Android - podstawy

Katarzyna Łukaszewicz

Podstawowe informacje

- Aplikacje Android są tworzone w języku **Kotlin** lub **Java**
- Kod jest kompilowany do pliku archiwum o rozszerzeniu .apk (*Android Package)*
- Android jest systemem **linuksowym**, w którym każda aplikacja jest innym użytkownikiem
- Każdy proces jest uruchamiany na własnej wirtualnej maszynie (VM), stąd domyślnie aplikacje działają niezależnie od siebie
- System Android wykorzystuje **zasadę minimalnego uprzywilejowania** (*principle of least privilege)* – domyślnie każda aplikacja ma dostęp jedynie do komponentów, które są niezbędne do jej działania

Komponenty aplikacji

- Komponenty stanowią budulec aplikacji, są jej podstawowymi częściami składowymi (app components = app building blocks)
- Komponenty mogą stanowić potencjalny punkt dostępu użytkownika do aplikacji
- Opisane w pliku *AndroidManifest.xml*
- Mogą zachodzić między nimi interakcje

Activity

Activity - definicja

- Jest to pojedynczy ekran z interfejsem użytkownika, za pomocą którego użytkownik może korzystać z funkcjonalności oferowanych przez aplikację
- Odpowiada za określoną funkcjonalność, czynność
- Zazwyczaj jedno z Activities określane jest mianem głównego (*main)* i to ono stanowi ekran startowy aplikacji (wyświetlane jest użytkownikowi w pierwszej kolejności po uruchomieniu aplikacji)
- Activities są przeważnie powiązane ze sobą, aby wspólnie tworzyć spójną całość – aplikację mobilną

Activity – przykłady

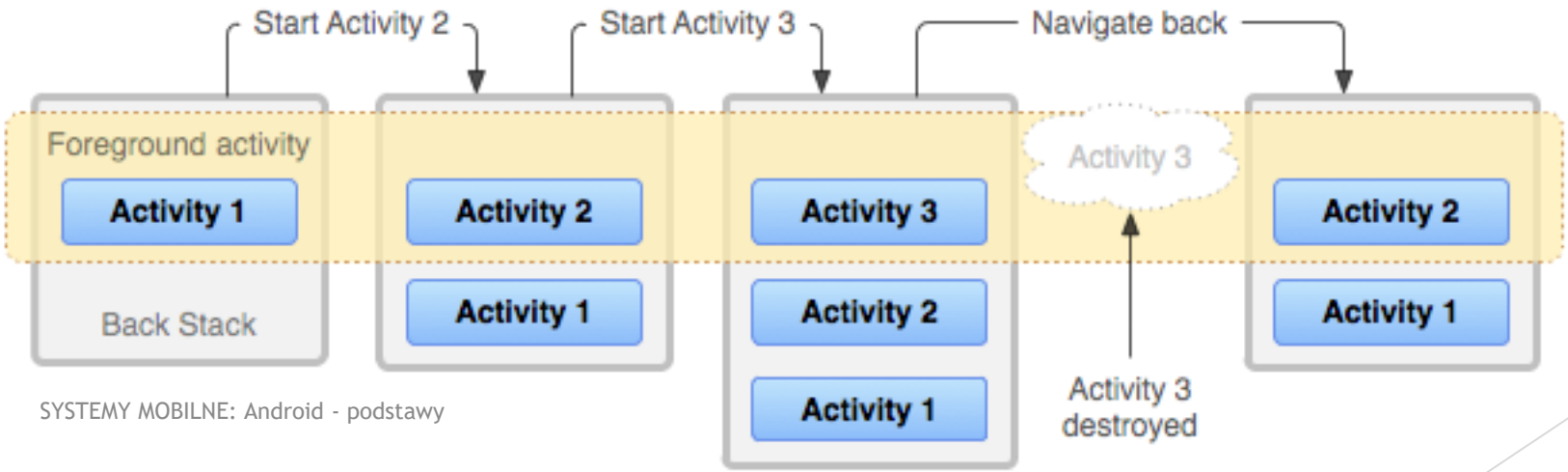
Activity – przykład aplikacji

- Rozpatrujemy aplikację EMAIL
- Możliwy uproszczony podział na Activities:

