1. Министерство науки и высшего образования Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**Высшая школа кибербезопасности и защиты информации**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

1. «**Безопасное хранилище данных с помощью SQLite в ОС Android**»
2. по дисциплине «Введение в мобильные технологии и средства связи»
3. Выполнил
4. студент гр. 4851003/70801 Гасанов Э.А.

<*подпись*>

1. Преподаватель
2. ассистент Мясников А.В.

<*подпись*>

1. Санкт-Петербург
2. 2020

# Требования к выполнению

1. Приложение должно иметь возможность сохранять изменять и удалять данные в следующем формате:
   1. Ресурс, для которого сохраняются учетные данные;
   2. Логин (может быть пустым);
   3. Пароль от ресурса;
   4. Заметки (заметки пользователя о данной записи).
2. По умолчанию пароль от ресурса не должен отображаться в открытом виде. У пользователя должна быть возможность скопировать пароль, либо посмотреть его.
3. При запуске приложение должно запрашивать у пользователя код/пароль, который в дальнейшем будет использован в качестве основного ключа шифрования (в дальнейшем будем называть его просто **ключ**). Без правильно введенного кода доступ к данным должен быть невозможен;
4. Приложение должно хранить данные во встроенной в приложение базе данных SQLite, перед записью и получением каждый атрибут записи должен шифроваться;
5. Необходима возможность смены защитного кода/пароля приложения. Для этого необходимо запросить у пользователя новый код, после чего перезаписать все данные, зашифрованные на новом **ключе**, сгенерированном на основе нового кода.
6. Приложение должно обладать возможностью импорта и экспорта сохраненных данных в зашифрованном виде. Формат данных, используемый для экспорта/импорта описать в отчете.

# Ход работы

Для данной лабораторной работы был создан интерфейс для работы с записями, хранящихся в БД SQLite. Но прежде, чем получить доступ к интерфейсу, необходимо ввести код-пароль, с помощью которого эти записи будут шифроваться. В соответствии с вариантом выполнялось AES CFB шифрование.

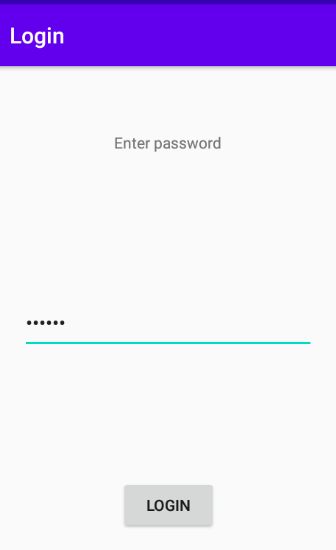


Рисунок 1, входной код-пароль

Экземпляр базы данных хранится в logged-activity, поэтому в других activity будет происходить заполнение текстовых полей. По завершении работы открытой activity, возвращается код, который обрабатывается в logged-activity и считывается переданная информация через SharedPreference для добавления в базу данных. То есть SharedPreference используется для передачи параметров между activity.

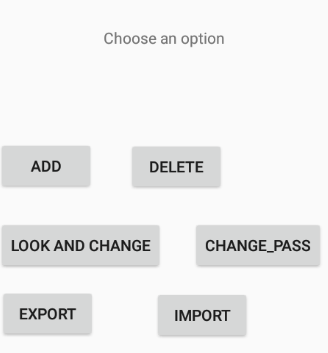


Рисунок 2, доступные действия с заметками в БД

Ниже будут описаны действия, которые позволяет выполнять созданное приложение:

**Добавление**

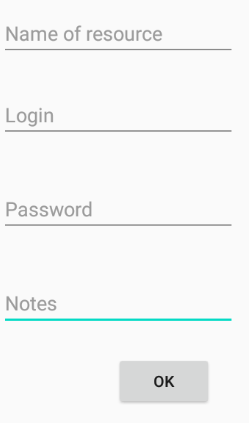


Рисунок 3, добавление заметок

Были найдены кнопки и текстовые строчки, чтобы поставить слушателей, а заполненные поля передавались в вызывающую activity через SharedPreference, где по возвращённому из вызванной activity коду происходит добавление в базу данных. Имена ресурсов не могут повторяться, поэтому при добавлении вызывается геттер, проворящий есть ли в БД запись с таким именем или нет.

**Удаление**

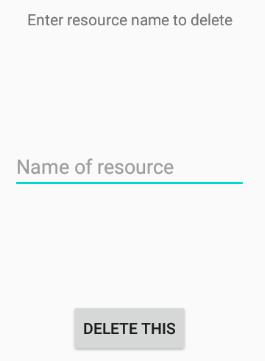


Рисунок 4, интерфейс удаления

Вводится имя ресурса на удаление, проверяется с помощью того же геттера, что такая запись в БД существует. Параметр передается через SharedPreferece. По уникальному имени ресурса находится id, нужный для удаления из базы данных.

