Galeria na androida

Celem projektu jest:

Zaprezentowanie różnicy wydajności ładowania całych plików do widoków w aplikacji, a po wcześniejszym ustaleniu jak duża ich część jest nam rzeczywiście potrzebna vs "Glide"[1].

Pokazanie przykładowego kodu mającego na celu zapisanie ostatnich danych użytkowania ("SharedPreferences"[2]).

Wykonanie projektu

Projekt był wykonany w programie Android Studio

Program ten korzysta z Gradle [3], który jest potrzebny do wygenerowania pliku .apk, który możemy przenieść na urządzenie, aby korzystać z napisanej aplikacji

Podstawowe pojęcia:

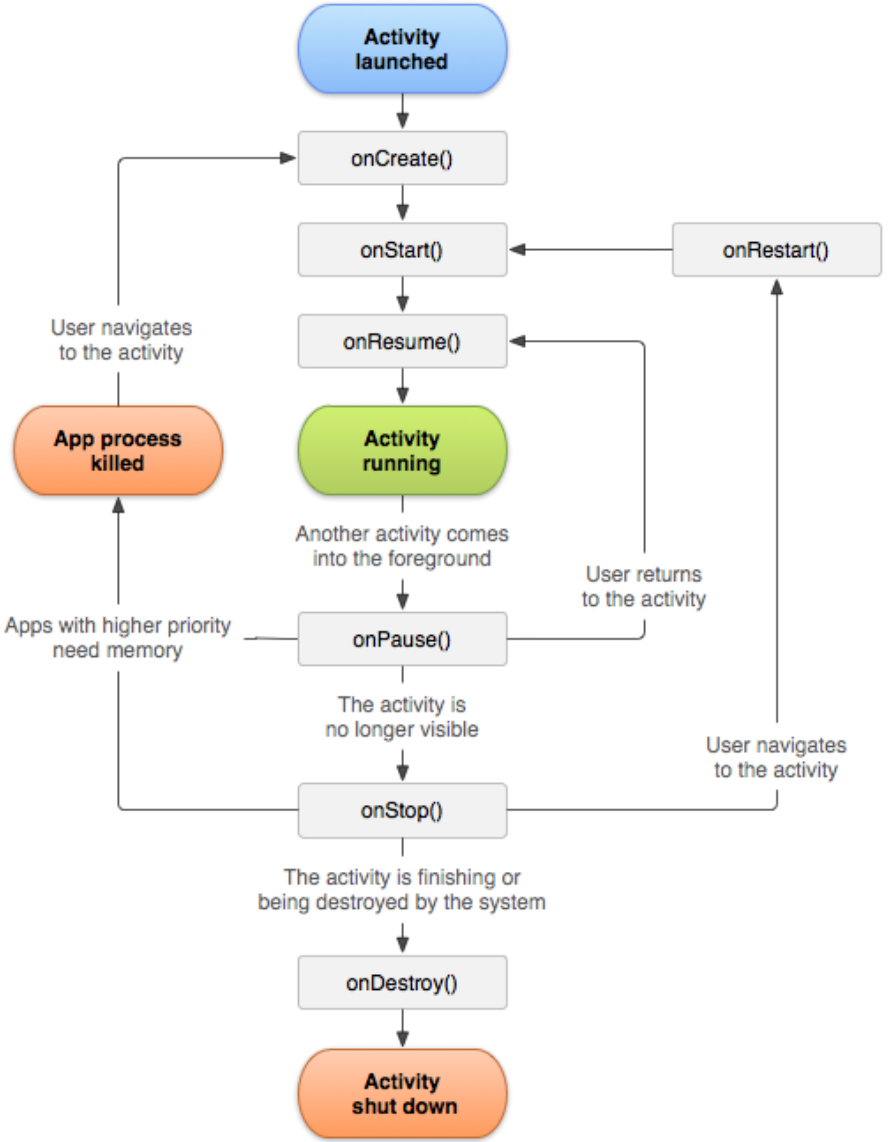
Activity – jest to jeden 'pełny' ekran, który widzi użytkownik

Layout – jest to *rodzaj / typ* ułożenia widoku.

Fragment – jest to część 'activity'. Powinniśmy ją stosować w momencie, gdy chcemy zamienić jedynie część widoku.

Intent – za ich pomocą nie musimy pisać aplikacji 'od zera', ich najczęstszym zastosowaniem jest wykorzystanie funkcjonalności innych aplikacji

Activity lifecycle:

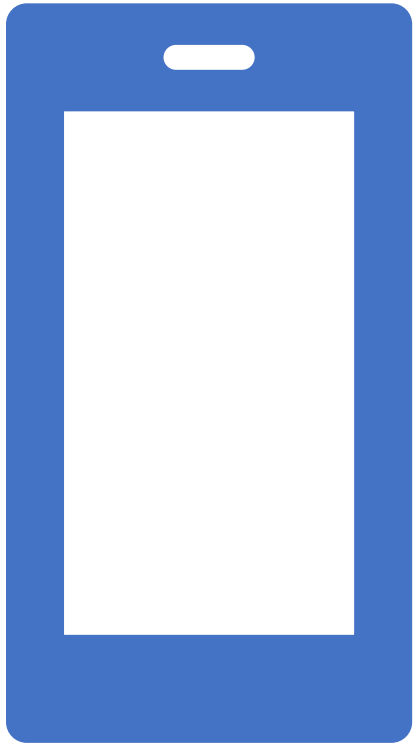


Glide

- Glide jest powiedziałbym bardzo wydajną biblioteką stworzoną do ładowania obrazów i filmów, która dodatkowo daje możliwość wyświetlania gifów, zamiast statycznego obrazu
- Samą bibliotekę implementujemy do Android bardzo szybko poprzez dopisanie do pliku 'build.gradle' poniższego kodu
- Przedstawiony kod dodajemy do pliku 'build.gradle' w folderze 'apk' aplikacji

```
dependencies {

    implementation 'com.github.bumptech.glide:glide:4.12.0'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
```



