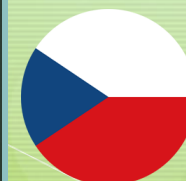


PotraFISZ!

Aplikacja mobilna do tworzenia fiszek do nauki języków europejskich





Platforma Android



NAJPOPULARNIEJSZY
MOBILNY SYSTEM
OPERACYJNY



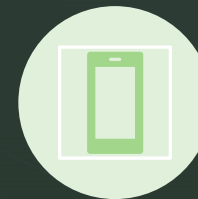
WSPARCIE DLA
DEWELOPERÓW



OTWARTY SYSTEM



DUŻA DOSTĘPNOŚĆ
URZĄDZEŃ



OGROMNA ILOŚĆ
APLIKACJI DO
NABYCIA



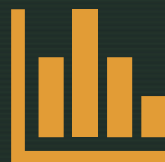
Oprogramowanie
oparte na IntelliJ IDEA



Wspiera starsze
wersje Androida

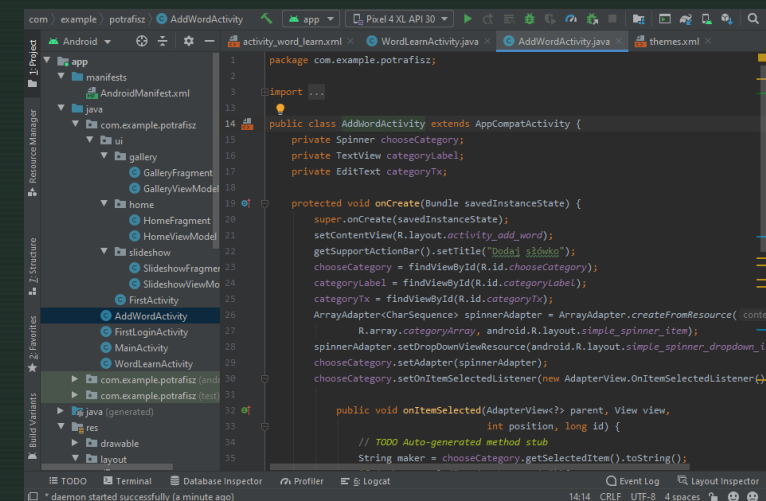


Umożliwia
projektowanie wyglądu
aplikacji



Pozwala na