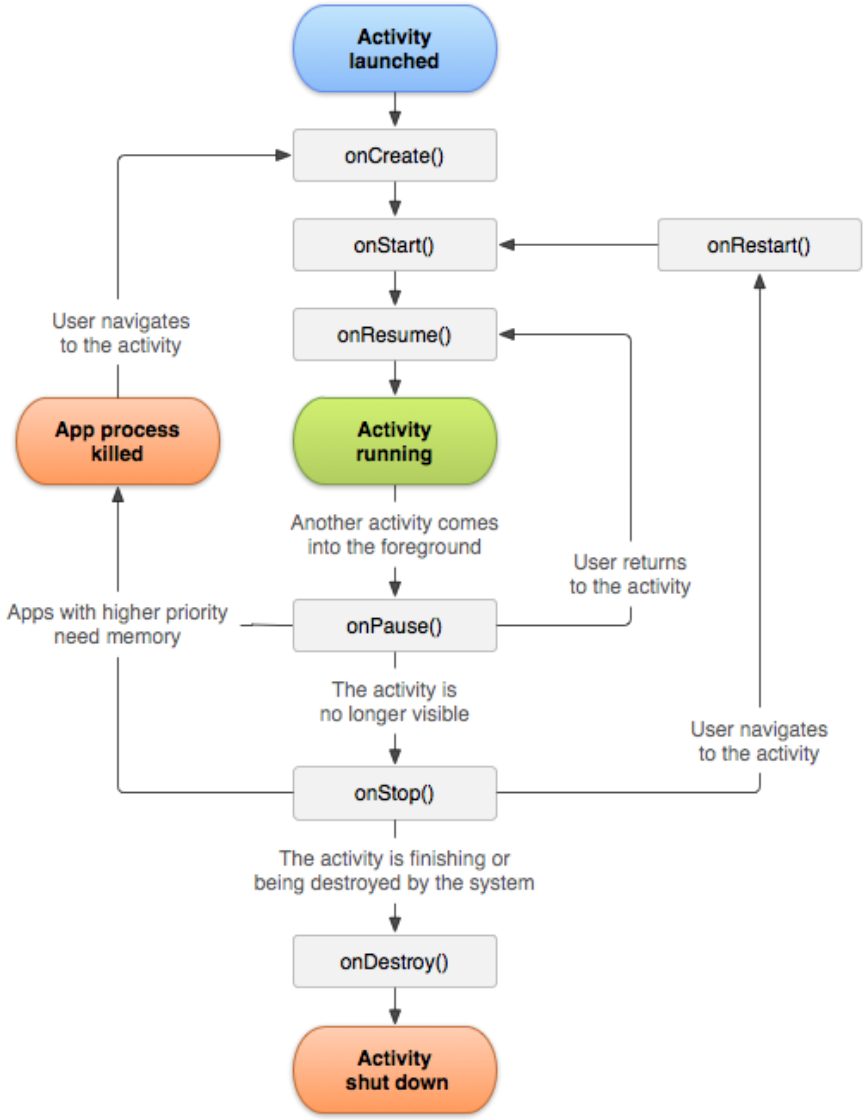
ACTIVITY 1	ACTIVITY 2	ACTIVITY 3
Wyświetlenie listy wiadomości email	Stworzenie nowej wiadomości email	Odczytanie wiadomości email

Activity – jak to działa w aplikacji?

- Po uruchomieniu aplikacji oczom użytkownika ukazuje się *Main Activity*
- Każda z aktywności może uruchomić inną
- Za każdym razem, gdy uruchamiana jest nowa aktywność, poprzednia jest zatrzymywana i trafia na stos LIFO (last in, first out)



Activity – cykl życia



Activity – stany

- Resumed – stan zwany też *running*. Activity jest na przednim planie, na nim skupiona jest uwaga użytkownika
- Paused – stan, w którym Activity wciąż pozostaje widoczne, jednak przesłania je inna aktywność, która w danym momencie koncentruje uwagę użytkownika. Obiekt Activity jest cały czas przechowywany w pamięci, zachowuje stan oraz pozostaje dołączony do window managera. Aktywność w tym stanie może być usunięta przez system w sytuacji ekstremalnego braku pamięci
- Stopped – stan, w którym Activity jest całkowicie przesłonięte przez inne Activity, działa jedynie w tle (wciąż istnieje). Obiekt Activity jest nadal przechowywany w pamięci i zachowuje swój stan, jednak nie jest dostępny w window manager. W każdej chwili może być usunięty przez system (w sytuacji, gdy będzie potrzebna pamięć).

