

المحاضرة السادسة:

التعامل مع قواعد البيانات / SQLite DataBase



مدرسوالمقرر:

م.محمودالجلخ.

م.رهامالعر.

م.هلابريمان.

التطبيق العملي.....3.

التطبيق العملي /قواعد البيانات:خطأ! الإشارة المرجعية غير معرّفة.

التعامل مع قواعد البيانات

قواعد البيانات التي يمكن لنظام الأندرويد التعامل معها هي SQLite، بحيث يمكن إنشاء قاعدة بيانات، وإنشاء جداول بداخلها، من ثم التعامل مع هذه الجداول من خلال العمليات المعروفة على قواعد البيانات (delete، update، select، insert).

سنتعامل مع قواعد البيانات SQLite ضمن التطبيق التالي:

سيتضمن التطبيق التعامل مع قاعدة البيانات (سنسميها user_info مثلاً)، وبداخلها جدول لتسجيل المستخدمين ضمنه (سنسميه reg_info مثلاً)، ويتكون الجدول من عمودين: الأول لاسم المستخدم، والثاني لكلمة المرور.

لتسهيل التعامل مع هذه الأسماء الأربعة، سنقوم بتخزينها ضمن صف على شكل ثوابت، يمكن الوصول إليها من خلال اسم الصف كونها معرفة، static والصف سنسميه TableInfo. بالشكل الآتي:

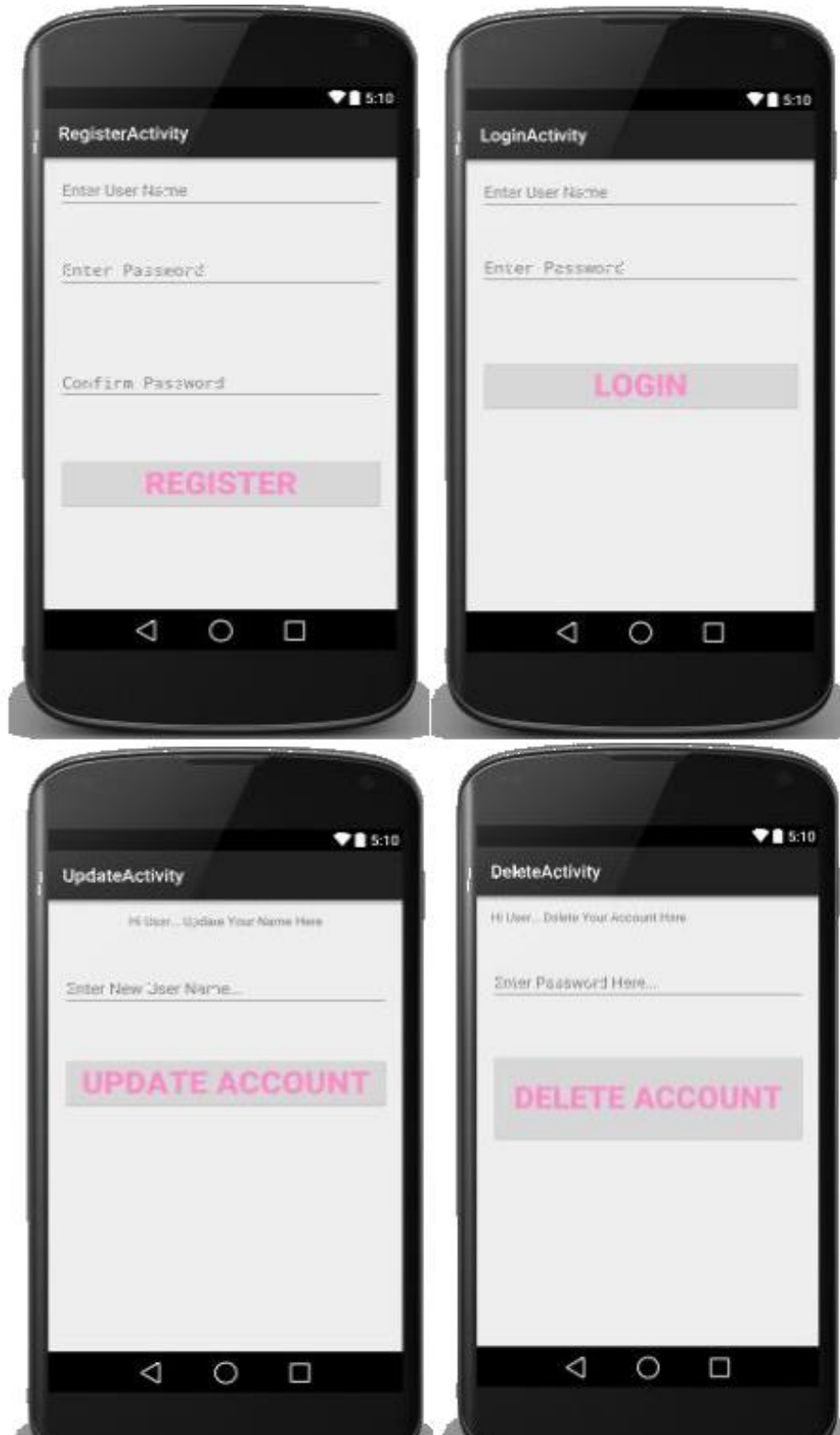
```
public static abstract class TableInfo implements BaseColumns
{
    public static final String USER_NAME="user_name";
    public static final String USER_PASS="user_pass";
    public static final String DATABASE_NAME="user_info";
    public static final String TABLE_NAME="reg_info";
}
```

