1 Utworzenie nowego projektu

1. W Android Studio: Start a new Android Studio project
2. W oknie “New Project” wprowadź:
   - Application: “My First App”
   - Company Domain: “example.com”
3. Click Next
4. W oknie: Target Android Devices zostaw domyślne wartości. Click Next
   - Pole: Minimal Required SDK - wskazuje najniższą możliwą wersję androida na której będzie działać aplikacja. Aby wspierać jak najwięcej urządzeń, powinieneś wybrać najniższą wersję, która zapewni Twojej aplikacji wszystkie niezbędne funkcjonalności.
5. W oknie: Add an Activity to Mobile wybierz: Empty Activity. Click Next
6. W oknie: Customize the Activity zostaw domyślne wartości. Click Finish
7. Po utworzeniu projektu, Android studio otworzy i wyświetli aplikację z napisem: Hello World.

2 Review projektu

1. upewnij się, czy okno Project jest otwarte (wybierz View >Tool Windows >Project) i czy okno Android jest wybrane z listy rozwijanej.

2. app >java >com.example.myfirstapp >MainActivity.java

   - ten plik się pojawia, zaraz po utworzeniu projektu. Zawiera definicje klasy dla activity utworzonego wcześniej. Po zbudowaniu i uruchomieniu aplikacji, powyższe Activity wczytuje plik layout, który wyświetla tekst: "Hello world!"

3. app >res >layout >activity_main.xml

   - plik XML definiuje wygląd activity. Zawiera element TextView z napisem: "Hello world"

4. app >manifest >AndroidManifest.xml

   - Opisuje podstawową charakterystykę aplikacji i definiuje każdy z komponentów.

5. Gradle Script >build.gradle

   - Andoid Studio wykorzystuje Gradle do kompilowania i budowania aplikacji. Jest plik build.gradle w każdym module projektu, jak również jest build.gradle dla całego projektu. W początkowej fazie przygody z Androidem, zainteresować Ciebie pliki build.gradle dla modułu.

3 Android Virtual Device (AVD)

1. Przed uruchomieniem aplikacji, potrzebujesz stworzyć Android Virtual Device (AVD). Opisuje charakterystykę telefonu, tabletu, zegarka, czy telewizora z systemem android, którego chcesz zasymulować z Andoid Emulator.

2. Uruchom AVD Manager: Tools >Android >AVD Manager.

3. Na ekranie: System Hardware wybierz urządzenie, np. Nexus 6. Click Next

4. Na ekranie: System Image wybierz wersję androida na której chcesz uruchamaić apliacje. Click Finish

4 Uruchomienie aplikacji

1. W Android Studio, Wybierz swój projekt i nacisnij przycisk Run

2. W oknie Select Deployment Target wybierz swój emulator i naciśnij OK.

3. uruchomienie emulatora może zająć kilka minut.

5 Tworzenie prostego UI

Utworzymy proste okno, zawierające pole do wprowadzania tekstu oraz dwa przyciski: Save i Show

1. Otwórz app >res >layout >activity_main.xml

   - Domyślnie otwiera się w widoku: Layout Editor. Na potrzeby projektu, zmień na Text na dole programu

2. Usuń całą zawartość i wklej ponizszy XML:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <EditText android:id="@+id/message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="message" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Save" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show" />
</LinearLayout>
```

   - **Linear Layout** - grupa widoku, która rozkłada elementy w pozycji pionowej lub poziomej. Odpowiedzialny jest za to atrybut: android:orientation. każdy element na ekranie pojawia się w takiej samej kolejności, w jakiej zdefiniwany jest w pliku xml. Dwa pozostałe atrybuty: **android:layout_width** i **android:layout_heigh** są obowiązkowe dla każdego elementu, służą do definiowania rozmiaru.

3. Punk kontrolny - uruchom aplikację, powinno wyświetlić się okno z elementami zdefiniowanymi w pliku: activity_main.xml

6 Przechodzenie do kolejnego Activity(okna) po naciśnięciu przycisku

1. do przycisków zdefiniowanych w pkt. 5, dodaj parametr: android:onClick, który po kliknięciu wywoła metodę z podaną nazwą. Do pierwszego buttona: **android:onClick=saveMessage"** i drugiego: **android:onClick=showMessages"**

2. do pliku: MainActivity.java dodaj metody, które zostaną wywołane po naciśnięciu przycisku:

```
public void saveMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.message);
    String message = editText.getText().toString();
    intent.putExtra("Show", false);
    intent.putExtra("Message", message);
    startActivity(intent);
}

public void showMessages(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    intent.putExtra("Show", true);
    startActivity(intent);
}
```

   - **Intent intent = new Intent(this, DisplayMessageActivity.class);**
     - definiujemy external intent, który po wywołaniu **startActivity(intent);** przejdzie do activity: **DisplayMessageActivity**

3. Dodanie kolejnego activity

   - utwórz klasę: DisplayMessageActivity.java, zawierającą kod:

```
public class DisplayMessageActivity extends AppCompatActivity{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_message);
        Intent intent = getIntent();
        if(intent.getBooleanExtra("Show", true)) {
            message("Save button");
        } else {
            message("Show button");
        }
    }
```

```
    public void message(String message) {
        TextView textView = new TextView(this);
        textView.setTextSize(40);
        textView.setText(message);
        ViewGroup layout = (ViewGroup) findViewById(
             R.id.activity_display_message);
        layout.addView(textView);
    }

}
```

  - **Intent intent = getIntent()** - pobieramy Intent przekazany z poprzedniego activity.
  - **message(Śave button”);** - wyświetlamy informacje w zależności od klikniętego przycisku

- żeby powyższe activity było widoczne, musisz je dodać do pliku AndroidManifest.xml:

```
<activity android:name=".DisplayMessageActivity"></activity>
```

- dodaj layout: **activity_display_message.xml** do katalogu: res >layout, odpowiedzialny za wyświetlanie elementów na ekranie:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_display_message"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.piotrek.myapplication.DisplayMessageActivity">

    <ListView
        android:id="@+id/message_list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

</RelativeLayout>
```

  - **ListView** - element odpowiedzialny za wyświetlanie listy możliwej do przewijania z poziomu aplikacji.

4. Punkt kontrolny - uruchom aplikacje. Po nacisnięciu przycisku, powinno przejść do nowego okna z nazwą wciśniętego buttona.

7 Zapis, odczyt, usuwanie z bazy danych

1. Utwórz Contract class, odpowiedzialną za definiowanie schematu bazy danych. Jest to kontener dla stałych, definiujących URI, tabele i kolumny. Poprzez dziedziczenie po interfejsie: **BaseColumns**, klasa wewnętrzna dziedziczy klucz główny, nazwany: _ID, który jest niezbędny dla niektórych klas androida.

```
public final class FeedReaderContract {

    private static final String SQL_CREATE_ENTRIES =
            "CREATE TABLE " + FeedReaderContract.FeedEntry.TABLE_NAME + " (" +
                    FeedReaderContract.FeedEntry._ID +
                    " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    FeedReaderContract.FeedEntry.COLUMN_NAME_TITLE +
                    " TEXT NOT NULL )";

    private static final String SQL_DELETE_ENTRIES =
            "DROP TABLE IF EXISTS " + FeedReaderContract.FeedEntry.TABLE_NAME;


    public static String getSqlCreateEntries() {
        return SQL_CREATE_ENTRIES;
    }

    public static String getSqlDeleteEntries() {
        return SQL_DELETE_ENTRIES;
    }

    // To prevent someone from accidentally instantiating the contract class,
    // make the constructor private.
    private FeedReaderContract() {}

    /* Inner class that defines the table contents */
    public static class FeedEntry implements BaseColumns {
        public static final String TABLE_NAME = "entry";
        public static final String COLUMN_NAME_TITLE = "title";
    }
}
```

2. utwórz Klase Helper, odpowiedzialną za tworzenie i utrzymywanie bazy danych i tabel.