SharedPreferences

- API służące do zapisu podstawowych typów danych
- Pozwala na odczytanie danych w aplikacji
- Zapisuje się w folderze aplikacji

```
private void getData(){
    SharedPreferences sharedPreferences = getSharedPreferences(dataFilename, MODE_PRIVATE);
    dirColumns = sharedPreferences.getInt( key: "dirColumns", dirColumns: 3);
    imgColumns = sharedPreferences.getInt( key: "imgColumns", imgColumns: 4);
    daysBeforeDeleting = sharedPreferences.getInt( key: "daysBeforeDeleting",getResources().getInteger(R.integer.daysBeforeDeleting));
    matchingPercentage = sharedPreferences.getInt( key: "matchingPercentage",getResources().getInteger(R.integer.matchingPercentage));
    comparingPercentage = sharedPreferences.getInt( key: "comparingPercentage", getResources().getInteger(R.integer.comparingPercentage));

    rotateWithGestures = sharedPreferences.getBoolean( key: "rotateWithGestures",getResources().getBoolean(R.bool.rotateWithGestures));
    deleteAfterEmptying = sharedPreferences.getBoolean( key: "deleteAfterEmptying",getResources().getBoolean(R.bool.deleteAfterEmptying));
    moveToBinBeforeDeleting = sharedPreferences.getBoolean( key: "moveToBinBeforeDeleting",getResources().getBoolean(R.bool.moveToBinBeforeDeleting));
    perfectMatch = sharedPreferences.getBoolean( key: "perfectMatch",getResources().getBoolean(R.bool.perfectMatch));
    fillData();
}
private void saveData() {
    SharedPreferences sharedPreferences = getSharedPreferences(dataFilename, MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putInt("dirColumns", dirColumns); //name:value
    editor.putInt("imgColumns", imgColumns);
    editor.putInt("daysBeforeDeleting", daysBeforeDeleting);
    editor.putInt("matchingPercentage", matchingPercentage);
    editor.putInt("comparingPercentage", comparingPercentage);
    editor.putBoolean("rotateWithGestures",rotateWithGestures);
    editor.putBoolean("deleteAfterEmptying",deleteAfterEmptying);
    editor.putBoolean("moveToBinBeforeDeleting",moveToBinBeforeDeleting);
    editor.putBoolean("perfectMatch", perfectMatch);
    editor.apply();
    super.onBackPressed();
}
```

Zadanie 1 cz.1

- Aby umożliwić w aplikacji sortowanie widoku folderów po ich nazwie, w pliku 'Directory.java' proszę o dodanie dwóch komparatorów (po nazwie i rozmiarze), w tym celu:

- 1. Do klasy Directory dodaj dwa komparatory porównujące zmienne "name" i "size"

Zadanie 1 cz.2

- 2. Zwróć uwagę na to, że zarówno nazwy jak i rozmiary mogą się powtarzać, w tym celu wewnątrz 'ogólnych' komparatorów dodaj dodatkowe warunki, pozwalające jak najdalej rozstrzygnąć 'pierwszeństwo' przy sortowaniu.

- Przykład implementacji:
- https://www.baeldung.com/java-comparator-comparable#1-creating-comparators

Zadanie 2 cz.1

W celu zapisania najprostszych typów danych zastosowałem API "SharedPreferences", którego celem jest przechowanie najprostszych informacji tak, aby przy kolejnym otworzeniu aplikacji nie trzeba było ustawiać wszystkiego od początku.

1. Uzupełnij plik 'starting_data_values.xml' przykładowymi podstawowymi danymi startowymi aplikacji.

Zad2 cz.2

- 2. W pliku SettingsScreen.java w metodzie saveData() zapisz do pliku (zmienna "dataFilename") każdą daną zainicjowaną wcześniej w pliku starting_data_values.xml

- Krótki przykład implementacji:
- https://developer.android.com/training/data-storage/shared-preferences

Bibliografia (teoria):

- [1] https://bumptech.github.io/glide/
- [2] https://developer.android.com/reference/android/content/SharedPreferences
- [3] https://www.geeksforgeeks.org/android-build-gradle/

Linki do stron, z których korzystałem podczas nauki / tworzenia aplikacji

- https://www.baeldung.com/java-comparator-comparable
- https://www.geeksforgeeks.org/gridview-in-android-with-example/
- https://developer.android.com/guide/topics/ui/layout/recyclerview
- https://www.youtube.com/watch?v=AkiltTv0CjA
- https://developer.android.com/training/system-ui/navigation
- https://www.youtube.com/watch?v=xHCsL5QOSv4
- https://www.youtube.com/watch?v=0Tjs_XBF5Pc

Linki do stron z grafikami

- https://developer.android.com/guide/components/activities/activity-lifecycle#alc