Activity - implementacja

- Jest to klasa dziedzicząca po klasie Activity
- Konieczne jest zaimplementowanie tzw. *callback methods*, czyli metod, które są wywoływane przez system w momencie zmiany stanu danej aktywności
- Dwie najważniejsze callback methods to:
  - onCreate() – obowiązkowa do implementacji. Wywoływana w trakcie tworzenia danej aktywności. Służy przede wszystkim do inicjalizowania najważniejszych komponentów związanych z aktywnością oraz do zdefiniowania układu UI (*layout)* poprzez wywołanie metody setContentView()
  - onPause() – system wywołuje tą metodę, gdy użytkownik opuszcza daną aktywność. Przeważnie są w niej zatwierdzane wszelkie dokonane zmiany, które powinny zostać utrwalone podczas bieżącej sesji

Activity - implementacja

- Każde Activity musi być zadeklarowane w pliku AndroidManifest.xml:

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

- Główne Activity (jedno w aplikacji) powinno mieć dodany przynajmniej następujący filtr:

```
<activity android:name=".ExampleActivity" android:icon="@drawable/app_icon">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Activity - implementacja

- Uruchamianie Activity w podstawowej formie dokonywane jest w następujący sposób:

```
Intent intent = new Intent(this, SignInActivity.class);
startActivity(intent);
```

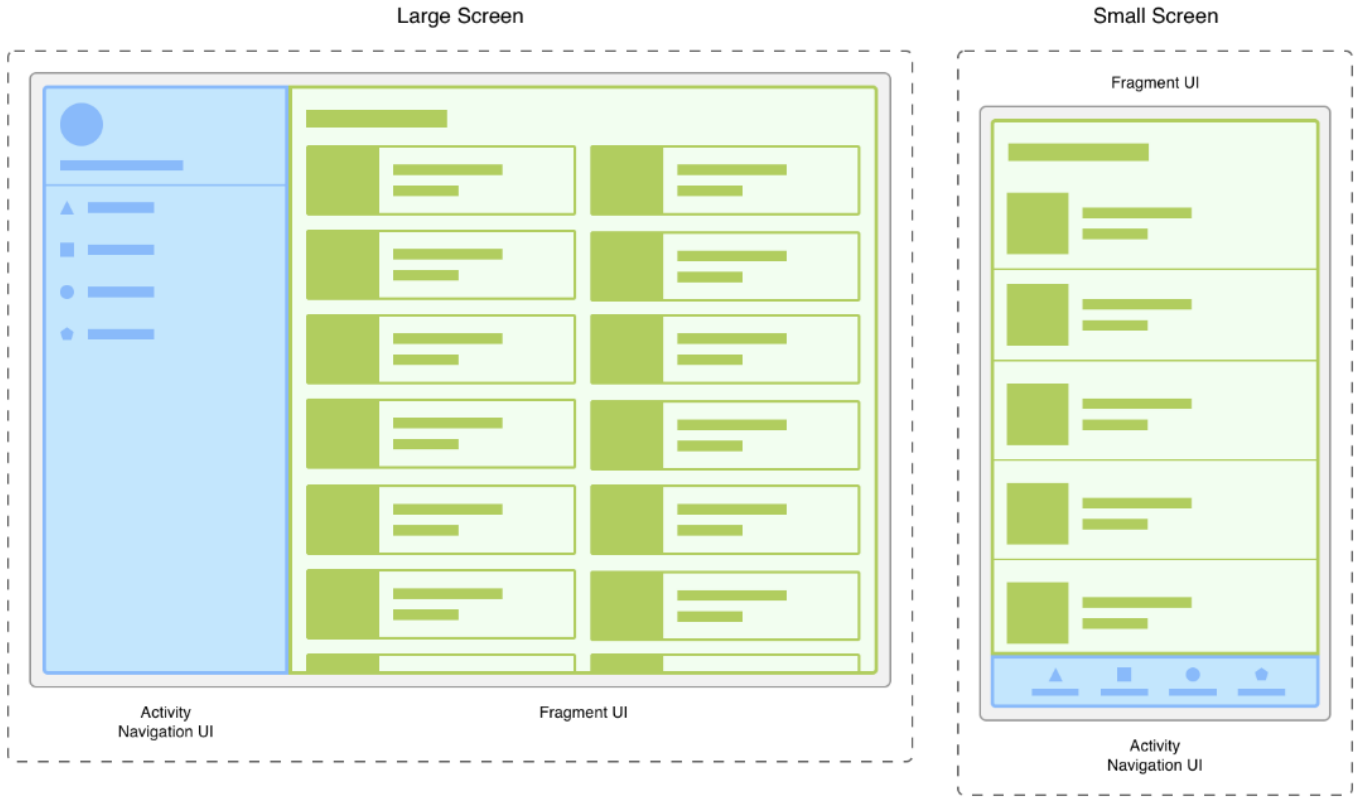
- Możliwe jest też przekazanie uruchamianej aktywności dodatkowych informacji:

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipientArray);
startActivity(intent);
```