التطبيق العملي/قواعد البيانات

واجهة التطبيق الأساسية ستكون بالشكل الآتي (تتضمن أربع أزرار: زر لعملية التسجيل / عملية insert ضمن الجدول، زر لعملية تسجيل الدخول / عملية select من الجدول، زر لعملية تعديل المعلومات / عملية update على أحد سجلات الجدول، زر لعملية حذف حساب المستخدم / عملية delete سجل من الجدول):



وعند اختيار كل زر من الأزرار الأربعة، سيتم الانتقال إلى النشاط المناسب لتنفيذ المطلوب، لذلك سيكون لدينا الأنشطة الأربعة الآتية (لكل واحدة من العمليات الأربع):



نترك برمجة الواجهات كما أخذنا في الجلسات السابقة.

نأتي على برمجة النشاطات: نبدأ بالنشاط الأساسي: MainActivity.java

عند النقر على زر Register سيتم الانتقال إلى النشاط الخاص بعملية التسجيل RegisterActivity

ويتم الانتقال باستخدام Intent (كما تعلمنا مسبقاً)، ولا حاجة لتمرير أي قيمة مع Intent.

وعند النقر على زر Login يتم الانتقال أيضاً إلى النشاط الخاص بعملية تسجيل الدخول LoginActivity

أيضاً تم ذلك باستخدام Intent.

وعند النقر على زر update لا يتم الانتقال مباشرة إلى النشاط الخاص بعملية تعديل المعلومات

UpdateActivity، وإنما يجب أن نتأكد من وجود المستخدم مسبقاً ضمن قاعدة البيانات، لذلك سننتقل بداية

إلى نشاط LoginActivity وبعد التأكد من وجود المستخدم، ننتقل إلى واجهة التعديل.

وعند النقر على زر delete لا يتم الانتقال مباشرة إلى النشاط الخاص بعملية حذف المعلومات من القاعدة

DeleteActivity، وإنما يجب أن نتأكد من وجود المستخدم مسبقاً ضمن قاعدة البيانات، لذلك سننتقل بداية

إلى نشاط LoginActivity وبعد التأكد من وجود المستخدم، ننتقل إلى واجهة الحذف.

نقف الآن أمام مشكلة أن الأزرار الثلاثة: delete، update، login كلها ستنتقل إلى النشاط الخاص

بعملية تسجيل الدخول LoginActivity، بالتالي، عندما نصل إلى LoginActivity لا نعرف من أي زر

وصلنا: أي هل نقوم بتنفيذ تسجيل الدخول فقط (إذا كان المستخدم موجوداً، نظهر له رسالة ترحيب، أما إن

لم يكن موجوداً، نظهر له رسالة بعدم وجوده ضمن القاعدة)، أم نقوم بتنفيذ تسجيل الدخول، وننتقل بعدها

لنتم عملية تعديل المعلومات، أم نقوم بتنفيذ تسجيل الدخول، وننتقل بعدها لنتم عملية حذف الحساب من

القاعدة.

