Proyecto en ionic v7 con SQLite

1 Creamos el proyecto en ionic

ionic start EjConSQLite blank

2 Instalamos los elementos de sqlite para capacitor (dentro de la carpeta del proyecto)

npm install cordova-sqlite-storage

npm install @awesome-cordova-plugins/sqlite

3.- Se importa el plugin y se agrega el plugin para que lo reconozca el proyecto en app.module.ts

```
  import { SQLite } from '@awesome-cordova-plugins/sqlite/ngx';
```

```
providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy },
SQLite],
```

4.- Creamos el servicio y las página

ionic generate service services/dbservice

ionic generate page pages/registrar

ionic generate page pages/modificar

5.- Creamos la clase noticias en la carpeta clases

ionic generate class clases/noticia

le agrego las propiedades id:number, titulo:string, texto:string

6.- Configuramos el servicio para que cree la BD y las tablas con los registros

```
import { SQLite, SQLiteObject } from '@awesome-cordova-plugins/sqlite/ngx';
import { Platform, ToastController } from '@ionic/angular';
import { BehaviorSubject, Observable } from 'rxjs';
import { Noticia } from '../clases/noticia';
```

```
      public database!: SQLiteObject;

  tblNoticias:string = "CREATE TABLE IF NOT EXISTS noticia(id INTEGER
PRIMARY KEY autoincrement, titulo VARCHAR(50) NOT NULL, texto TEXT NOT
NULL);";
```

```
listaNoticias = new BehaviorSubject<Noticia[]>([]);
private isDbReady:
  BehaviorSubject<boolean> = new BehaviorSubject(false);
```

7.- Métodos que van en el Servicio de conexión a la Base de Datos:

```
constructor(private sqlite: SQLite,
  private platform: Platform,
  public toastController: ToastController) {
  this.crearBD();
}

/**
 * Método que crea la BD si no Existe o carga la existente
 */
crearBD() {
  this.platform.ready().then(() => {
    this.sqlite.create({
      name: 'noticias.db',
      location: 'default'
    }).then((db: SQLiteObject) => {
      this.database = db;
      this.presentToast("BD creada");
      //llamo a crear la(s) tabla(s)
      this.crearTablas();
    }).catch(e => this.presentToast(e));
  })
}

/**
   * Método que crea la tabla de la BD si no Existe o carga la existente
   */
async crearTablas() {
  try {
    await this.database.executeSql(this.tblNoticias, []);
    this.presentToast("Tabla creada");
    this.cargarNoticias();
    this.isDbReady.next(true);
  } catch (error) {
    this.presentToast("Error en Crear Tabla: " + error);
  }
}

/**
```

```
    * Método que carga en la listaNoticias TODO el contenido de la tabla
noticia
    */
  cargarNoticias() {
    let items: Noticia[] = [];
    this.database.executeSql('SELECT * FROM noticia', [])
      .then(res => {
        if (res.rows.length > 0) {
          for (let i = 0; i < res.rows.length; i++) {
            items.push({
              id: res.rows.item(i).id,
              titulo: res.rows.item(i).titulo,
              texto: res.rows.item(i).texto
            });
          }
        }
      });
    this.listaNoticias.next(items);
  }

  /**
    * Método que inserta un registro en la tabla noticia
    */
  async addNoticia(titulo: any, texto: any) {
    let data = [titulo, texto];
    await this.database.executeSql('INSERT INTO noticia(titulo,texto)
VALUES(?,?)', data);
    this.cargarNoticias();
  }

  /**
    * Método que actualiza el título y/o el texto filtrando por el id
    */
  async updateNoticia(id: any, titulo: any, texto: any) {
    let data = [titulo, texto, id];
    await this.database.executeSql('UPDATE noticia SET titulo=?, texto=?
WHERE id=?', data);
    this.cargarNoticias();
  }

  /**
    * Método que elimina un registro por id de la tabla noticia
    */
  async deleteNoticia(id: any) {
    await this.database.executeSql('DELETE FROM noticia WHERE id=?', [id]);
```

```
    this.cargarNoticias();
  }

  /**
     * Método que verifica la suscripción del Observable
     */
  dbState() {
    return this.isDbReady.asObservable();
  }

  /**
     * Método que se ejecuta cada vez que se hace un cambio en la tabla de
la BD
     */
  fetchNoticias(): Observable<Noticia[]> {
    return this.listaNoticias.asObservable();
  }

  async presentToast(mensaje: string) {
    const toast = await this.toastController.create({
      message: mensaje,
      duration: 3000
    });
    toast.present();
  }
```

8.- Llamamos al servicio en el home

Modificamos html y ts del home

Modificamos html y ts de las otras páginas (registrar y modificar)

9.- Agregamos la plataforma Android al proyecto (Se recomienda usar ante cada cambio en el proyecto antes de iniciarlo)

ionic build

ionic capacitor add Android /*este se puede hacer solo 1 vez y repetir si detectas que no actualiza tus cambios*/

ionic cap sync

10.- Abrimos el proyecto en Android studio

npx cap open android