**Изменения или просмотр записи**

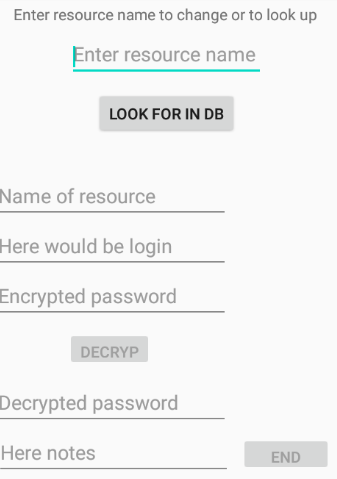


Рисунок 5, интерфейс просмотра или изменения записи в БД

Название ресурса вводится в первом поле, проверяется геттерами по имени и id(так как мы хотим ,возможно, перезаписать конкретную запись), что запись существует. Пока не будет введена запись, другие кнопки не доступны.

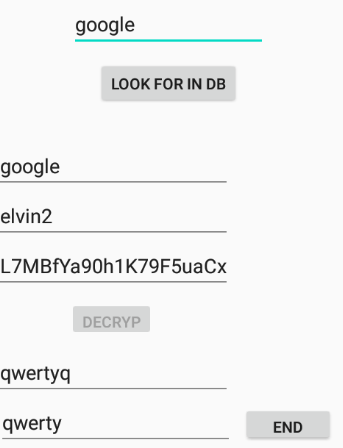


Рисунок 6, кнопка decrypt после расшифрования недоступна для повторного нажатия.

Изменять запись можно прямо в этом интерфейсе, так как по нажатию на кнопку end, все изменения примут силу и запишутся в БД.

**Изменение ключа шифрования**

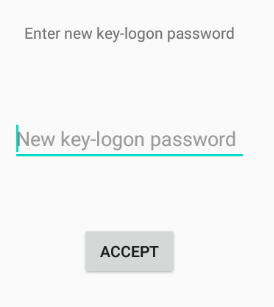


Рисунок 7, новый ключ шифрования

Новый ключ шифрования будет использоваться так же, как новый ключ-пароль для входа в приложение. Новый пароль для передачи между activity шифруется XOR’ом. Чтобы расшифровать БД на старом ключе, нужно геттером взять старый ключ и расшифровать элементы записи, затем установить сеттером новый ключ, и им же зашифровать эти же записи. Передвижение по БД осуществляется циклом с помощью курсора.

**Экспорт**

Экспорт осуществляется на SD-карту устройства в зашифрованном виде, по пути /storage/emulated/o/MyFiles/fileSD

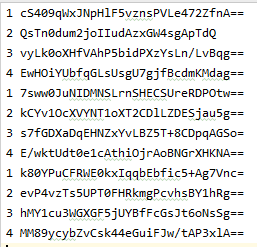


Рисунок 8, экспорт данных в зашифрованном виде

**Импорт**

База данных заполняется зашифрованными данными в по номерам, указанным перед записью, где 1 – это имя ресурса, 2- логин, 3- пароль, 4- заметки.

**Вывод**

Таким образом, мы реализовали интерфейс для работы с заметками с зашифрованном виде с помощью AES CFB. Для хранения данных использовали SQLite, а для доступа к приложению – код-пароль, который является ключом к БД.

**Листинг**

MainActivity

**package** com.example.login;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** java.util.\*;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 Button **btn**;  
 EditText **psd**;  
 *//public ArrayList<String> originalPassword = new ArrayList<>();* **public static** String *originalPassword*;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 SharedPreferences pref = getSharedPreferences(**"NEW\_PASS"**, ***MODE\_PRIVATE***);  
 String new\_passs = pref.getString(**"new\_pass"**, **"N/A"**);  
 **if**(new\_passs.equals(**"N/A"**))  
 {  
 *originalPassword*=**"123366"**;  
 }  
 **else** {  
 *originalPassword*=*decode*(Base64.*getDecoder*().decode(new\_passs),**"test"**);  
 }  
 **psd**=(EditText) findViewById(R.id.***psd***);  
 **btn**=(Button) findViewById(R.id.***btn***);  
  
 **btn**.setOnClickListener((**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **if**(**psd**.getText().toString().equals(*originalPassword*))  
 {  
 *//Toast.makeText(getApplicationContext(),"Succsess login",Toast.LENGTH\_LONG).show();* Intent myintent = **new** Intent(MainActivity.**this**,LoggedActivity.**class**);  
 startActivity(myintent);  
 }  
 **else** {  
 Toast.*makeText*(getApplicationContext(),**"No login"**,Toast.***LENGTH\_LONG***).show();  
  
