1. Создаем слой LinearLayout (vertical).

2. Изменить цвет фона на серый:

```
background="#666"
```

3. Добавить:

–картинку ImageView;

– текст TextView;

– поле вводаPlainText(в коде обозначенEditText) – для ввода login;

– поле ввода PlainText(в коде обозначенEditText) – для ввода e-mail;

– поле ввода PlainText(в коде обозначен EditText) – дляввода пароля;

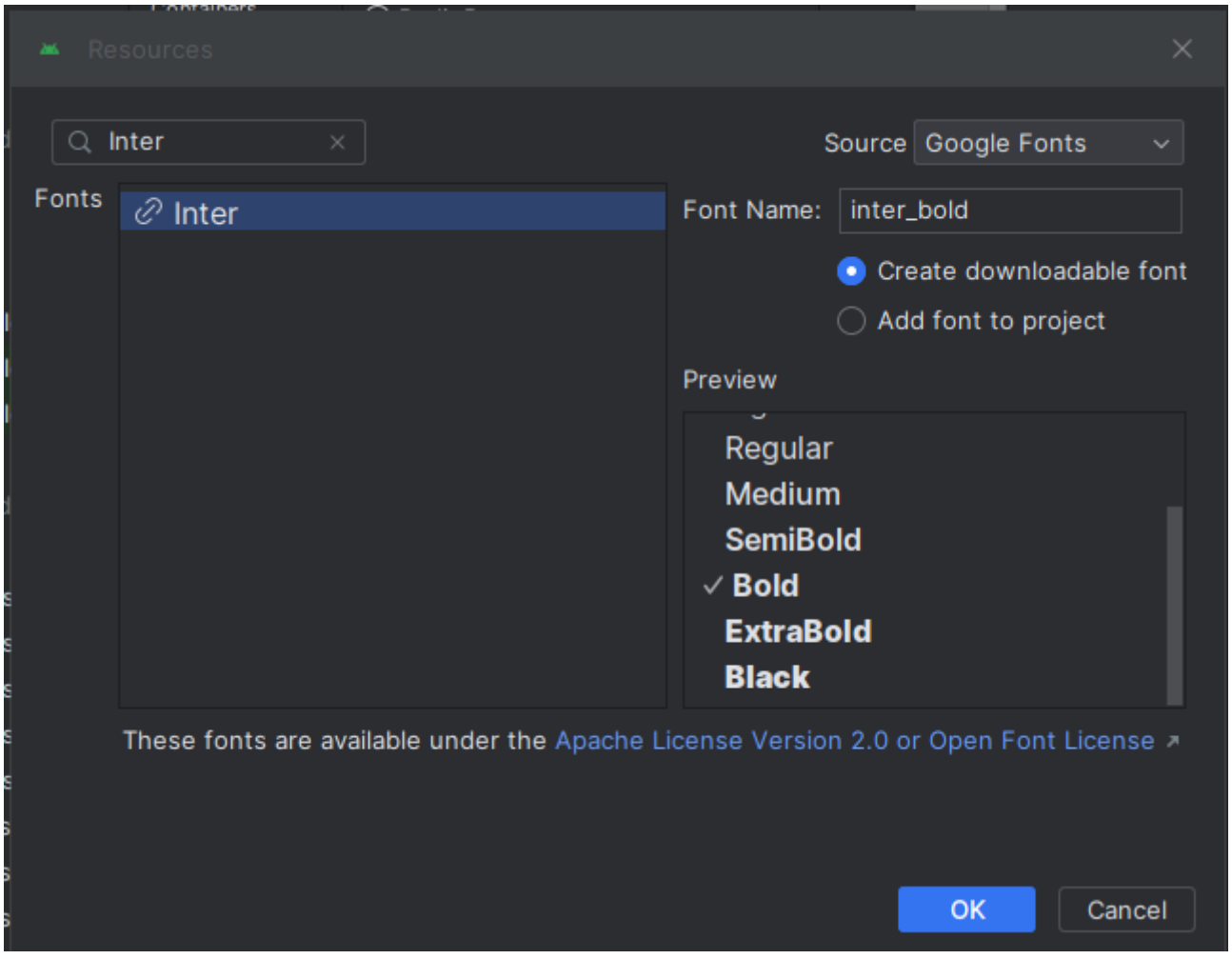
– кнопку Button.