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "FeedReader.db";
```

```
    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(FeedReaderContract.getSqlCreateEntries());
    }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // This database is only a cache for online data, so its upgrade policy is
        // to simply to discard the data and start over
        db.execSQL(FeedReaderContract.getSqlDeleteEntries());
        onCreate(db);
    }
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }
}
```

3. utwórz layout: **single_message.xml** reprezentujący pojedyńczą wiadomość do wyświetlenia. Zawiera pole tekstowe i przycisk służący do usuwania.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_vertical">
    <TextView
        android:id="@+id/text_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textSize="20sp" />

    <Button
        android:id="@+id/task_delete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:onClick="delete"
        android:text="remove" />
</RelativeLayout>
```

4. zmodyfikuj klasę: DisplayMessageActivity, tak, aby umożliwiała operacje na bazie danych:

```
public class DisplayMessageActivity extends AppCompatActivity {

    private FeedReaderDbHelper feedReaderDbHelper;
    private Intent intent;
    private ListView messageListView;
    ArrayAdapter<String> mAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_message);

        feedReaderDbHelper = new FeedReaderDbHelper(this);
        intent = getIntent();
        messageListView = (ListView) findViewById(R.id.message_list);

        if(intent.getBooleanExtra("Show", true)) {
            displayData();
        } else {
            saveData();
        }

    }

    public void saveData() {

        String message = intent.getStringExtra("Message");

// Gets the data repository in write mode
        SQLiteDatabase db = feedReaderDbHelper.getWritableDatabase();

// Create a new map of values, where column names are the keys
        ContentValues values = new ContentValues();
        values.put(FeedReaderContract.FeedEntry.COLUMN_NAME_TITLE, message);

// Insert the new row, returning the primary key value of the new row
        long newRowId = db.insertWithOnConflict(
         FeedReaderContract.FeedEntry.TABLE_NAME, null, values,
          SQLiteDatabase.CONFLICT_REPLACE);
        db.close();

        successMessage(message);
    }
```

```
public void displayData() {

    feedReaderDbHelper = new FeedReaderDbHelper(this);

    SQLiteDatabase db = feedReaderDbHelper.getReadableDatabase();

    // Define a projection that specifies which columns from the database
    // you will actually use after this query.
    String[] projection = {
            FeedReaderContract.FeedEntry._ID,
            FeedReaderContract.FeedEntry.COLUMN_NAME_TITLE,
    };

    Cursor cursor = db.query(
            FeedReaderContract.FeedEntry.TABLE_NAME,   // The table to query
            projection,                                // The columns to return
            null,                            // The columns for the WHERE clause
            null,                             // The values for the WHERE clause
            null,                                      // don't group the rows
            null,                                   // don't filter by row groups
            null                                   // The sort order
    );

    ArrayList<String> messageList = new ArrayList<>();

    while(cursor.moveToNext()) {
        int idx =
                cursor.getColumnIndex
                 (FeedReaderContract.FeedEntry.COLUMN_NAME_TITLE);

        messageList.add(cursor.getString(idx));
    }

    if (mAdapter == null) {
        mAdapter = new ArrayAdapter<String>(this,
                R.layout.single_message, // what view to use for the items
                R.id.text_message, // where to put the String of data
                messageList); // where to get all the data
        messageListView.setAdapter(mAdapter); // set it as
         //the adapter of the ListView instance
    } else {
        mAdapter.clear();
        mAdapter.addAll(messageList);
        mAdapter.notifyDataSetChanged();
    }
```

```
        cursor.close();
        db.close();
    }

    public void successMessage(String savedMessaged) {

        TextView textView = new TextView(this);
        textView.setTextSize(40);
        textView.setText("Message added to database: " + savedMessaged);

        ViewGroup layout = (ViewGroup) findViewById(R.id.activity_display_message);
        layout.addView(textView);
    }

    public void delete(View view) {
        View parent = (View) view.getParent();
        TextView textView = (TextView) parent.findViewById(R.id.text_message);
        String message = String.valueOf(textView.getText());
        SQLiteDatabase db = feedReaderDbHelper.getWritableDatabase();
        db.delete(FeedReaderContract.FeedEntry.TABLE_NAME,
                FeedReaderContract.FeedEntry.COLUMN_NAME_TITLE + " = ?",
                new String[]{message});
        db.close();

        displayData();
    }
}
```

5. Uruchom aplikacje - powinna wykonywać operacje na bazie danych, takie: dodawanie, usuwanie, wyświetlanie wiadomości.

źródło:
https://developer.android.com/training/basics/firstapp/index.html
https://www.sitepoint.com/starting-android-development-creating-todo-app/