 }  
 }  
 }));  
 }  
 **public static** String getter()  
 {  
 *//return originalPassword;* **int** length =32;  
 String letter = **"G"**;  
 String adding=**""**;  
 **if**(*originalPassword*.length()<=length)  
 {  
 **int** how\_much=length-*originalPassword*.length();  
 **for**(**int** i=0;i<how\_much;i++)  
 {  
 *//originalPassword=originalPassword+adding;* adding=adding.concat(letter);  
 }  
 **return** *originalPassword*+adding;  
 }  
 **return ""**;  
 }  
 **public static void** setter(String new\_code)  
 {  
 *originalPassword*=new\_code;  
 }  
  
  
  
  
  
 **public static** String decode(**byte**[] pText, String pKey) {  
 **byte**[] res = **new byte**[pText.**length**];  
 **byte**[] key = pKey.getBytes();  
  
 **for** (**int** i = 0; i < pText.**length**; i++) {  
 res[i] = (**byte**) (pText[i] ^ key[i % key.**length**]);  
 }  
  
 **return new** String(res);  
 }  
  
}

LoggedActivity

**package** com.example.login;  
  
**import** androidx.annotation.Nullable;  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.content.ContentValues;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.os.Environment;  
**import** android.os.Parcelable;  
**import** android.support.v4.os.IResultReceiver;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.Toast;  
  
**import** java.io.BufferedReader;  
**import** java.io.BufferedWriter;  
**import** java.io.File;  
**import** java.io.FileNotFoundException;  
**import** java.io.FileReader;  
**import** java.io.FileWriter;  
**import** java.io.IOException;  
**import** java.security.InvalidAlgorithmParameterException;  
**import** java.security.InvalidKeyException;  
**import** java.security.NoSuchAlgorithmException;  
**import** java.util.ArrayList;  
**import** java.util.Base64;  
  
**import** javax.crypto.NoSuchPaddingException;  
  
**public class** LoggedActivity **extends** AppCompatActivity {  
 **static** String *key*;  
 **private** MainActivity **pass**;  
 String **Resourse**,**Login**,**password**,**notes**;  
 String **delete\_from\_database**;  
 **static** SQLiteDatabase *database*;  
 **final** AES **aes**=**new** AES();  
  
  
 **public** DBHelper **dbHelper**;  
  
 Button **add\_info**;  
 Button **del\_button**;  
 Button **change\_button**;  
 Button **change\_pass**;  
 Button **export\_button**;  
 Button **import\_button**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_logged***);  
 *key*=MainActivity.*getter*();*//получаем код-пароль* **dbHelper** = **new** DBHelper(**this**);  
 *database*= **dbHelper**.getWritableDatabase();  
  
  
  
  
  
  
  
 **add\_info**=(Button) findViewById(R.id.***add\_info***);  
 **add\_info**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
  
  
  
 *//Toast.makeText(getApplicationContext(),"Adding in DataBase",Toast.LENGTH\_LONG).show();* Intent myintent = **new** Intent(LoggedActivity.**this**,AddActivity.**class**);  
 *//myintent.putExtra("EXTRA\_DATA\_BASE", (Parcelable) dbHelper);* startActivityForResult(myintent,1);  
 }  
 });  
  
  
 **del\_button**=(Button)findViewById(R.id.***del\_button***);  
 **del\_button**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
 *//Toast.makeText(getApplicationContext(),"Deleteing from DataBase",Toast.LENGTH\_LONG).show();* Intent myintent = **new** Intent(LoggedActivity.**this**,DeleteActivity.**class**);  
 *//myintent.putExtra("EXTRA\_DATA\_BASE", (Parcelable) dbHelper);* startActivityForResult(myintent,2);  
  
 }  
 });  
  
 **change\_button**=(Button)findViewById(R.id.***change\_button***);  
 **change\_button**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v)  
 {  
 *//Toast.makeText(getApplicationContext(),"Changing or lookng in DataBase",Toast.LENGTH\_LONG).show();* Intent myintent = **new** Intent(LoggedActivity.**this**,ChangeActivity.**class**);  
 *//myintent.putExtra("EXTRA\_DATA\_BASE", (Parcelable) dbHelper);* startActivityForResult(myintent,3);  
  
 }  
 });  
 **change\_pass**=(Button)findViewById(R.id.***button2***);  
 **change\_pass**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//Toast.makeText(getApplicationContext(),"Changing key-logon password",Toast.LENGTH\_LONG).show();* Intent myintent = **new** Intent(LoggedActivity.**this**,ChangePassActivity.**class**);  
 *//myintent.putExtra("EXTRA\_DATA\_BASE", (Parcelable) dbHelper);* startActivityForResult(myintent,4);  
 }  
 });  
 **export\_button**=(Button)findViewById(R.id.***ex\_button***);  
 **export\_button**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v)  
 {  
  
  
 Cursor cursor2 = *database*.query(DBHelper.***TABLE\_CONTACTS***, **null**, **null**, **null**, **null**, **null**, **null**);  
  