4. Отформатировать добавленные элементы:

4.1. ImageView

4.2. TextView:

Шрифт– черезDesignищем fontFamily – More Fonts – Inter, Bold



Цвет текста – в коде #fdfdfd

```
android:textColor="#fdfdfd"
```

Выравнивание текста по центру:

```
android:layout_gravity="center"
android:gravity="center_horizontal"
```

Отступсверху 170 пикселей:

```
android:layout_marginTop="170dp "
```

Размер текста 35 пикселей:

```
android:textSize="35sp"
```

Стиль текста – жирный:

```
android:textStyle="bold"
```

4.3. EditText – дляводалogin

Изменим ID на user_login:

```
android:id="@+id/user_login"
```

Ширина 240 пикселей, высота 50 пискселей:

```
android:layout_width="240dp"
android:layout_height="50dp"
```

Заменяем слово text на hint (подсказка)

```
android:hint="Введите логин"
```

Меняем цвет текста подсказки на зеленоватый:

```
android:textColorHint="#ada"
```

Отступ 20 пикселейсверху:

```
android:layout_marginTop="20dp"
```

Выравнивание ПОЛЯ по центру:

Запишите сами

Цвет текста, который пользователь будет вводить в поле (белый):

```
android:textColor="#fdfdfd"
```

4.4.EditText – дляводае-mail

Копируем предыдущий код и меняем:

Id на user_email

inputType на textWebEmailAddress (типводимойинформации)

hint на «Введите email»

4.5. EditText – дляводапароля

Копируем предыдущий код и меняем:

Id на user_pass

inputType на textPassword (типводимойинформации – любой символ в этом поле будет в виде кружка)

hint на «Введите пароль»

4.6. Button

Меняем id на button_reg

```
android:id="@+id/button_reg"
```

Ширинакнопки 220 пикселей:

```
android:layout_width="220dp"
```

Меняем текст на кнопке «Регистрация»

```
android:text="Регистрация"
```

Стильтекста – жирный:

```
android:textStyle="bold"
```

Размер текста на кнопке:

```
android:textSize="16sp"
```

Цвет текста на кнопке серый – 666:

```
android:textColor="#666"
```

Выравнивание кнопки по центру…

Сами

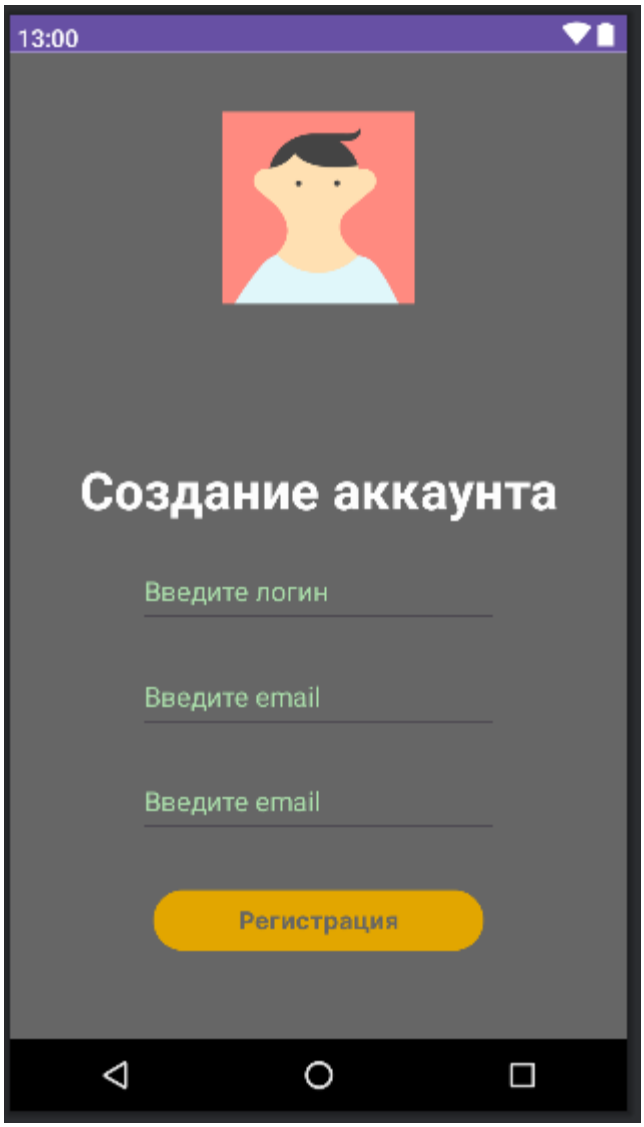
Цветкнопки e2a600:

```
android:backgroundTint="#e2a600"
```

