PROGRAMACIÓN DE APLICACIONES MÓVILES

PGY4121

Profesor:

DuocUC®

ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

Actividad N°2.2: Persistencia

Persistencia

Objetivos

Lo que se espera que aprendas en esta Actividad es:

- En esta actividad ustedes van a conocer el uso de la memoria interna Store y persistir los datos

¿Persistencia?

PGY4121 Persistencia

- La persistencia nos permite almacenar los datos generados o utilizados por la aplicación mediante y para el funcionamiento de éste

PGY4121 SQLite

- Es un sistema de almacenamiento de datos multiplataforma que funciona en IOS y Android. Desarrollado por SQLite, un motor de base de datos SQL para crear potentes aplicaciones basadas en datos completamente JavaScript

PGY4121 Ionic Storage

- Ionic Storage es una alternativa gratuita y Opensource para desarrolladores.
- Almacena los datos por pares, clave/valor y Objetos JSON

PGY4121 Ionic Storage

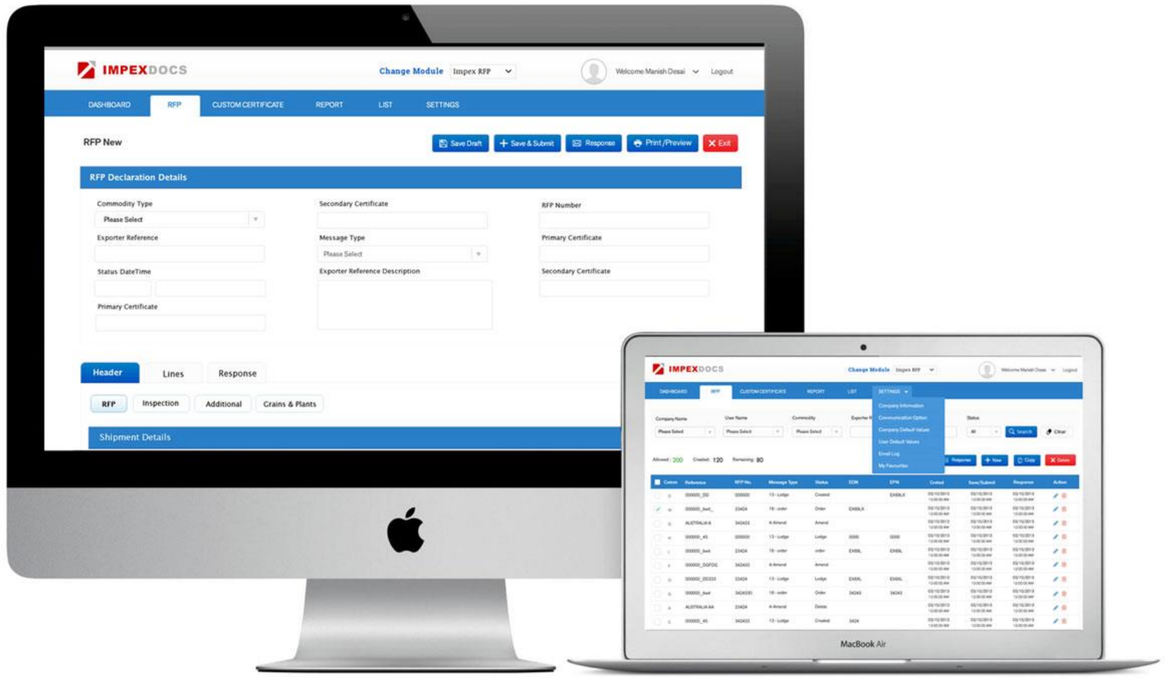
- Utiliza una variedad de motores de almacenamiento por debajo
- El motor de almacenamiento es elegido por el desarrollador dependiendo de la plataforma

PGY4121 Motor de Almacenamiento

- Ionic Storage utiliza SQLite para plataformas nativas

PGY4121 Motor de Almacenamiento

- Ionic Storage utiliza IndexedDB, WebSQL, localStorage para plataformas web en ese orden



Uso de Storage

PGY4121 Obtener Información

» para obtener la información usamos la función get()

```
// Or to get a key/value pair
storage.get('age').then((val) => {
  console.log('Your age is', val);
});
```