Fragment – definicja

- Jest to modularny wycinek Activity, posiadający własny cykl życia, jednak zależny od nadrzędnego Activity
- Fragment nie może istnieć samodzielnie (musi być wbudowany w aktywność)
- Może być dodawany/usuwany z Activity w trakcie działania aplikacji
- Wiele różnych aktywności może korzystać z jednego fragmentu
- Jedno Activity może być współtworzone przez wiele fragmentów
- Służy przede wszystkim do tworzenia bardziej wyszukanego interfejsu użytkownika (przeważnie na większych wyświetlaczach)
- Nie musi być częścią UI aktywności (może jedynie niewidzialnym „pomocnikiem” Activity, do którego oddelegowanych jest część aktywności)

Fragment - przykład



Service

Service - definicja

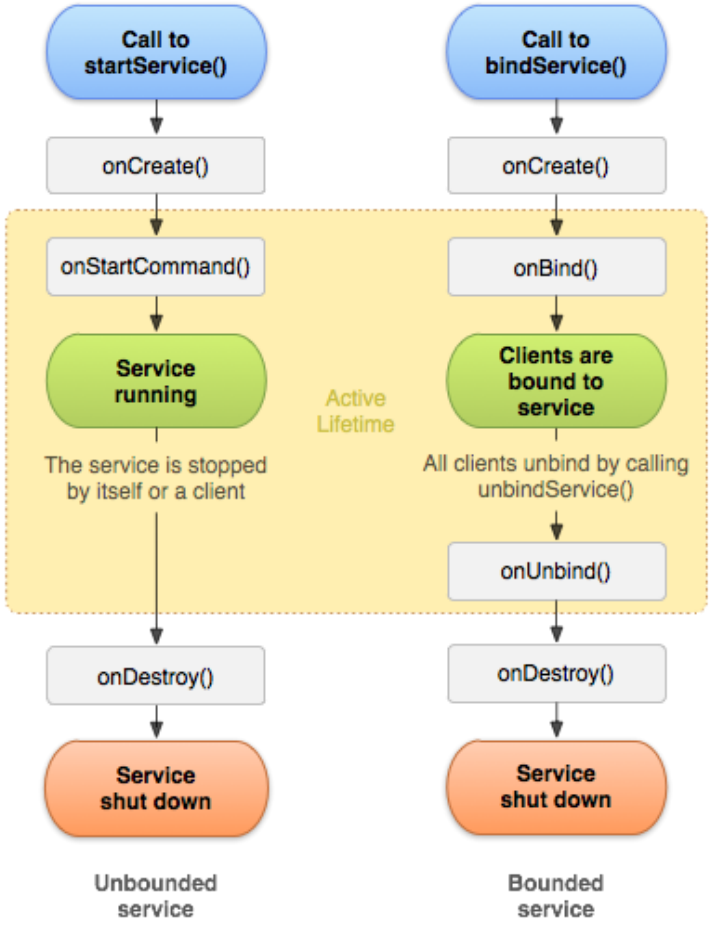
- Komponent, który przeznaczony jest do wykonywania w tle czasochłonnych operacji
- Nie posiada żadnego UI
- Gdy inny komponent uruchomi serwis, działa on w tle nawet, gdy użytkownik przełączy się do innej aplikacji
- Domyślnie serwis działa w głównym wątku procesu, w którym został uruchomiony
- Do bardzo czasochłonnych lub blokujących interakcję z użytkownikiem operacji powinien być tworzony nowy wątek (zredukuje to ryzyko wystąpienia błędu ANR – *Application Not Responding*)