monitorowanie zużycia
zasobów telefonu

Android Studio

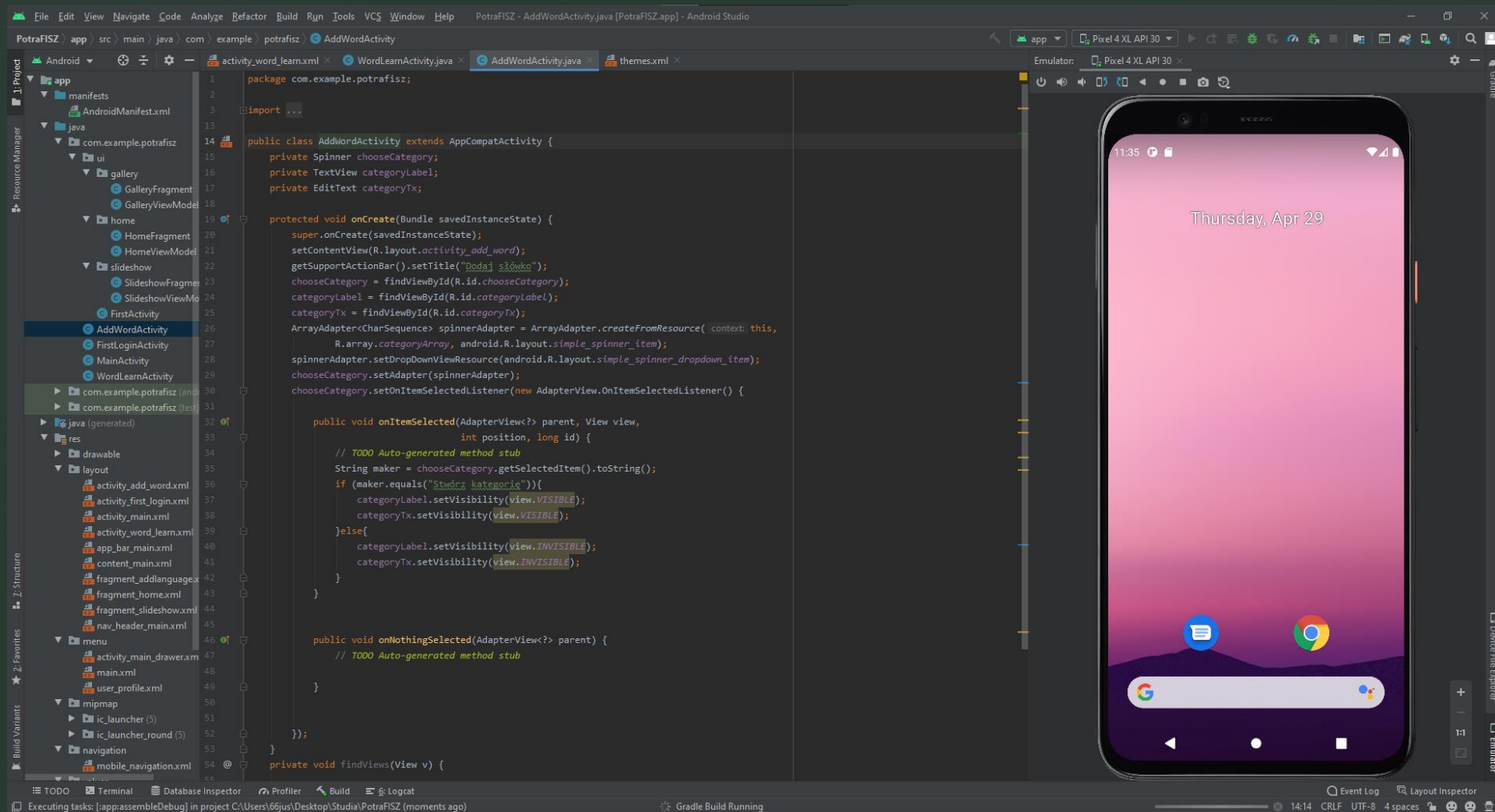


Pora na właściwą aplikację!

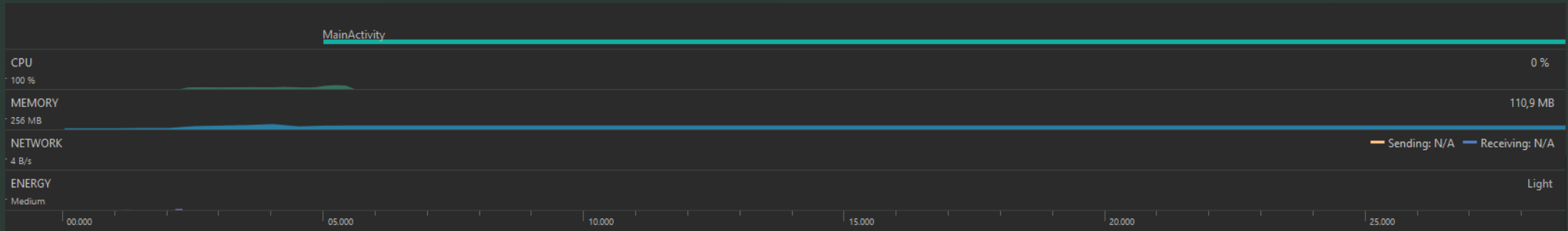
- Użytkownicy Androida: Aby zainstalować aplikację na swoim telefonie należy zeskanować widoczny kod QR i postępować zgodnie z instrukcjami na telefonie.
- Użytkownicy iOS: Należy na komputerze przejść na [tę](#) stronę i poczekać na dane logowania, które prześlą prowadzący. Po zalogowaniu przejść do zakładki *Dashboard*, przy logo aplikacji kliknąć *view*, zmienić model na Pixel 4XL i kliknąć *Tap to play*