Busca la clave “age” .get(‘age’) entonces retorna el valor .then(val)=>{} y lo muestra en la consola console.log(‘Your age is’,val)

PGY4121 Setear Información

» para setear la información usamos la función set()

```
// set a key/value
storage.set('name', 'Max');
```

Para la clave “name” setea el valor “max” .set(‘name’,’max’)

Primer parámetro clave, segundo parámetro valor

PGY4121 Storage con JSON

» para obtener la información usamos la función get()

```
constructor(private storage: Storage) {
    var obj = {
      name:"ian",
      age:24
    }
    storage.set('obj', obj);
    storage.get('obj').then((val) => {
      console.log(val);
      console.log(val.name)
    });
  }
```

```
▶ {name: "ian", age: 24}
ian
```

PGY4121 | Funciones Storage

- clear(): Limpia las entradas de clave/valor
- get(): Obtiene el valor asociado a una clave
- keys(): Retorna todas las claves del store
- length(): Retorna el número de claves almacenadas en el store
- ready(): Retorna cuando el store esta listo
- remove(): Elimina un valor asociado a una clave
- set(): Almacena una clave y su valor

DuocUC®

Uso de SQLite

DuocUC®

PGY4121 Uso de SQLite

- Lo primero que se debe hacer es instalar el plugin, las instrucciones están en la documentación oficial, deben considerar que desde Ionic v6 se utiliza por defecto Capacitor y por lo tanto esas son las instrucciones válidas de instalación (para mayor seguridad se recomienda verificar la versión de Ionic que se está utilizando y así acceder a la documentación adecuada para dicha versión): https://ionicframework.com/docs/native/sqlite

```
npm install cordova-sqlite-storage
npm install @awesome-cordova-plugins/sqlite
```

PGY4121 Uso de SQLite

```
npm install cordova-sqlite-storage
npm install @awesome-cordova-plugins/sqlite
```

» app.module.ts

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

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule],
  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy },SQLite],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

PGY4121 Uso de SQLite

**Se recomienda aislar en un servicio todo lo relacionado con la base de datos

» Creación de base de datos:

```
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));
  })
}
```

PGY4121 Uso de SQLite

» Declaración y Creación de tabla:

```
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);";
```

```
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);
  }
}
```

PGY4121 Uso de SQLite

» Ejecutar SQL Insert:

```
addNoticia(titulo,texto){
  let data=[titulo,texto];
  return this.database.executeSql('INSERT INTO noticia(titulo,texto) VALUES(?,?)',data)
  .then(()=>{
    this.cargarNoticias();
  });
}
```

PGY4121 Uso de SQLite

» Ejecutar SQL Update y Delete Table:

```
updateNoticia(id,titulo,texto){
  let data=[titulo,texto,id];
  return this.database.executeSql('UPDATE noticia SET titulo=?, texto=? WHERE id=?',data)
  .then(()=>{
    this.cargarNoticias();
  });
}

deleteNoticia(id){
  return this.database.executeSql('DELETE FROM noticia WHERE id=?',[id])
  .then(()=>{
    this.cargarNoticias();
  });
}
```

PGY4121 Uso de SQLite

- Ejecutar SQL Select Table:

```
cargarNoticias() {
  return this.database.executeSql('SELECT * FROM noticia',[])
  .then(res=>{
    let items:Noticia[]=[];
    if(res.rows.length>0){
      for (var 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);
  });
}
```

Documentación Oficial

Recuerda!

Consulta siempre la documentación oficial al momento de implementar persistencia en nuestra aplicación.

DuocUC®

PGY4121 Documentación Oficial

» https://ionicframework.com/docs/native/sqlite

» https://ionicframework.com/docs/angular/storage

** Importante: al acceder a los github desde la documentación oficial, no olvides comprobar la versión de ionic con que fue desarrollado ese código

Resumen

PGY4121 Resumen

- En resumen lo que hemos vistos son las opciones que tenemos para mantener persistente los datos en nuestra aplicación

- No dejen de consultar la documentación oficial para resolver los problemas que se presenten en el desarrollo.