  
 **if** (cursor2.moveToFirst())  
 {  
 **int** idIndex = cursor2.getColumnIndex(DBHelper.***KEY\_ID***);  
 **int** resourceIndex = cursor2.getColumnIndex(DBHelper.***KEY\_RESOURCE***);  
 **int** loginIndex = cursor2.getColumnIndex(DBHelper.***KEY\_LOGIN***);  
 **int** passIndex=cursor2.getColumnIndex(DBHelper.***KEY\_PASSWORD***);  
 **int** notesIndex=cursor2.getColumnIndex(DBHelper.***KEY\_NOTES***);  
 **int** id=0;  
 */\*SharedPreferences sPref = getSharedPreferences("EXPORT\_DB", Context.MODE\_PRIVATE);  
 SharedPreferences.Editor ed = sPref.edit();\*/* **final** String LOG\_TAG = **"myLogs"**;  
  
 **final** String FILENAME = **"file"**;  
  
 **final** String DIR\_SD = **"MyFiles"**;  
 **final** String FILENAME\_SD = **"fileSD"**;  
 **if** (!Environment.*getExternalStorageState*().equals(  
 Environment.***MEDIA\_MOUNTED***))  
 {  
 Toast.*makeText*(getApplicationContext(),**"CD-card not valid"**,Toast.***LENGTH\_LONG***).show();  
 Log.*d*(LOG\_TAG, **"SD-карта не доступна: "** + Environment.*getExternalStorageState*());  
 **return**;  
 }  
 *// получаем путь к SD* File sdPath = Environment.*getExternalStorageDirectory*();  
 *// добавляем свой каталог к пути* sdPath = **new** File(sdPath.getAbsolutePath() + **"/"** + DIR\_SD);  
 *// создаем каталог* sdPath.mkdirs();  
 *// формируем объект File, который содержит путь к файлу* File sdFile = **new** File(sdPath, FILENAME\_SD);  
 BufferedWriter bw = **null**;  
 **try** {  
 bw = **new** BufferedWriter(**new** FileWriter(sdFile));  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
 **do** {  
 */\* id=cursor2.getInt(idIndex);  
 ed.putInt("export\_id"+String.valueOf(id),cursor2.getInt(idIndex));  
 ed.putString("export\_resource"+String.valueOf(id),cursor2.getString(resourceIndex));  
 ed.putString("export\_login"+String.valueOf(id),cursor2.getString(loginIndex));  
 ed.putString("export\_pass"+String.valueOf(id),cursor2.getString(passIndex));  
 ed.putString("export\_notes"+String.valueOf(id),cursor2.getString(notesIndex));  
 ed.commit();\*/* **try** {  
 *// открываем поток для записи  
  
 // пишем данные* bw.write(**"1 "**+cursor2.getString(resourceIndex));  
 bw.write(**"\n"**);  
 bw.write(**"2 "**+cursor2.getString(loginIndex));  
 bw.write(**"\n"**);  
 bw.write(**"3 "**+cursor2.getString(passIndex));  
 bw.write(**"\n"**);  
 bw.write(**"4 "**+cursor2.getString(notesIndex));  
 bw.write(**"\n"**);  
  
  
 Log.*d*(LOG\_TAG, **"Файл записан на SD: "** + sdFile.getAbsolutePath());  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
  
  
 } **while** (cursor2.moveToNext());  
 *// закрываем поток* **try** {  
 bw.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 Toast.*makeText*(getApplicationContext(),**"DB exported successfully"**,Toast.***LENGTH\_LONG***).show();  
 } **else** {  
 Toast.*makeText*(getApplicationContext(),**"DB is empty"**,Toast.***LENGTH\_LONG***).show();  
  
 }  
  
  
 cursor2.close();  
  
  
  
 }  
 });  
  
 **import\_button**=(Button)findViewById(R.id.***im\_button***);  
 **import\_button**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *database* = **dbHelper**.getWritableDatabase();  
 ContentValues contentValues =**new** ContentValues();  
  
 **final** String LOG\_TAG = **"myLogs"**;  
 **final** String FILENAME = **"file"**;  
 **final** String DIR\_SD = **"MyFiles"**;  
 **final** String FILENAME\_SD = **"fileSD"**;  
 **if** (!Environment.*getExternalStorageState*().equals(  
 Environment.***MEDIA\_MOUNTED***)) {  
 Log.*d*(LOG\_TAG, **"SD-карта не доступна: "** + Environment.*getExternalStorageState*());  
 **return**;  
 }  
 *// получаем путь к SD* File sdPath = Environment.*getExternalStorageDirectory*();  
 *// добавляем свой каталог к пути* sdPath = **new** File(sdPath.getAbsolutePath() + **"/"** + DIR\_SD);  
 *// формируем объект File, который содержит путь к файлу* File sdFile = **new** File(sdPath, FILENAME\_SD);  
 **try** {  
 *// открываем поток для чтения* BufferedReader br = **new** BufferedReader(**new** FileReader(sdFile));  
 String str = **""**;  
 **int** id=0;  
 String id\_str=**""**;  
 *// читаем содержимое* **while** ((str = br.readLine()) != **null**)  
 {  
 id\_str=str.substring(0,1);  
 id=Integer.*parseInt*((id\_str));  
  