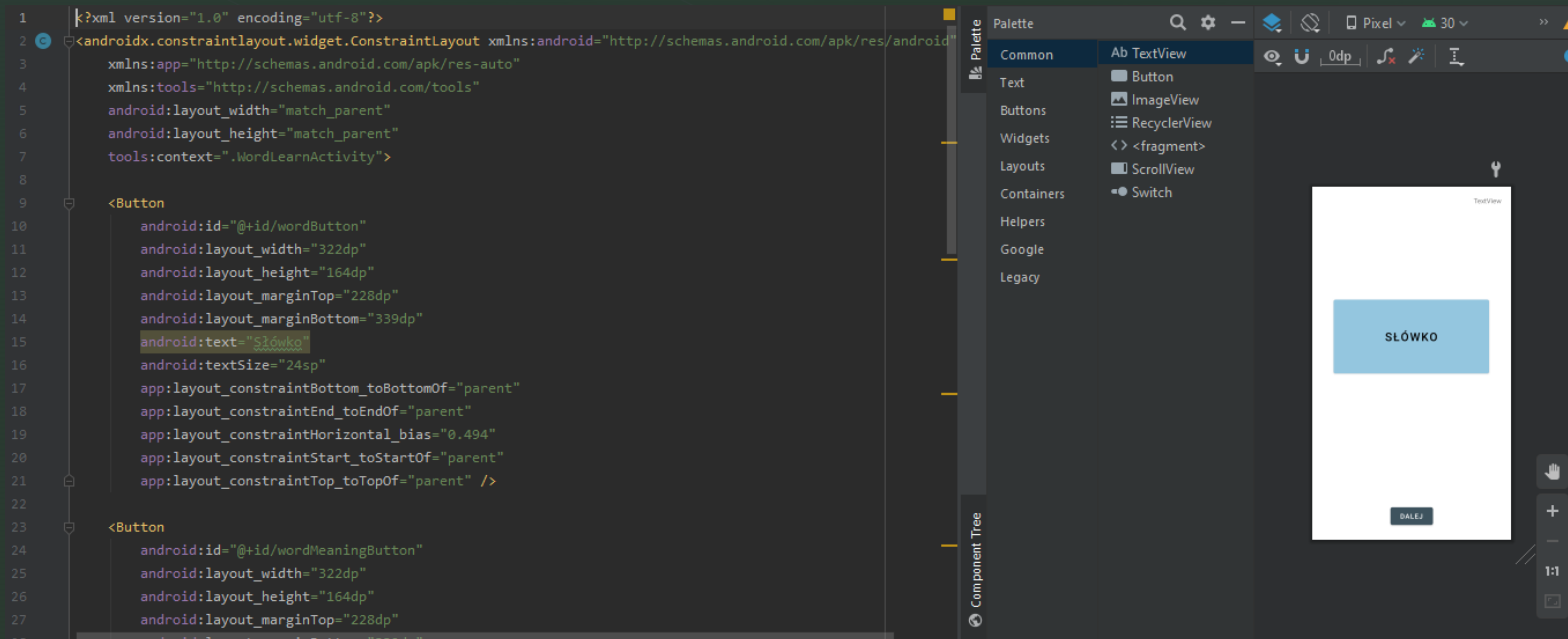
Interfejs aplikacji



Monitor zasobów



Edytor wyglądu aplikacji

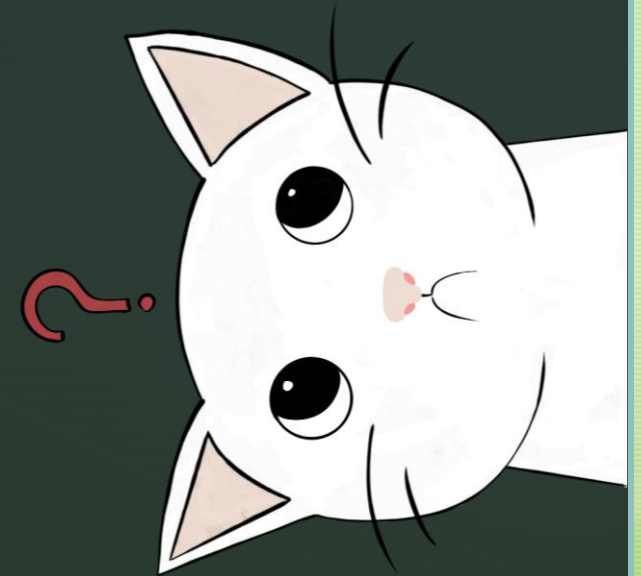


Podstawowe pojęcia – programowanie na Androida

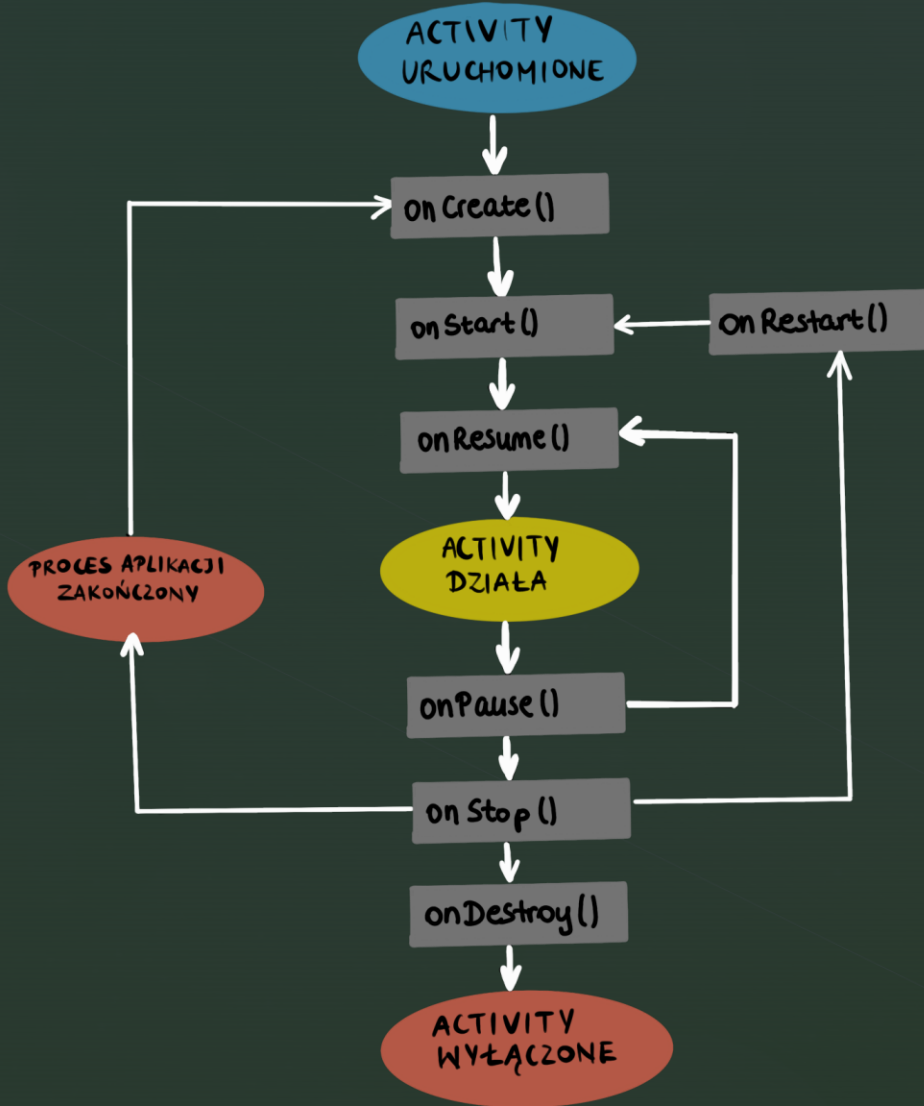
- *Activity* – aktywność, czyli to co widzimy w oknie naszej aplikacji. Klasa odpowiedzialna za interakcję z użytkownikiem.
- *View* – prostokątny fragment widoku ekranu, w którym umieszczane są elementy interaktywne (np. przyciski). Klasa, której zadaniem jest obsługa zdarzeń i wyświetlanie.
- *Event* – zdarzenie, pozwala na pobranie danych od użytkownika, np. poprzez kliknięcie przycisku. Ze zdarzeniami powiązane są dwa pojęcia: *listener* (do „wysłuchania” co użytkownik chce przekazać) i *handler* (do obsłużenia żądania użytkownika).
- *Manifest* – niezbędny plik .xml stanowiący źródło każdego projektu. Zawiera niezbędne informacje dotyczące aplikacji, które są przetwarzane przez system czy narzędzia budujące aplikację.