لحل هذه المشكلة، سنمرر مع Intent أثناء الانتقال من الأزرار الثلاثة: delete، update، login قيمة

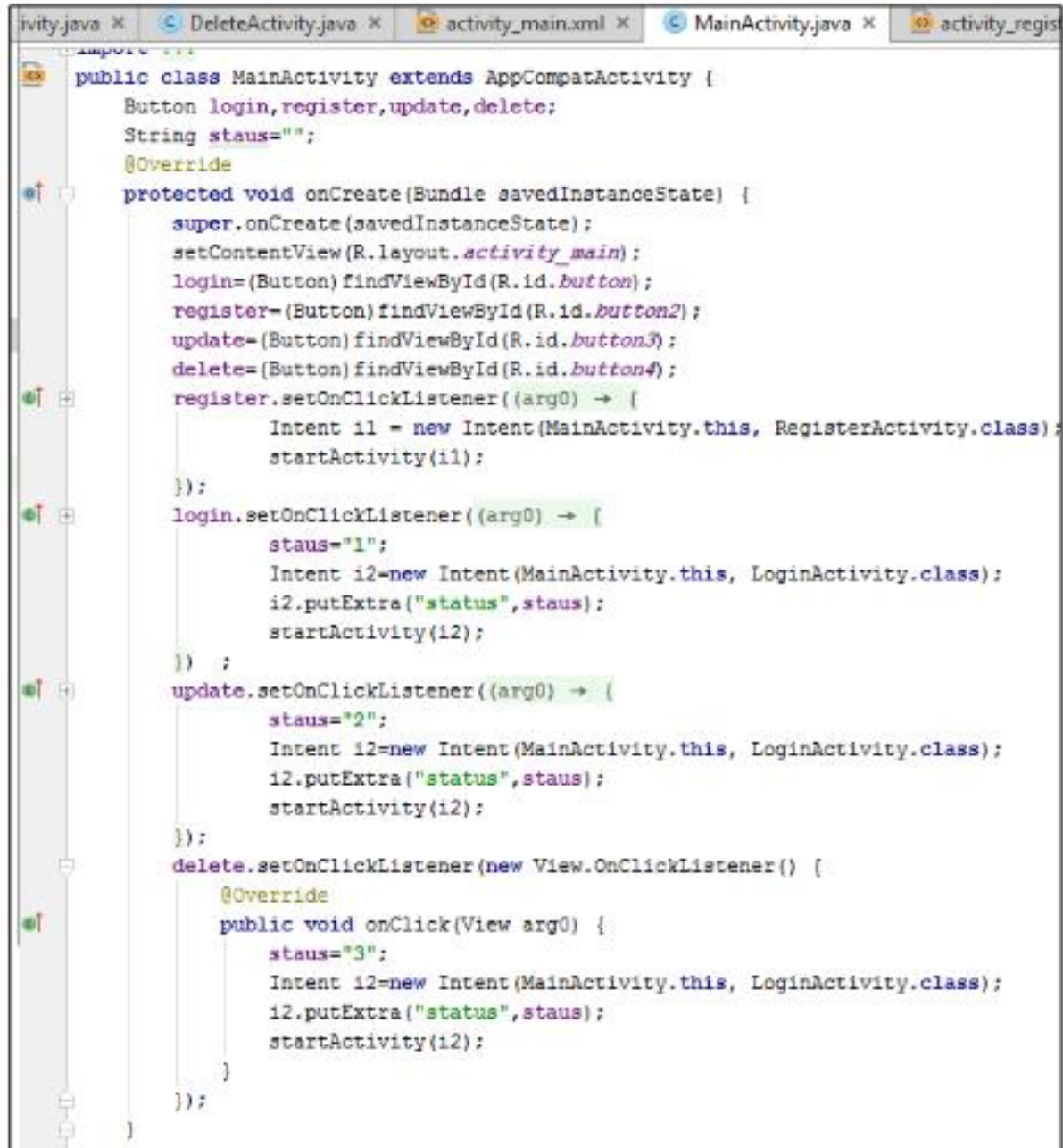
نسميها status مثلاً، وتأخذ عند كل زر قيمة مختلفة:

سنعطيها قيمة=1 ضمن الزر login

سنعطيها قيمة=2 ضمن الزر update

سنعطيها قيمة=3 ضمن الزر delete

بالتالي سيكون الكود الكامل لنشاط MainActivity.java والذي يحوي برمجة الأزرار الأربعة بالشكل الآتي:



```

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.content.Intent;

public class MainActivity extends AppCompatActivity {
    Button login, register, update, delete;
    String staus="";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        login=(Button) findViewById(R.id.button);
        register=(Button) findViewById(R.id.button2);
        update=(Button) findViewById(R.id.button3);
        delete=(Button) findViewById(R.id.button4);
        register.setOnClickListener((arg0) -> {
            Intent i1 = new Intent(MainActivity.this, RegisterActivity.class);
            startActivity(i1);
        });
        login.setOnClickListener((arg0) -> {
            staus="1";
            Intent i2=new Intent(MainActivity.this, LoginActivity.class);
            i2.putExtra("status",staus);
            startActivity(i2);
        });
        update.setOnClickListener((arg0) -> {
            staus="2";
            Intent i2=new Intent(MainActivity.this, LoginActivity.class);
            i2.putExtra("status",staus);
            startActivity(i2);
        });
        delete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                staus="3";
                Intent i2=new Intent(MainActivity.this, LoginActivity.class);
                i2.putExtra("status",staus);
                startActivity(i2);
            }
        });
    }
}

```

نبدأ الآن ببرمجة النشاطات الأربعة:

بداية مع عملية التسجيل:

قبل تنفيذ عملية التسجيل، نحتاج لإنشاء قاعدة البيانات، من ثم إنشاء الجدول ضمنها

للتعامل مع عمليات قاعدة البيانات، سنعرف صف جديد سنضع بداخله كل ما يتعلق بالتعامل مع قاعدة البيانات، من إنشاء القاعدة، إلى إنشاء الجدول، إلى الإدخال ضمن الجدول، سنسميه مثلاً DataBaseOperations، سنقوم بتوريث الصف من صف أب هو الصف SQLiteOpenHelper وهو صف يساعد بالتعامل مع قواعد البيانات (إنشاء وإدارة القاعدة). عندما نكتب extends SQLiteOpenHelper سيطهر لدينا مشكلة أن هناك طرق يجب عادة تحقيقها، نقوم بتحقيقها، وهي الطريقة onCreate(...) والطريقة onUpgrade(..) هذه لن نتعامل معها. بعد القيام بتحقيق الطرق المطلوبة، تظهر مشكلة أخرى، وهي أنه لا يوجد باني للصف، فيتم اقتراح إنشاء باني يقوم باستدعاء باني الأب (من خلال super). نقوم بإنشاء الباني الذي يستدعي باني الأب، هذا الباني سيقوم فعلياً بإنشاء قاعدة البيانات، عند اختياره سيكون له أربع وسطاء، نترك فقط أول وسيط Context ونحذف الوسطاء الباقية، أما super، فنعطيه الوسطاء التالية:

غرض Context الممرر كبارامتر

الوسيط الثاني هو اسم قاعدة البيانات التي نريد إنشاءها، كما نتذكر فقد خزنا الاسم ك ثابت ضمن الصف TableInfo، لذلك سيكون الوسيط الثاني هو TableInfo.DATABASE_NAME

الوسيط الثالث يسمى cursorFactory، نضع فيه null

الوسيط الأخير يثل إصدار قاعدة البيانات (رقم int)، سنعرف عنه ك ثابت

وهنا عند استدعاء super سيري هل القاعدة منشأة مسبقاً أو لا، إن كانت غير موجودة، يقوم بإنشائها، وإن كانت موجودة، يقوم بفتحها ليتم التعامل معها.

بعد إنشاء القاعدة، نحتاج لإنشاء الجدول حتى نتمكن من التعامل معه، يتم عادة إنشاء الجداول ضمن الطريقة onCreate.

ضمن onCreate يمكن تنفيذ أي تعليمة SQL ما عدا delete، select، update، insert، مثل تعليمات إنشاء الجداول، تعديلها، حذفها، إنشاء الفهارس. ..

يتم إنشاء الجدول باستدعاء الطريقة `execSQL` على الغرض الممر كوسيط للطريقة `onCreate` ، وهو غرض من الصف `SQLiteDatabase` اسمه `db`. وتأخذ الطريقة `execSQL` كوسيط لها السلسلة التي تمثل تعليمة SQL التي نريد تنفيذها.