Отступсверху 30 пикселей:

```
android:layout_marginTop="30dp"
```

ИТОГ:




```
<LinearLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#666"
android:orientation="vertical">
```

```
<ImageView
android:id="@+id/imageView2"
android:layout_marginTop="40dp"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:src="@tools:sample/avatars" />

<TextView
android:id="@+id/textView2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="100dp"
android:gravity="center_horizontal"
android:text="Созданиеаккаунта"
android:textColor="#fdfdfd"
android:textSize="35sp"
android:textStyle="bold" />

<EditText
android:id="@+id/user_login"
android:layout_width="240dp"
android:layout_height="50dp"
android:ems="10"
android:layout_marginTop="20dp"
android:layout_gravity="center"
android:textColor="#fdfdfd"
android:inputType="text"
android:textColorHint="#ada"
android:hint="Введителогин" />

<EditText
android:id="@+id/user_email"
android:layout_width="240dp"
android:layout_height="50dp"
android:ems="10"
android:layout_marginTop="20dp"
android:layout_gravity="center"
android:textColor="#fdfdfd"
android:inputType="textEmailAddress"
android:textColorHint="#ada"
android:hint="Введите email" />

<EditText
android:id="@+id/user_pass"
android:layout_width="240dp"
android:layout_height="50dp"
android:ems="10"
android:layout_marginTop="20dp"
android:layout_gravity="center"
android:textColor="#fdfdfd"
android:inputType="textPassword"
android:textColorHint="#ada"
android:hint="Введитепароль" />

<Button
```

```
android:id="@+id/button_reg"
android:layout_width="220dp"
android:layout_height="wrap_content"
android:fontFamily="@font/inter_bold"
android:text="Регистрация"
android:textStyle="bold"
android:textSize="16sp"
android:textColor="#666"
android:layout_gravity="center"
android:backgroundTint="#e2a600"
android:layout_marginTop="30dp"
/>
</LinearLayout>
```

Функционал страницы

Переходим в файл MainActivity.kt

Создаем несколько переменных, которые будут ссылаться на объекты из Design. При нажатии на объекты будем получать данные от пользователей, регистрировать пользователя в файл или базу данных.

1. Создаем переменную userLogin

```
val userLogin: EditText = findViewById(R.id.user_login)
```

Переменная userLogin на основе класса EditText.

В переменную будем записывать определенный объект через функцию findViewById(R.id.user_login)

2. Скопировав создаем еще 3-и переменные userEmail,userPass и button.

```
val userLogin: EditText = findViewById(R.id.user_login)
val userEmail: EditText = findViewById(R.id.user_email)
val userPass: EditText = findViewById(R.id.user_pass)
val button: Button = findViewById(R.id.button_reg)
```

Сделайте импорт класса по запросу программы.