 **if**(id==1)  
 {  
 str=str.substring(2);  
 contentValues.put(DBHelper.***KEY\_RESOURCE***, str);  
 }  
 **if**(id==2)  
 {  
 str=str.substring(2);  
 contentValues.put(DBHelper.***KEY\_LOGIN***, str);  
 }  
 **if**(id==3)  
 {  
 str=str.substring(2);  
 contentValues.put(DBHelper.***KEY\_PASSWORD***, str);  
 }  
 **if**(id==4)  
 {  
 str=str.substring(2);  
 contentValues.put(DBHelper.***KEY\_NOTES***, str);  
 *database*.insert(DBHelper.***TABLE\_CONTACTS***, **null**, contentValues);  
 Toast.*makeText*(getApplicationContext(),**"Succsesfully imported"**,Toast.***LENGTH\_LONG***).show();  
  
 }  
  
 }  
 }  
 **catch** (FileNotFoundException e)  
 {  
 e.printStackTrace();  
 }  
 **catch** (IOException e)  
 {  
 e.printStackTrace();  
 }  
  
  
  
 }  
 });  
  
  
  
  
  
  
  
  
 }  
  
  
  
  
  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 *// here AcivityB is finished. Call your method* **super**.onActivityResult(requestCode, resultCode, data);  
  
 *database* = **dbHelper**.getWritableDatabase();  
  
 ContentValues contentValues =**new** ContentValues();  
  
  
 **if**(requestCode==1)  
 {  
 *// meaning that it is add activity ended* SharedPreferences pref = getSharedPreferences(**"SAVED\_TEXT"**, ***MODE\_PRIVATE***);  
 **Resourse** = pref.getString(**"resource"**, **"N/A"**);  
 **Login** = pref.getString(**"login"**, **"N/A"**);  
 **password** = pref.getString(**"password"**, **"N/A"**);  
 **notes** = pref.getString(**"notes"**, **"N/A"**);  
  
  
 contentValues.put(DBHelper.***KEY\_RESOURCE***, **Resourse**);  
 contentValues.put(DBHelper.***KEY\_LOGIN***, **Login**);  
 contentValues.put(DBHelper.***KEY\_PASSWORD***, **password**);  
 contentValues.put(DBHelper.***KEY\_NOTES***, **notes**);  
  
 *database*.insert(DBHelper.***TABLE\_CONTACTS***, **null**, contentValues);  
  
  
  
  
 pref.edit().clear().commit();  
  
  
  
 }  
 Cursor cursor2 = *database*.query(DBHelper.***TABLE\_CONTACTS***, **null**, **null**, **null**, **null**, **null**, **null**);  
  
 **if** (cursor2.moveToFirst()) {  
 **int** idIndex = cursor2.getColumnIndex(DBHelper.***KEY\_ID***);  
 **int** resourceIndex = cursor2.getColumnIndex(DBHelper.***KEY\_RESOURCE***);  
 **int** loginIndex = cursor2.getColumnIndex(DBHelper.***KEY\_LOGIN***);  
 **do** {  
 Log.*d*(**"mLog"**, **"ID = "** + cursor2.getInt(idIndex) +  
 **", resource name = "** + cursor2.getString(resourceIndex) +  
 **", login = "** + cursor2.getString(loginIndex));  
 } **while** (cursor2.moveToNext());  
 } **else** Log.*d*(**"mLog"**,**"0 rows"**);  
  
 cursor2.close();  
 **if**(requestCode==2)  
 {  
 SharedPreferences pref = getSharedPreferences(**"DEL\_TEXT"**, ***MODE\_PRIVATE***);  
 **delete\_from\_database** = pref.getString(**"resource\_deletable"**, **"N/A"**);  
 **int** id\_del=0;  
  
  
 Cursor cursor = *database*.query(DBHelper.***TABLE\_CONTACTS***, **null**, **null**, **null**, **null**, **null**, **null**);  
  
 **if** (cursor.moveToFirst()) {  
 **int** idIndex = cursor.getColumnIndex(DBHelper.***KEY\_ID***);  
 **int** resourceIndex = cursor.getColumnIndex(DBHelper.***KEY\_RESOURCE***);  
 **do** {  
 String decrypted=*decrypt*(cursor.getString(resourceIndex));  
 **if**(decrypted.equals(**delete\_from\_database**))  
 {  
 id\_del=cursor.getInt(idIndex);  
 **break**;  
 }  
  
 } **while** (cursor.moveToNext());  
 }  
 cursor.close();  
  
  
  
 **int** delCount = *database*.delete(DBHelper.***TABLE\_CONTACTS***, DBHelper.***KEY\_ID*** + **"="** + id\_del, **null**);  
  