Pojęć ciąg dalszy...

- *Fragment* – dosłownie, fragment interfejsu użytkownika. Ma własny cykl życia czy zdarzenia, ale musi podlegać jakiejś aktywności.
- *Gradle* – system budujący aplikację oparty na Java Virtual Machine. Umożliwia automatyzację niektórych zadań. Napisany jest w Groove.



CYKL ŻYCIA ACTIVITY



Baza danych, czyli gdzie zapisywane są słówka



Zorganizowany zbiór
informacji w formie
elektronicznej



Do zapisywania danych
wykorzystuje się
najczęściej język SQL



Różnorodność typów baz
danych



,prosta i łatwa w obsłudze baza danych

- Bezserwerowa, relacyjna, lekka baza danych
- Często wybierana jako baza dla aplikacji iOS czy Android
- Opiera się na języku SQL
- Nie potrzebuje wstępnej konfiguracji





Room

- *Room to biblioteka ORM (Object Relational Mapping)*
- *Zapewnia warstwę abstrakcji w stosunku do SQLite*
- Weryfikuje zapytania SQL w czasie kompilacji
- Znacząco zmniejsza ilość kodu „Boilerplate”
- *Komponenty Room: Entity, Dao, Database*

Entity – tabela w bazie danych

```
Dictionary.java x
1  package com.example.potrafisz.data;
2
3  import ...
4
5
6
7  @Entity(tableName = "words_table")
8  public class Dictionary {
9      @PrimaryKey(autoGenerate = true)
10     public int id;
11
12     @ColumnInfo(name = "word")
13     public String word;
14
15     @ColumnInfo(name = "meaning")
16     public String meaning;
17
18     @ColumnInfo(name = "language")
19     public String language;
20
21     @ColumnInfo(name = "category")
22     public String category;
23 }
```


DAO – Definiuje zapytania SQL

```
DictionaryDao.java
12 public interface DictionaryDao {
13     @Insert(onConflict = OnConflictStrategy.IGNORE)
14     void addWord(Dictionary... Dictionaries);
15
16     @Query("SELECT * FROM words_table WHERE language LIKE :language AND category LIKE :category")
17     public abstract List<Dictionary> getAll(String language, String category);
18
19     @Query("SELECT * FROM words_table WHERE language LIKE :language ")
20     public abstract List<Dictionary> getAll(String language);
21
22     @Query("SELECT word FROM words_table")
23     public abstract List<String> getAll();
24
25     @Query("SELECT DISTINCT category FROM words_table WHERE language LIKE :name")
26     List<String> getAllCategory(String name);
27
28     @Insert(onConflict = OnConflictStrategy.IGNORE)
29     void addCategory(CategoryStorage... categories);
30
31     @Query("SELECT DISTINCT category FROM category_storage")
32     List<String> getAllCategories();
33
34     @Insert(onConflict = OnConflictStrategy.IGNORE)
35     void addLanguage(LanguageStorage... languages);
36
37     @Query("SELECT DISTINCT language FROM language_storage")
38     List<String> getAllLanguages();
39 }
```

Podstawowe zapytania SQL

- **INSERT** - Wstawianie wierszy
- **SELECT** - Wybieranie danych
- **DELETE** – Usuwanie danych
- **UPDATE** – Aktualizacja danych
- W Roomie, zapytania tworzymy za pomocą metody bezpośredniej wraz z odpowiednią adnotacją

@Query

- Główna adnotacja używana w klasach DAO. Umożliwia wykonywanie operacji odczytu / zapisu w bazie danych.

```
@Query("SELECT username FROM users_table")  
List<String> getUsername();  
  
@Query("SELECT time FROM users_table")  
List<Long> getTime();  
  
@Query("SELECT procent FROM users_table")  
List<Integer> getPercent();
```

```
@Query("UPDATE users_table SET time = :time WHERE id = :id")  
void update(long time, int id);
```