3. Создадим обработчик событий при нажатии на кнопки, обратившись к функции setOnClickListener

```
button.setOnClickListener
```

3.1. В обработчике создадим несколько полей в которые будем помещать то, что получим от пользователя

```
val login = userLogin.text.toString().trim()
```

Здесь: обращаемся к тексту userLogin, текст этот приводим к формату «Строка» и дополнительно вызываем функцию trimдля удаления всех пробелов из текста

Аналогично создаем другие переменные email и pass, обращаясь теперь к userEmail и userPass:

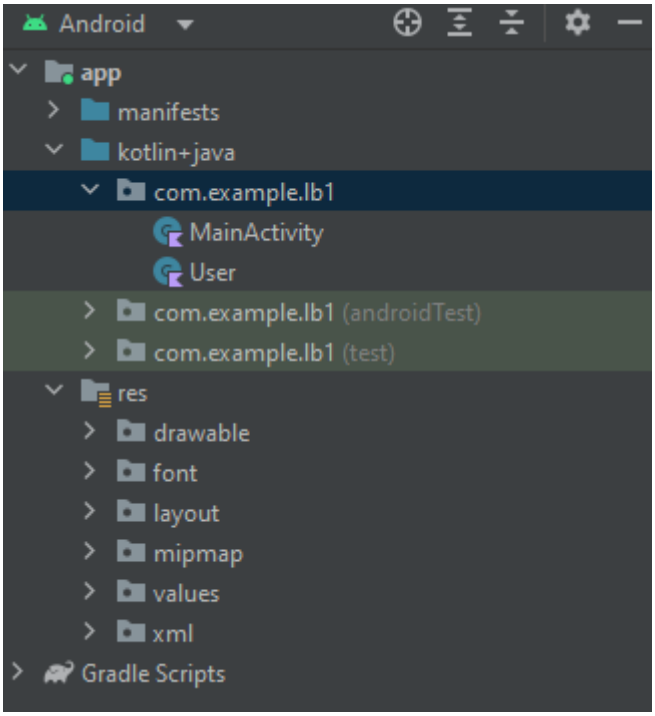
```
button.setOnClickListener{
val login = userLogin.text.toString().trim()
val email = userEmail.text.toString().trim()
val pass = userPass.text.toString().trim()
```

Далее пишем условие «Если одно из полей будет пустым, то создаем всплывающую подсказку с фразой «Не все поля заполнены», если все ОК, то будем выполнять регистрацию пользователей».

```
if(login == "" || email == "" || pass == "")
    Toast.makeText(this, "Не все поля заполнены",
Toast.LENGTH_LONG).show()
else {

}
```

3.2. Создадим отдельный класс User на основе которых будем описывать конкретных пользователей.

На папке с MainActivity жмем левую кнопку мыши – New – KotlinClass/File



После названия класса в круглых скобках укажем переменные login, email, pass.

По сути создали конструктор через которые сможем сразу установить все эти значения при создании объекта. На базе такого класса будем создавать объекты.

```
packagecom.example.lb1

class User (val login: String, val email: String, val pass: String){
}
```

Возвращаемся в MainActivity в else:

В else будем создавать новый объект user, он будет основан на классе, который мы только создали User.

В этот класс User при создании объекта будем передавать 3 значения – login, email, pass. Их мы уже получим от пользователя (описали выше).

```
else{
valuser = User(login, email, pass)
```

В ИТОГЕ:

```
packagecom.example.lb1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
```

```
class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)

valuserLogin: EditText = findViewById(R.id.user_login)
valuserEmail: EditText = findViewById(R.id.user_email)
valuserPass: EditText = findViewById(R.id.user_pass)
valbutton: Button = findViewById(R.id.button_reg)

button.setOnClickListener{
vallogin = userLogin.text.toString().trim()
valemail = userEmail.text.toString().trim()
valpass = userPass.text.toString().trim()

if(login == "" || email == "" || pass == "")
Toast.makeText(this, "Невсеполязаполнены", Toast.LENGTH_LONG).show()
else {
valuser = User(login, email, pass)

            }
}
}
}
```

Данные после регистрации можно хранить в файлах или в базе данных. Рассмотрим базу данных SQLite.

