create.page.scss'],
})
export class CreatePage implements OnInit {
  nameVal: string = "";
  emailVal: string = "";
  constructor(
    private crud: CrudService
  ) {
    this.crud.databaseConn();
  }
  ngOnInit() { }

  ionViewDidEnter() {
    this.crud.getAllUsers()
  }

  createUser(){
    this.crud.addItem(this.nameVal, this.emailVal);
  }

  remove(user) {
    this.crud.deleteUser(user);
  }
}
```

Para mostrar la colección de datos, use la directiva ngFor para iterar sobre la colección USERS y mostrarla en el componente de IU del elemento iónico.



Agrega el siguiente código en `create.page.html`:

```
<ion-header>
  <ion-toolbar>
    <ion-title>Ionic SQLite JMGREP Developers Schools</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>

  <!-- Create -->
  <ion-item>
    <ion-label position="floating">Nombre</ion-label>
    <ion-input [(ngModel)]="nameVal"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label position="floating">Email</ion-label>
    <ion-input [(ngModel)]="emailVal"></ion-input>
  </ion-item>

  <ion-button color="success" expand="block" (click)="createUser()">
    Agregar usuario
  </ion-button>

  <!-- Read -->
  <ion-item *ngFor="let user of crud.USERS">
    <ion-label>
      <h2><strong>{{ user.name }}</strong></h2>
      <p>{{ user.email }}</p>
    </ion-label>

    <div class="item-note" item-end>
      <ion-icon color="primary" name="create" style="zoom:1.3" [routerLink]="['/edit/',
user.user_id]"></ion-icon>

      <ion-icon color="danger" name="trash" style="zoom:1.3"
(click)="remove(user.user_id)"></ion-icon>
    </div>
  </ion-item>
</ion-content>
```

### Implementando actualizar y editar:

Por último, implementaremos la función de edición o actualización, así que asegúrese de crear el formulario e insertar los valores utilizando el servicio angular.

Además, usamos la API `ActivatedRoute` para obtener el ID de usuario de la URL, y se pasa al método `getUser()`. Esto está obteniendo el objeto de usuario único de la base de datos SQLite, igualmente usando el método `updateUser()` para actualizar los datos del usuario.

#### Actualiza edit.page.ts:

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

@Component({
  selector: 'app-create',
  templateUrl: './create.page.html',
  styleUrls: ['./create.page.scss'],
})

export class CreatePage implements OnInit {

  nameVal: string = "";
  emailVal: string = "";

  constructor(
    private crud: CrudService
  ) {
    this.crud.databaseConn();
  }

  ngOnInit() { }

  ionViewDidEnter() {
    this.crud.getAllUsers()
  }

  createUser(){
    this.crud.addItem(this.nameVal, this.emailVal);
  }

  remove(user) {
    this.crud.deleteUser(user);
  }

}
```

Abre y actualiza **edit.page.html**:

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button></ion-back-button>
    </ion-buttons>
    <ion-title>Editar</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-item>
    <ion-label position="floating">Nombre</ion-label>
    <ion-input [(ngModel)]="nameVal"></ion-input>
  </ion-item>

  <ion-item>
    <ion-label position="floating">Email</ion-label>
    <ion-input [(ngModel)]="emailVal"></ion-input>
  </ion-item>

  <ion-button color="dark" expand="block" (click)="onUpdate()">
    Actualizar
  </ion-button>
</ion-content>
```

### Probar IONIC Crud App:

En el paso final, debe seguir las instrucciones recomendadas para iniciar la aplicación de ejemplo de ionic crud:

Incluir las plataformas:

#### # iOS

ionic cordova platform add ios

#### # Android

ionic cordova platform add android

### A partir de entonces, cree la compilación ejecutable:

#### # iOS

ionic cordova build ios

#### # Android

ionic cordova build Android

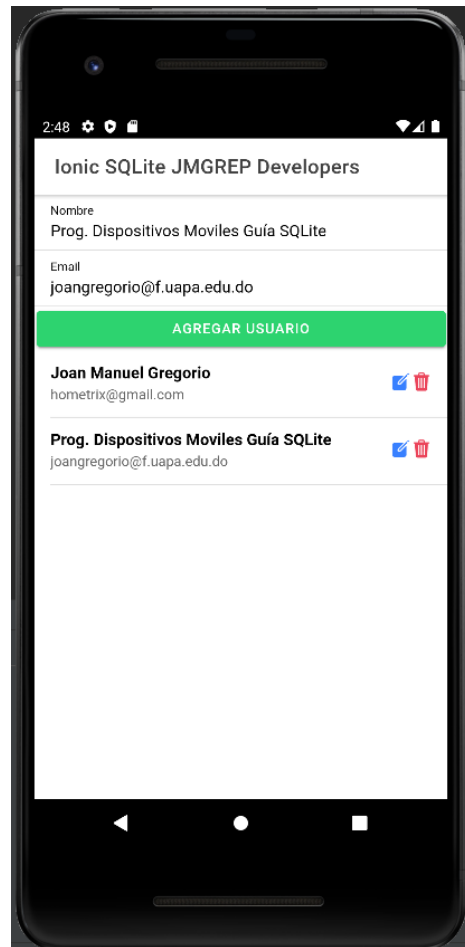
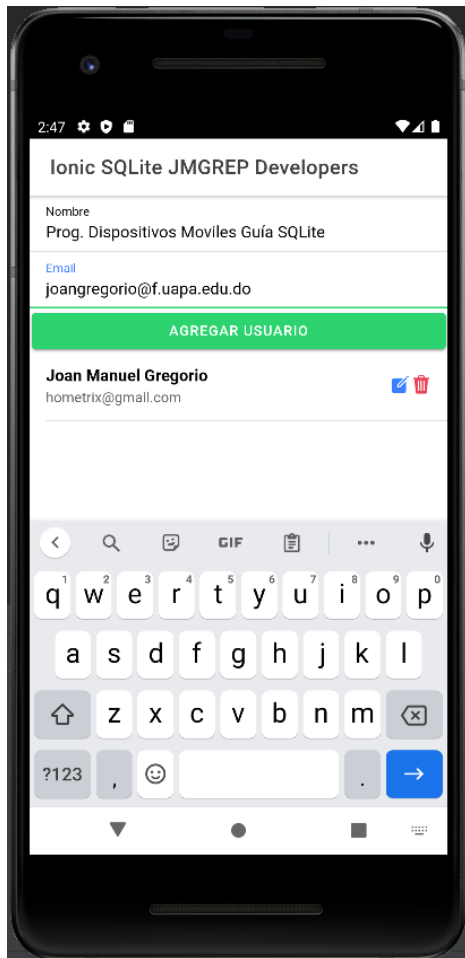
En última instancia, ejecute la aplicación en el dispositivo:

# iOS

ionic cordova run ios -l

# Android

ionic cordova run android -l



**Autor:**

Joan Manuel Gregorio Pérez.  
Ingeniero en software, Magister Gestión de la tecnología

**JMGREP Developers**

**Redes:**

Twitter: @hometrix  
linkedin: hometrix  
<http://joangregorioperez.com>