 Log.*d*(**"mLog"**, **"deleted rows count = "** + delCount);  
  
  
  
 }  
  
  
  
 Cursor cursor3 = *database*.query(DBHelper.***TABLE\_CONTACTS***, **null**, **null**, **null**, **null**, **null**, **null**);  
  
 **if** (cursor3.moveToFirst()) {  
 **int** idIndex = cursor3.getColumnIndex(DBHelper.***KEY\_ID***);  
 **int** resourceIndex = cursor3.getColumnIndex(DBHelper.***KEY\_RESOURCE***);  
 **int** loginIndex = cursor3.getColumnIndex(DBHelper.***KEY\_LOGIN***);  
 **do** {  
 Log.*d*(**"mLog"**, **"ID = "** + cursor3.getInt(idIndex) +  
 **", resource name = "** + cursor3.getString(resourceIndex) +  
 **", login = "** + cursor3.getString(loginIndex));  
 } **while** (cursor3.moveToNext());  
 } **else** Log.*d*(**"mLog"**,**"0 rows"**);  
  
 cursor3.close();  
 **if**(requestCode==3)  
 {  
 SharedPreferences pref = getSharedPreferences(**"CHANGE\_TEXT"**, ***MODE\_PRIVATE***);  
 **int** id=pref.getInt(**"identificator"**,0);  
 **Resourse**=pref.getString(**"resource\_change"**,**"N/A"**);  
 **Login** = pref.getString(**"login\_change"**, **"N/A"**);  
 **password** = pref.getString(**"password\_change"**, **"N/A"**);  
 **notes** = pref.getString(**"notes\_change"**, **"N/A"**);  
 contentValues.put(DBHelper.***KEY\_RESOURCE***, **Resourse**);  
 contentValues.put(DBHelper.***KEY\_LOGIN***, **Login**);  
 contentValues.put(DBHelper.***KEY\_PASSWORD***, **password**);  
 contentValues.put(DBHelper.***KEY\_NOTES***, **notes**);  
 **int** updCount = *database*.update(DBHelper.***TABLE\_CONTACTS***, contentValues, DBHelper.***KEY\_ID*** + **"= ?"**, **new** String[] {String.*valueOf*(id)});  
  
 }  
  
  
  
  
 Cursor cursor4 = *database*.query(DBHelper.***TABLE\_CONTACTS***, **null**, **null**, **null**, **null**, **null**, **null**);  
  
 **if** (cursor4.moveToFirst()) {  
 **int** idIndex = cursor4.getColumnIndex(DBHelper.***KEY\_ID***);  
 **int** resourceIndex = cursor4.getColumnIndex(DBHelper.***KEY\_RESOURCE***);  
 **int** loginIndex = cursor4.getColumnIndex(DBHelper.***KEY\_LOGIN***);  
 **do** {  
 Log.*d*(**"mLog"**, **"ID = "** + cursor4.getInt(idIndex) +  
 **", resource name = "** + cursor4.getString(resourceIndex) +  
 **", login = "** + cursor4.getString(loginIndex));  
 } **while** (cursor4.moveToNext());  
 } **else** Log.*d*(**"mLog"**,**"0 rows"**);  
  
 cursor4.close();  
  
  
 **if**(requestCode==4)  
 {  
 SharedPreferences pref = getSharedPreferences(**"SAVED\_PASSWORD"**, ***MODE\_PRIVATE***);  
 String new\_key;  
 new\_key = pref.getString(**"new\_password"**, **"N/A"**);  
  
  
 Cursor cursor = *database*.query(DBHelper.***TABLE\_CONTACTS***, **null**, **null**, **null**, **null**, **null**, **null**);  
  
 **if** (cursor.moveToFirst()) {  
 String resource;  
 String login;  
 String pass;  
 String notes;  
 **int** idIndex = cursor.getColumnIndex(DBHelper.***KEY\_ID***);  
 **int** resourceIndex = cursor.getColumnIndex(DBHelper.***KEY\_RESOURCE***);  
 **int** loginIndex = cursor.getColumnIndex(DBHelper.***KEY\_LOGIN***);  
 **int** passIndex= cursor.getColumnIndex(DBHelper.***KEY\_PASSWORD***);  
 **int** notesIndex= cursor.getColumnIndex(DBHelper.***KEY\_NOTES***);  
 **int** id=0;  
 String old\_key=MainActivity.*getter*();  
 **do** {  
 MainActivity.*setter*(old\_key);  
 id=cursor.getInt(idIndex);  
 resource=*decrypt*(cursor.getString(resourceIndex));  
 login=*decrypt*(cursor.getString(loginIndex));  
 pass=*decrypt*(cursor.getString(passIndex));  
 notes=*decrypt*(cursor.getString(notesIndex));  
  