Service - przykłady

Service - rodzaje

STARTED	BOUND
• uruchamiany przez komponent aplikacji za pomocą metody startService() • raz uruchomiony może działać w tle przez nieokreślony czas, nawet jeśli komponent, który go uruchomił zostanie usunięty (destroyed) • przeważnie serwis wykonuje pojedynczą operację i nie zwraca rezultatu komponentowi wywołującemu	• uruchamiany przez komponent aplikacji za pomocą metody bindService() • umożliwiający interakcję z serwisem, wysyłanie żądań oraz pozyskiwanie wyników, • może być powiązany z wieloma komponentami • działa tak długo, jak komponent, z którym jest powiązany (jest usuwany, gdy nie istnieje żaden powiązany komponent)

Service – cykl życia



Service - implementacja

- Klasa dziedzicząca po klasie Service
- Wymagane jest nadpisanie jednej z następujących callback methods:
  - onStartCommand()
  - onBind()
- Podobnie jak Activity, każda instancja Service musi być zadeklarowana w pliku AndroidManifest.xml
- Konieczne jest dodanie wpisu:

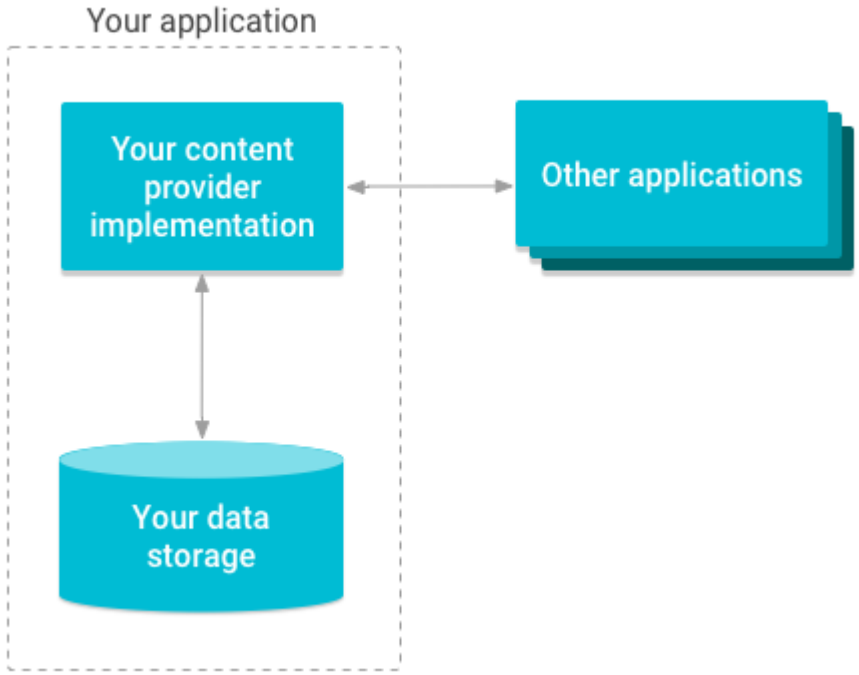
```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

Content Provider

Content Provider - definicja

- Zarządza współdzielonymi danymi w aplikacji
- Za jego pośrednictwem inne aplikacje mogą odczytywać (a niekiedy nawet modyfikować) przechowywane dane
- Content Provider jest potrzebny, gdy:
  - dane mogą albo powinny być dzielone z innymi aplikacjami,
  - dane nie powinny być widoczne dla innych aplikacji, ale konieczne jest utworzenie specjalnych podpowiedzi wyszukiwania albo wymagane jest umożliwienie kopiowania złożonych danych, czy też plików do innych aplikacji
- Przykład: system Android zapewnia obiekt Content Provider, który obsługuje informacje o kontaktach użytkownika → w ten sposób każda aplikacja o odpowiednich uprawnieniach może mieć dostęp do informacji o poszczególnych osobach z książki adresowej

Content Provider - przykład



Broadcast Receiver

Broadcast Receiver

- Komponent odpowiadający za powiadomienia
- Wiele tego typu komponentów jest zaimplementowana oryginalnie w systemie, np.:
  - informacja o wygaszeniu ekranu
  - powiadomienie o niskim poziomie baterii
  - informacja o zrobieniu zdjęcia
- Nie posiada standardowego interfejsu użytkownika. Powiadomienia mogą być wyświetlanie na pasku stanu
- Komponent ten jest implementowany jako podklasa BroadcastReceiver. Każde powiadomienie to obiekt Intent