1. Создаем новый класс DbHelper

В этом классе опишем подключение к базе данных и некоторые методы, чтобы сама база данных создавалась – в ней создавались таблицы, пользователи и т.д.

Опишем конструктор, который будет создавать переменные и устанавливать значения по этим переменным.

Создадим переменную Context. Поставив курсор на Context нажмите Alt+Enter для импорта.

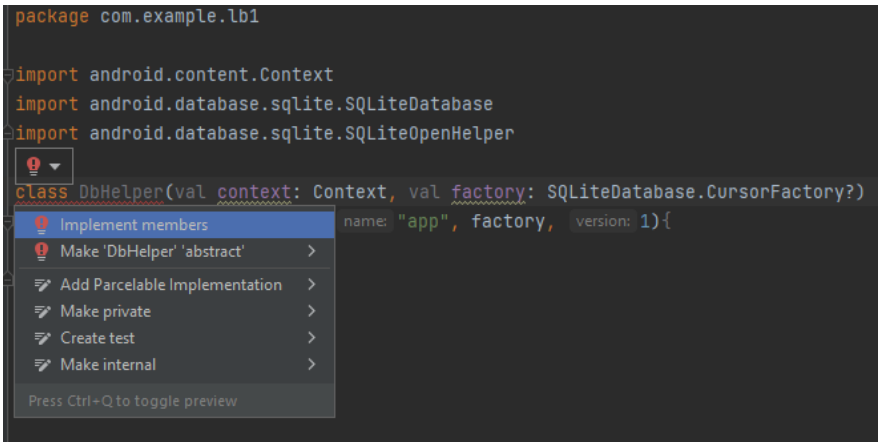
Также будем принимать переменную factory, которая будет основана на классе SQLiteDatabase. Также обращаемся ко встроенному классу CursorFactory? (в конце ставим знак вопрос, т.е. в эту переменную factory мы сможем принимать значения null и чтобы корректно обрабатывать это пустое значение, необходимо поставить ЗНАК ВОПРОС.

Через двоеточие указываем то что будем обращаться к классу SQLite

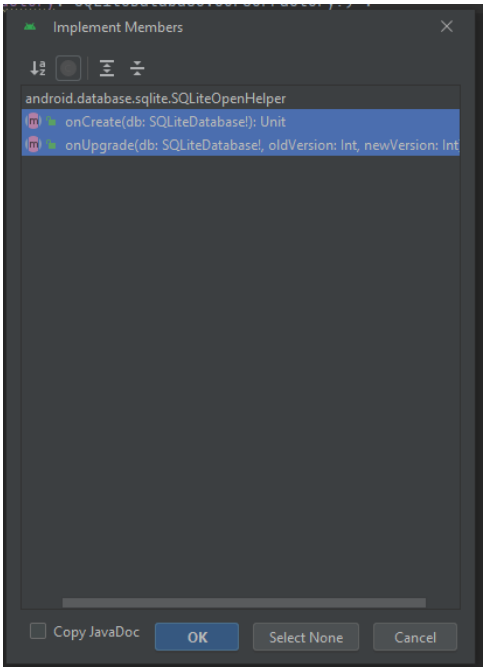
```
class DbHelper(val context: Context, val factory:
SQLiteDatabase.CursorFactory?) :
SQLiteOpenHelper(context, "app", factory, 1){

    }
```

Таким образом создали класс DbHelper в котором принимаем 2-а параметра context и factory. Эти 2-а параметра передаем в класс «родитель» – SQLiteOpenHelper. И указываем название базы данных – "app" и версию базы данных.