 SharedPreferences sPref = getSharedPreferences(**"NEW\_PASS"**, Context.***MODE\_PRIVATE***);  
 SharedPreferences.Editor ed = sPref.edit();  
 ed.putString(**"new\_pass"**, *encode*(new\_key,**"test"**));  
 ed.commit();  
  
 MainActivity.*setter*(new\_key);  
 resource=encrypt\_string(resource);  
 login=encrypt\_string(login);  
 pass=encrypt\_string(pass);  
 notes=encrypt\_string(notes);  
  
 contentValues.put(DBHelper.***KEY\_RESOURCE***, resource);  
 contentValues.put(DBHelper.***KEY\_LOGIN***, login);  
 contentValues.put(DBHelper.***KEY\_PASSWORD***, pass);  
 contentValues.put(DBHelper.***KEY\_NOTES***, notes);  
 **int** updCount = *database*.update(DBHelper.***TABLE\_CONTACTS***, contentValues, DBHelper.***KEY\_ID*** + **"= ?"**, **new** String[] {String.*valueOf*(id)});  
  
  
  
  
 } **while** (cursor.moveToNext());  
 } **else** Log.*d*(**"mLog"**,**"0 rows"**);  
  
 cursor.close();  
  
  
  
  
  
  
  
 }  
  
  
  
  
  
  
  
  
  
  
  
 }  
  
 **static boolean** ifDBHasThis(String resource)  
 {  
 Cursor cursor = *database*.query(DBHelper.***TABLE\_CONTACTS***, **null**, **null**, **null**, **null**, **null**, **null**);  
  
 **if** (cursor.moveToFirst()) {  
 **int** idIndex = cursor.getColumnIndex(DBHelper.***KEY\_ID***);  
 **int** resourceIndex = cursor.getColumnIndex(DBHelper.***KEY\_RESOURCE***);  
 **do** {  
 String decrypted=*decrypt*(cursor.getString(resourceIndex));  
 **if**(decrypted.equals(resource))  
 {  
 cursor.close();  
  
 **return true**;  
 }  
 } **while** (cursor.moveToNext());  
 } **else** cursor.close();  
 **return false**;  
 }  
  
  
  
  
  
 **static** ArrayList<String> GetInfoFromDB(String resource)  
 {  
 ArrayList<String> info=**new** ArrayList<>();  
  
 Cursor cursor2 = *database*.query(DBHelper.***TABLE\_CONTACTS***, **null**, **null**, **null**, **null**, **null**, **null**);  
  
 **if** (cursor2.moveToFirst()) {  
 **int** idIndex = cursor2.getColumnIndex(DBHelper.***KEY\_ID***);  
 **int** resourceIndex = cursor2.getColumnIndex(DBHelper.***KEY\_RESOURCE***);  
 **int** loginIndex = cursor2.getColumnIndex(DBHelper.***KEY\_LOGIN***);  
 **int** passIndex = cursor2.getColumnIndex(DBHelper.***KEY\_PASSWORD***);  
 **int** notesIndex = cursor2.getColumnIndex(DBHelper.***KEY\_NOTES***);  
 **do** {  
 String decrypted=*decrypt*(cursor2.getString(resourceIndex));  
 **if**(decrypted.equals(resource))  
 {  
 info.add(*decrypt*(cursor2.getString(resourceIndex)));  
 info.add(*decrypt*(cursor2.getString(loginIndex)));  
 info.add(cursor2.getString(passIndex));  
 info.add(*decrypt*(cursor2.getString(notesIndex)));  
 cursor2.close();  
 **return** info;  
 }  
  
 } **while** (cursor2.moveToNext());  
 } **else** cursor2.close();  
 **return** info;  
 }  
  
  
 **static int** GetIdFromDB(String resource)  
 {  
 **int** id=0;  
 Cursor cursor2 = *database*.query(DBHelper.***TABLE\_CONTACTS***, **null**, **null**, **null**, **null**, **null**, **null**);  
  
 **if** (cursor2.moveToFirst()) {  
 **int** idIndex = cursor2.getColumnIndex(DBHelper.***KEY\_ID***);  
 **int** resourceIndex = cursor2.getColumnIndex(DBHelper.***KEY\_RESOURCE***);  
 **do** {  
 String decrypted=*decrypt*(cursor2.getString(resourceIndex));  
 **if**(decrypted.equals(resource))  
 {  
 id=cursor2.getInt(idIndex);  
 cursor2.close();  
 **return** id;  
 }  
  
 } **while** (cursor2.moveToNext());  
 } **else** cursor2.close();  
 **return** id;  
  