@Insert

- Kiedy tworzona jest metoda DAO i dodaje się do niej adnotacje @Insert, Room generuje implementację, która wstawia wszystkie parametry do bazy danych w jednej transakcji.

```
@Insert(onConflict = OnConflictStrategy.IGNORE)  
void addWord(Dictionary... Dictionaries);
```

@Delete

- Metoda z adnotacją @Delete usuwa z bazy danych zbiór jednostek podanych jako parametry. Używa kluczy podstawowych, aby znaleźć jednostki do usunięcia.

```
@Delete  
void delete(Dictionary dictionary);
```


Database – główny punkt dostępu

```
DictionaryDatabase.java x
1  package com.example.potrafisz.data;
2
3  import ...
4
5
6
7
8
9
10 @Database(entities = {Dictionary.class, CategoryStorage.class, LanguageStorage.class, User.class}, version = 1, exportSchema = false)
11 public abstract class DictionaryDatabase extends RoomDatabase {
12     public abstract DictionaryDao dictionaryDao();
13
14     public static DictionaryDatabase INSTANCE;
15
16     public static DictionaryDatabase getDbInstance(Context context){
17
18         if (INSTANCE == null){
19             INSTANCE = Room.databaseBuilder(context.getApplicationContext(), DictionaryDatabase.class, "dictionary_database")
20                 .allowMainThreadQueries()
21                 .build();
22         }
23         return INSTANCE;
24     }
25 }
26
```

Implementacja i wykorzystanie

```
def room_version = "2.3.0"  
implementation "androidx.room:room-runtime:$room_version"  
annotationProcessor "androidx.room:room-compiler:$room_version"
```

```
DictionaryDatabase db = DictionaryDatabase.getDbInstance(this.getApplicationContext());  
List<Dictionary> words = db.dictionaryDao().getAll(MainActivity.language, MyCustomListAdapter.globalCategory);  
List<Long> user = db.dictionaryDao().getTime();
```

Zadania

Pytania:

- 1) Co jest potrzebne do poprawnego działania bazy danych w android studio?
- 2) Jaka czynność powoduje wywołanie metody *onPause()*?
- 3) Korzystając ze screena klasy Dictionary, napisz dwa wyrażenia w języku SQL, które:
 - a) zwróćą znaczenia słówek, których kategoria to "Praca", a język to "Rosyjski"
 - b) zwróćą słówka, których znaczenie to "Odpoczywać", kategoria "Dom", a język "Niemiecki"

- 4) W pliku Spinner.java znajduje się funkcja, która w zależności od wybranej pozycji w spinnerze zapisuje inną kategorię dla słowa w bazie danych. Zmienna *category* wskazuje na spinner, zmienna *newCategory* na TextView. Popraw wyrażenie if, aby została zachowana zamierzona funkcjonalność. Możesz użyć w tym celu aplikacji PotraFisz, aby sprawdzić jak zachowuje się w zależności od spinnera.
- 5) W pliku EditTestResult.java znajduje się klasa obsługująca wynik testu. Na podstawie kodu z zajęć zmień kolor tekstu *percentLabel* na żółty, jeżeli uzyskano wynik poniżej 50% i na fioletowy, jeśli powyżej. Zmień także wyświetlany w polu *catPic* obraz na odpowiednio „confused_cat” i „study_cat”. Dodatkowo, wprowadź uprzednio używane kolory do pliku EditColors.xml (np. wpisując ich odpowiedniki wybrane w color picker).

Odpowiedzi na pytania 1 – 3 należy przysłać w pliku .txt, jako odpowiedź do zadań 4,5 proszę odesłać zmodyfikowane pliki.

Bibliografia

- https://www.youtube.com/watch?v=aS_9RbCyHg&t=4193s&ab_channel=edureka%21
- <http://www.android4devs.pl/2011/07/activity-podstawowe-informacje-cykl-zycia/>
- <https://developer.android.com/reference/android/app/Activity>
- https://www.tutorialspoint.com/android/android_event_handling.htm
- <https://analitik.edu.pl/sqlite-i-python-jak-szybko-i-lekko-zaczac-uzywac-bazy-danych/>
- <https://www.oracle.com/pl/database/what-is-database/>
- <https://www.programistamobile.pl/room/>
- ... oraz wiele tematów na stackoverflow.com