Созданный класс подчеркнут красной линией – нам надо реализовать некоторые методы SQLiteOpenHelper здесь, в классе наследнике. Для этого жмем на ошибку – красная лампочка и выбираем:



Выбираем оба метода и жмем Enter



Получаем два новых добавленных метода – onCreate и onUpgrade:

```
packagecom.example.lb1

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DbHelper(valcontext: Context, valfactory:
SQLiteDatabase.CursorFactory?) :
SQLiteOpenHelper(context, "app", factory, 1){
override fun onCreate(db: SQLiteDatabase?) {
TODO("Not yet implemented")
    }

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {
TODO("Not yet implemented")
    }

}
```

В методе onCreate будем создавать всю базу данных, а в методе onUpgrade будем выполнять удаление всей базы данных, а также повторное ее создание.

В методах удаляем «TODO("Not yet implemented")».

В методе onCreate создаем переменную query в которой будем хранить SQL-команду которую будем позже выполнять:

```
val query = "CREATE TABLE user (id INTEGER PRIMARY KEY, login TEXT,
email TEXT, pass TEXT)"
```

Команда для создания таблицы с именем user, в которой будут поля: id с типом INT–целое, также это первичный ключ – PRIMARY KEY; поле login с типом данных TEXT и другие.

В итоге класс:

```
package com.example.lb1

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DbHelper (val context: Context, val factory:
SQLiteDatabase.CursorFactory?) : SQLiteOpenHelper (context, "app",
factory, 1){
    override fun onCreate(db: SQLiteDatabase?) {
        val query = "CREATE TABLE user (id INTEGER PRIMARY KEY, login
TEXT, email TEXT, pass TEXT)"
    }
    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int,
newVersion: Int) {
    }
}
```

Для создания запроса обратимся к параметру db, используя функцию execSQL() – будем передавать нашу SQL-команду, которая записана в переменной query:

```
db!!.execSQL(query)
```

Два !! необходимы для корректной обработки возможных значений null.

За счет SQL-команды будет создана таблица с 4-мя полями: id, login, email, pass.

В итоге:

```
packagecom.example.lb1

import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DbHelper(valcontext: Context, valfactory:
SQLiteDatabase.CursorFactory?) :
SQLiteOpenHelper(context, "app", factory, 1){
override fun onCreate(db: SQLiteDatabase?) {
valquery = "CREATE TABLE user (id INTEGER PRIMARY KEY, login TEXT, email
TEXT, pass TEXT)"
db!!.execSQL(query)
    }

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {

    }
}
```

Очищаем базу данных и заново ее создаем:

```
override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {
```

```
db!!.execSQL("DROP TABLE IF EXISTS users")
onCreate(db)
}
```

В этом же классе напишем функцию, которая будет принимать 1 объект на основе класса User. И за счет этого объекта будем регистрировать пользователя в базе данных:

```
fun addUser(user: User){
val values = ContentValues()
values.put("login", user.login)
values.put("email", user.email)
values.put("pass", user.pass)

val db = this.writableDatabase
db.insert("user", null, values)

db.close()
}
```

Здесь: создан объект values, который создан на основе класса ContentValues()

Обращаемся к values, к функции put, указываем что по ключу "login" будем подставлять значение user.login, т.е. то что получаем из нашего объекта.

В схожем ключе будем подставлять user.email и user.pass

Val db = this.writableDatabase – это выполнение SQL-команды. Т.е. создается объект «db», обращаемся к текущей базе данных, как к базе в которую можно что-то записать – this.writableDatabase

Далее обращаемся к «db», к команде insert() – она позволяет что-то добавить. В скобках указываем что добавляем в таблицу «user», указываем, что сдвига по колонкам не будет (null), и в качестве значений которые будем подставлять указываем наш объект values.

db.close() –закрываем базу данных.