 }  
  
 **static** String getter()  
 {  
 *key*=MainActivity.*getter*();*//получаем код-пароль* **return** *key*;  
 }  
  
 **private static** String decrypt(String arg)  
 {  
 **byte**[] encode = Base64.*getDecoder*().decode(arg);*//* **byte**[] iv = **new byte**[16];  
 **byte**[] encoded=**null**;  
 String test=**""**;  
 **try** {  
 encoded=*read*(encode,**"CFB"**,iv);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **final** AES aes=**new** AES();  
 **try** {  
 test= aes.decrypt(encoded,MainActivity.*getter*(),**"CFB"**,iv);  
 } **catch** (InvalidKeyException e) {  
 e.printStackTrace();  
 } **catch** (InvalidAlgorithmParameterException e) {  
 e.printStackTrace();  
 }  
 **return** test;  
  
  
 }  
 **private static byte**[] read(**byte**[] encode, String mode, **byte**[] iv) **throws** IOException {  
 **byte**[] encoded = encode;  
 **if** (**"CFB"**.equals(mode)) {  
 **int** y = 0;  
 **for** (**int** i = encoded.**length** - 16; y < 16; i++) {  
 iv[y] = encoded[i];  
 y++;  
 }  
  
 **byte**[] encoded\_shrinked = **new byte**[encoded.**length**-16];  
 System.*arraycopy*(encoded, 0, encoded\_shrinked, 0, encoded.**length**-16);  
  
 **return** encoded\_shrinked;  
 }  
 **return** encoded;  
 }  
  
  
  
  
  
  
 **private** String encrypt\_string(String arg)  
 {  
 **byte**[] a=**null**;  
 String key=MainActivity.*getter*();*// получаем свежий ключ* **try** {  
 a=**aes**.encrypt(arg,key,**"CFB"**);  
 } **catch** (InvalidKeyException e) {  
 e.printStackTrace();  
 } **catch** (InvalidAlgorithmParameterException e) {  
 e.printStackTrace();  
 } **catch** (NoSuchPaddingException e) {  
 e.printStackTrace();  
 } **catch** (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 }  
 System.***out***.println(Base64.*getEncoder*().encodeToString(a));  
 **return** (Base64.*getEncoder*().encodeToString(a));  
  
  
  
  
 }  
  
  
  
  
  
  
 **public static** String encode(String pText, String pKey) {  
 **byte**[] txt = pText.getBytes();  
 **byte**[] key = pKey.getBytes();  
 **byte**[] res = **new byte**[pText.length()];  
  
 **for** (**int** i = 0; i < txt.**length**; i++) {  
 res[i] = (**byte**) (txt[i] ^ key[i % key.**length**]);  
 }  
  
 **return** Base64.*getEncoder*().encodeToString(res);  
 }  
  
  
  
  
  
  
}

Deleteactivity

**package** com.example.login;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** java.security.InvalidAlgorithmParameterException;  
**import** java.security.InvalidKeyException;  
**import** java.security.NoSuchAlgorithmException;  
**import** java.util.Base64;  
  
**import** javax.crypto.NoSuchPaddingException;  
  
**public class** DeleteActivity **extends** AppCompatActivity {  
 EditText **Resourse**;  
 Button **ok\_del**;  
 String **res**;  
 **final** AES **aes**=**new** AES();  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_delete***);  
  
 **Resourse**=(EditText)findViewById(R.id.***editTextTextPersonName***);  
 **ok\_del**=(Button) findViewById(R.id.***ok\_button\_del***);  
  
 **ok\_del**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **res**=**Resourse**.getText().toString();  
 **if**(!LoggedActivity.*ifDBHasThis*(**res**))  
 {  
 Toast.*makeText*(getApplicationContext(),**"Nothing to delete in DATA\_BASE"**,Toast.***LENGTH\_LONG***).show();  
 *//ok.setEnabled(false);* **return**;  
 }  
  
  
 SharedPreferences sPref = getSharedPreferences(**"DEL\_TEXT"**, Context.***MODE\_PRIVATE***);  
 SharedPreferences.Editor ed = sPref.edit();  
 *//res=encrypt\_string(res);* ed.putString(**"resource\_deletable"**, **res**);  
  
 ed.commit();  
  
  
  
 Intent intent= **new** Intent();  
 intent.putExtra(**"result"**,2);  
 setResult(***RESULT\_OK***,intent);  
 finish();  
  
 }  
 });  
  
  
  
  
 }  
  
  
  
 **private** String encrypt\_string(String arg)  
 {  
 **byte**[] a=**null**;  
 String key=MainActivity.*getter*();*// получаем свежий ключ* **try** {  
 a=**aes**.encrypt(arg,key,**"CFB"**);  
 } **catch** (InvalidKeyException e) {  
 e.printStackTrace();  
 } **catch** (InvalidAlgorithmParameterException e) {  
 e.printStackTrace();  
 } **catch** (NoSuchPaddingException e) {  
 e.printStackTrace();  
 } **catch** (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 }  
 System.***out***.println(Base64.*getEncoder*().encodeToString(a));  
 **return** (Base64.*getEncoder*().encodeToString(a));  
  
  
  
  
 }  
  
  
  
  
  
  
  
  
  
  
  
  
  
}