Laboratorium nr 1 – przypomnienie podstawowych komponentów aplikacji Android

1. Konfiguracja środowiska programistycznego dla platformy Android

Skonfiguruj swoje środowisko dla platformy Android:

- Instalacja Android studio: https://developer.android.com/studio/install

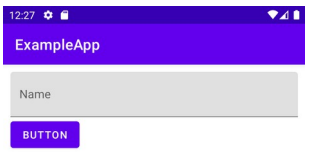
- Przygotowanie środowiska: https://docs.unity3d.com/2018.2/Documentation/Manual/android-sdksetup.html

2. Napisz aplikację, która będzie używała podstawowych komponentów aplikacji Android:

https://developer.android.com/training/basics/firstapp

1. Activity

- Utwórz nowy projekt z pustą aktywnością
- Umieść w aktywności pole do wprowadzania tekstu i przycisk:



- Po kliknięciu na przycisk uruchom drugą aktywność (`UserActivity`), do której w intencji przekażesz wartość pola tekstowego (nazwę użytkownia). Wyświetl tą wartość w drugiej aktywności.

- Kod służacy do uruchomienia i przekazania danych do aktywności HelloActivity:

```
val userName = userNameInput.text.toString()

val intent = Intent(this, HelloActivity::class.java)
intent.putExtra(HelloActivity.USER_NAME_EXTRA, userName)
startActivity(intent)
```

2. Service

https://www.vogella.com/tutorials/AndroidServices/article.html

- Stwórz Service, który po wystartowaniu zaczyna odliczać czas co sekundę i wyświetla tę wartość w Logcat. Powinno być możliwe wystartowanie serwisu kilka razy, każdy z oddzielnym licznikiem (użyj metody `onStartCommand`)
- Pamiętaj, że serwis jest wywoływany w tym samym wątku co wątek, który startuje – użyj wątków, coroutines lub RxJava do odliczania żeby nie zablokować głównego wątku, np.:

```
GlobalScope.launch {
     var number = 0;
     while (!isDestroyed) {
             number++;
             Log.d(TAG, "New number $number");
             delay(1000);
     }
}
```

- W aktywności UserAcitvity dodaj 2 przyciski: jeden do startowania serwisu, drugi do zatrzymania (zatrzymanie może być wszystkich serwisów naraz).
- Do startowania i zatrzymania serwisów możesz uzyc kodu:

```
private fun onStartServiceBtnClick() {
    val intent = Intent(this, SimpleService::class.java)
    startService(intent)
}

private fun onStopServiceBtnClick() {
    val intent = Intent(this, SimpleService::class.java)
    stopService(intent)
}
```

3. Broadcast receiver

https://www.vogella.com/tutorials/AndroidBroadcastReceiver/article.html

- Utwórz klasę BroadcastReceivera `NumberReceiver`, która z intencji odczytuje 2 wartości: nazwę użytkownia i liczbę i wyświetla je w Logcat.

```
override fun onReceive(context: Context?, intent: Intent?) {
        Log.d("NumberReceiver", "Received message")
        val number = intent?.getIntExtra(NUMBER_EXTRA, 0) ?: 0
        val user = intent?.getStringExtra(USER_NAME_EXTRA) ?: ""
}
```

- Zarejestruj broadcast receiver w aktywności `UserActivity` (pamiętaj o wyrejestrowaniu w metodzie `onDestroy`)
- Do utworzonego wczesniej serwisu przy jego starcie dodaj wysyłanie nazwy użytkownika
- Dodaj kod w serwisie, który przy zatrzymaniu serwisu wysyła nazwę uzytkownika i liczbę, na której zatrzymał się licznik do receivera `NumberReceiver`

4. Zapisywanie danych

- Dodaj do aplikacji obsługę zapisu danych w bazie SQLite używając biblioteki Room:

  https://developer.android.com/training/data-storage/room

```
@Dao
interface UserDao {
    @Query("SELECT * FROM user")
    fun getAll(): List<User>

    @Query("SELECT * FROM user")
    fun getAllCursor(): Cursor

    @Insert
    fun insert(user: User)
}
```

```
@Entity
data class User(
     @PrimaryKey(autoGenerate = true) var uid: Int? = null,
     @ColumnInfo(name = "user_name") var userName: String,
     @ColumnInfo(name = "number") var number: Int
)

@Database(entities = [User::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
     abstract fun userDao(): UserDao
}
```

- Zapisz otrzymane w broadcast receiverze nazwę użytkownika i liczbę

```
GlobalScope.launch {
    (context!!.applicationContext as
ExampleApplication).database.userDao().insert(
        User(userName = user, number = number)
    )
}
```

- W aktywności `UserActivity` dodaj przycisk, który odczyta dane zapisane w bazie (wyświetli ilość wierszy lub ich wartości w Logcat)
- Uruchom aplikację na emulatorze lub telefonie z kontem root. W Android Studio wejdź w Device File Explorer i przejdź do folderu aplikacji: `data/data/….twoj.identyfikator.aplikacji.../databases` i skopiuj utworzoną bazę danych na komputer.

5. ContentProvider

- Utwórz w aplikacji `ContentProvider`, który umozliwi odczyt zapisanych danych (zaimplementuj jedynie metodę `query`, która zwróci wszystkie dane)

  https://www.geeksforgeeks.org/content-providers-in-android-with-example/

```
override fun query(
      uri: Uri,
      projection: Array<out String>?,
      selection: String?,
      selectionArgs: Array<out String>?,
      sortOrder: String?
): Cursor? {
    return userDao.getAllCursor()
}
```

`AndroidManifest.xml`

```
<provider
      android:authorities="com.example.exampleapp.provider"
      android:name=".contentprovider.ExampleContentProvider"
      android:exported="true"/>

<queries>
      <package android:name="om.example.exampleapp.provider" />
</queries>
```

- Utwórz drugą aplikację, która wykorzystuje utworzony ContentProvider i wyświetla (może być w Logcat) odczytane dane z pierwszej aplikacji

```
val projection = arrayOf("user_name", "number")

val contentUri: Uri =
Uri.parse("content://com.example.exampleapp.provider/.contentprovider.Exa
mpleContentProvider")

val cursor = contentResolver.query(contentUri, projection, null, null,
null)

if (cursor?.moveToFirst() == true) {
    do {
        Log.d("MainActivity", "Username: ${cursor.getString(0)}, $
{cursor.getString(1)}")
    } while (cursor.moveToNext())
}
```

Na zaliczenie laboratorium nr 1 należy przesłać utworzone projekty:

- spakowane w jednym archiwum wraz ze skopiowanym plikiem bazy danych (wypełnionym przynajmniej dwoma wierszami)
- lub link do repozytorium z kodem i plikiem bazy danych

Termin nadsyłania: 8 marca, godzina rozpoczęcia kolejnych zajęć laboratoryjnych