В итое:

```
packagecom.example.lb1

import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DbHelper(valcontext: Context, valfactory:
SQLiteDatabase.CursorFactory?) :
SQLiteOpenHelper(context, "app", factory, 1){
override fun onCreate(db: SQLiteDatabase?) {
valquery = "CREATE TABLE user (id INTEGER PRIMARY KEY, login TEXT, email
TEXT, pass TEXT)"
db!!.execSQL(query)
    }

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {
db!!.execSQL("DROP TABLE IF EXISTS users")
onCreate(db)
```

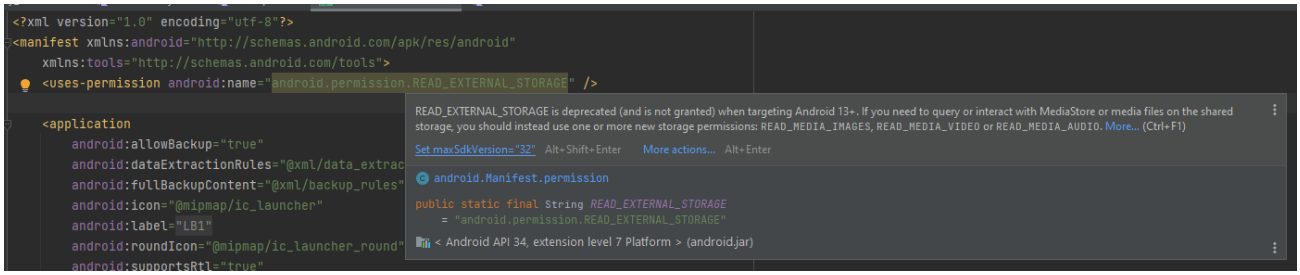
```
    }

fun addUser(user: User){
val values = ContentValues()
values.put("login", user.login)
values.put("email", user.email)
values.put("pass", user.pass)

val db = this.writableDatabase
db.insert("user", null, values)

db.close()
    }
}
```

В файле Manifest

Добавляем одно дополнительное разрешение

```
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Наводим на подсвечиваемую строку XXXXX и выбираем SetmaxSdkVersion=”32”:



В итоге:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools">
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
android:maxSdkVersion="32" />

<application
android:allowBackup="true"
android:dataExtractionRules="@xml/data_extraction_rules"
android:fullBackupContent="@xml/backup_rules"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/Theme.LB1"
tools:targetApi="31">
<activity
android:name=".MainActivity"
android:exported="true">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
```

```
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<meta-data
android:name="preloaded_fonts"
android:resource="@array/preloaded_fonts" />
</application>

</manifest>
```

Возвращаемся в MainActivity

Создаем новый объект на основе класса DbHelperв ветви else. В этот класс передаем контекст с которым мы работаем, а также могли бы передать базу данных с которой работаем, но указываем null, т.к. база данных будет создана.

```
valdb = DbHelper(this, null)
```

Далее обращаемся к функции addUser() и передаем в нее объект на основе класса user.

```
db.addUser(user)
```

Также создаем подсказку «Пользователь $login добавлен»:

```
Toast.makeText(this, "Пользователь $login добавлен",
Toast.LENGTH_LONG).show()
```

Также обращаемся к полям и очищаем их – userLogin, userEmail, userPass:

```
userLogin.text.clear()
userEmail.text.clear()
userPass.text.clear()
```

ИТОГ:

```
packagecom.example.lb1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast

class MainActivity : AppCompatActivity() {
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
```

```
setContentView(R.layout.activity_main)

valuserLogin: EditText = findViewById(R.id.user_login)
valuserEmail: EditText = findViewById(R.id.user_email)
valuserPass: EditText = findViewById(R.id.user_pass)
valbutton: Button = findViewById(R.id.button_reg)

button.setOnClickListener{
vallogin = userLogin.text.toString().trim()
valemail = userEmail.text.toString().trim()
valpass = userPass.text.toString().trim()

if(login == "" || email == "" || pass == "")
Toast.makeText(this, "Невсеполязаполнены", Toast.LENGTH_LONG).show()
else {
valuser = User(login, email, pass)

valdb = DbHelper(this, null)
db.addUser(user)
Toast.makeText(this, "Пользователь$loginдобавлен",
Toast.LENGTH_LONG).show()

userLogin.text.clear()
userEmail.text.clear()
userPass.text.clear()
            }
}
}
```