سنكتب تعليمة SQL لإنشاء الجدول ضمن السلسلة `CREATE_QUERY` (نتذكر طريقة كتابة تعليمة إنشاء جدول بالشكل الآتي:

Create table Table_name(column1_name column1_datatype,column2_name column2_datatype);

ننتبه للفراغات بين كلمة `table` المحجوزة واسم الجدول، وبين اسم العمود ونمطه.

فيكون الكود المطلوب لإنشاء القاعدة والجدول ضمن الشكل الآتي:

```

package com.example.lenovo.sqlite;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

/**
 * Created by Lenovo on 21/04/2016.
 */

public class DataBaseOperations extends SQLiteOpenHelper {
    public static final int database_version=1;
    public String CREATE_QUERY="CREATE TABLE "+ TableData.TableInfo.TABLE_NAME+ " (" + TableData.TableInfo.USER_NAME+ " TEXT,"+ TableData.TableInfo.USER_PASS+ " TEXT)";
    public DataBaseOperations(Context context) {
        super(context, TableData.TableInfo.DATABASE_NAME, null, database_version);
        Log.d("db operations", "DB Created...");
    }

    @Override
    public void onCreate(SQLiteDatabase sdb) {
        sdb.execSQL(CREATE_QUERY);
        Log.d("db operations", "Table Created...!!");
    }
}

```

الآن أصبحنا قادرين على التعامل مع الجدول، أول عملية سنقوم بتحقيقها، هي عملية الإدخال:

ننشئ طريقة جديدة ضمن الصف `DataBaseOperations` لهذا الغرض، نسميها `putInformation`

سنعطي لهذه الطريقة 3 وسطاء:

غرض من الصف DataBaseOperatins الذي يسهل لنا التعامل مع قواعد البيانات، اسم المستخدم الذي سنقوم بتسجيله، وكلمة المرور كذلك.

لن نعيد هذه الطريقة أي قيمة

أول ما نقوم به عند تنفيذ أي عملية على قواعد البيانات، هو تحديد كيفية التعامل مع قاعدة البيانات (قراءة منها، أو كتابة ضمنها).

يمكن الحصول على غرض يمثل قاعدة البيانات التي نريد التعامل معها باستدعاء الطريقة

`getWritableDatabase()` إذا كنا سنستخدم القاعدة لعملية كتابة أو تعديل ضمنها. والطريقة

`getReadableDatabase()` إذا كنا سنستخدم القاعدة لعملية قراءة بيانات منها فقط.

في حالتنا، سنقوم بعملية إدخال، لذلك نختار قاعدة للكتابة ضمنها.

كل من الطريقتين السابقتين تعيد غرض يمثل قاعدة البيانات `SQLiteDatabase`

لتنفيذ عملية الكتابة/الحشر ضمن القاعدة، لدينا صف مساعد اسمه `ContentValues` يساعد في عملية

الإدخال، تمثيله الفعلي عبارة عن `Hash Map` (key لها هو اسم العمود ضمن الجدول، وvalue هي

القيمة التي سيتم إدخالها ضمن هذا العمود المحدد).

بعد إنشاء غرض منه، يتم إدخال العناصر ضمنه باستدعاء الطريقة `put` (نحدد لها key وvalue).

أصبح كل شيء جاهزاً لتنفيذ الإدخال. نستدعي الآن الطريقة `insert` على غرض `SQLiteDatabase`

الذي حصلنا عليه بدايةً، تأخذ هذه الطريقة الوسطاء الآتية:

اسم الجدول الذي سنحشر ضمنه

الوسيط الثاني نكتب ضمنه إذا كان هناك أعمدة تقبل القيمة `null` ضمنها، في تطبيقنا لا يوجد سوا عمودين،

ولا يقبل أي منهما القيمة `null`، لذلك ستكون قيمة هذا الوسيط `null`.

الوسيط الثالث هو غرض `ContentVlaues` الذي قمنا بتهيئته.

بالتالي، يكون كود الطريقة `putInformation` كاملاً بالشكل الآتي:

```

public void putInformation(DataBaseOperations dop,String name,String pass)
{
    SQLiteDatabase SQ=dop.getWritableDatabase();
    ContentValues cv=new ContentValues();
    cv.put(TableData.TableInfo.USER_NAME,name);
    cv.put(TableData.TableInfo.USER_PASS, pass);
    long k= SQ.insert(TableData.TableInfo.TABLE_NAME,null,cv);
    Log.d("db operations","One Row Inserted...");
}

```

نعود الآن إلى النشاط المسؤول عن التسجيل RegisterActivity.

بعد تعريف العناصر الموجودة على الواجهة. عملنا سيكون عند النقر على الزر، لذلك سنكتب ضمن Listener الخاصة بالزر.

أول ما نقوم به هو إحصار القيم التي أدخلها المستخدم ضمن حقول النص (اسم المستخدم، كلمة المرور، تأكيد كلمة المرور)، ونخزنها ضمن متحولات من النمط String.

سنقوم بدايةً بالتأكد من تطابق كلمتي المرور (في حال لم تكونا متطابقتين، نعرض رسالة (باستخدام Toast) للمستخدم بذلك، ونفرغ حقلي النص /كلمة المرور وتأكيد كلمة المرور من محتواهما) أما إذا كانت كلمتي المرور متطابقتين، سننفذ عملية الإدخال، نحتاج بدايةً لإنشاء غرض من الصف DataBaseOperations والذي يأخذ كوسيط السياق الحالي الموجودين ضمنه.

نستدعي الطريقة putInformation التي تنفذ الإدخال، ونعرض بعدها رسالة للمستخدم بنجاح عملية التسجيل من خلال Toast.

في النهاية وبعد نجاح التسجيل، لا يجب أن نبقى ضمن نفس الواجهة/ التسجيل، لذلك، نعود إلى واجهة التطبيق الأساسية، وذلك من خلال استدعاء الطريقة finish()، وهي تقوم بإعادتنا إلى واجهة التطبيق الأساسية من أي واجهة موجودين ضمنها.

ويكون الكود الكامل لنشاط التسجيل بالشكل الآتي:

```

RegisterActivity.java × DeleteActivity.java × activity_main.xml × MainActivity.java × activity_register.xml × RegisterActivity.java ×
public class RegisterActivity extends AppCompatActivity {
    EditText un,up,cp;
    String username,password,confirmass;
    Button reg;
    Context c=RegisterActivity.this;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        un=(EditText)findViewById(R.id.editText);
        up=(EditText)findViewById(R.id.editText2);
        cp=(EditText)findViewById(R.id.editText3);
        reg= (Button) findViewById(R.id.button5);
        reg.setOnClickListener((v) -> {
            username=un.getText().toString();
            password=up.getText().toString();
            confirmass=cp.getText().toString();
            if(!(password.equals(confirmass)))
            {
                Toast.makeText(getApplicationContext(),"Password not matching !!!",Toast.LENGTH_LONG).show();
                up.setText("");
                cp.setText("");
            }
            else
            {
                DataBaseOperations db=new DataBaseOperations(c);
                //Toast.makeText(getApplicationContext(),"DataBase Created", Toast.LENGTH_SHORT).show();
                db.putInformation(db,username,password);
                Toast.makeText(getApplicationContext(),"Registration completed", Toast.LENGTH_LONG).show();
                finish();
            }
        });
    }
}

```

عند تشغيل التطبيق يظهر التنفيذ بالشكل الاتي: اول مرة نقوم بإدخال كلمتي مرور غير متطابقتين:

RegisterActivity

Nahla

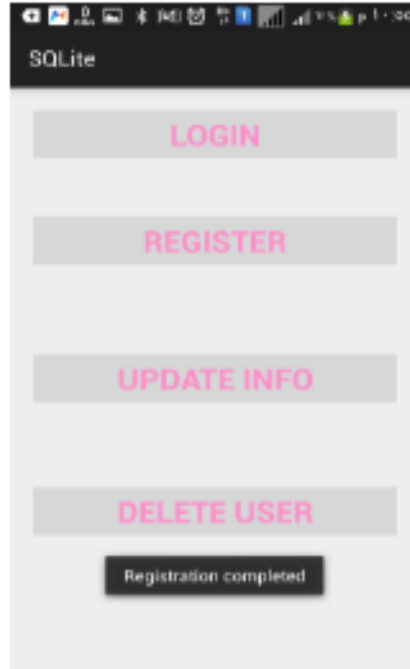
Enter Password

Confirm Password

REGISTER

Password not matching !!!

نقوم الآن بإدخال بيانات صحيحة وكلمات مرور متطابقة، لنحصل على النتيجة الآتية:



تبقى الوظائف المتبقية للجلسة الأخيرة.

انتهت المحاضرة