Android ViewModel oraz wzorzec MVVM

mgr inż. Stanisław Lota

ViewModel

ViewModel to klasa, która jest przeznaczona do przechowywania i zarządzania UI-podobnych danych w cyklu życia kontrolerów interfejsu użytkownika w świadomy sposób. Dzięki temu dane przetrwają zmiany konfiguracji, takie jak obracanie ekranu.

Na przykład, jeśli chcemy wyświetlić listę użytkowników w swojej aplikacji to powinniśmy przypisać odpowiedzialność za pozyskiwanie i przechowywanie listy użytkowników do klasy ViewModel, a nie do aktywności lub fragmentu.

ViewModel

Android może zdecydować o zniszczeniu lub ponownym utworzeniu kontrolera interfejsu użytkownika w odpowiedzi na określone działania użytkownika lub zdarzenia urządzenia, które są całkowicie poza kontrolą użytkownika.

Jeśli system zniszczy lub ponownie utworzy kontroler interfejsu użytkownika, wszelkie przejściowe dane związane z interfejsem użytkownika, które w nim przechowujemy zostaną utracone.

ViewModel

Android Architecture Components udostępnia klasę ViewModel dla kontrolera interfejsu użytkownika, który jest odpowiedzialny za przygotowanie danych dla interfejsu użytkownika. **Dodatkowo zawiera bibliotekę Navigation,która pozwala zapomnieć o stosowaniu intencji podczas nawigacji.**

Obiekty klasy ViewModel są automatycznie zachowywane podczas zmian konfiguracji, dzięki czemu przechowywane w nich dane są natychmiast dostępne dla następnej czynności lub instancji fragmentu.

ViewModel

Innym problemem jest to, że kontrolery interfejsu użytkownika często muszą wykonywać wywołania asynchroniczne, których powrót może zająć trochę czasu. Kontroler interfejsu użytkownika musi zarządzać tymi wywołaniami i upewnić się, że system wyczyści je po ich zniszczeniu, aby uniknąć potencjalnych wycieków pamięci.

To zarządzanie wymaga dużo konserwacji, a w przypadku, gdy obiekt jest ponownie tworzony w celu zmiany konfiguracji, jest to marnowanie zasobów, ponieważ obiekt może być zmuszony do ponownego wysłania wywołań, które już wykonał.

ViewModel

Kontrolery interfejsu użytkownika, takie jak działania i fragmenty, są przeznaczone głównie do wyświetlania danych interfejsu użytkownika, reagowania na akcje użytkownika lub obsługi komunikacji systemu operacyjnego, takiej jak żądania uprawnień.

Wymaganie, aby kontrolery interfejsu użytkownika były również odpowiedzialne za ładowanie danych z bazy danych lub sieci, zwiększa obciążenie klasy i czytelność kodu.

ViewModel

Przypisanie nadmiernej odpowiedzialności do kontrolerów interfejsu użytkownika może spowodować, że pojedyncza klasa będzie próbowała samodzielnie obsłużyć całą pracę aplikacji, zamiast delegować pracę do innych klas.

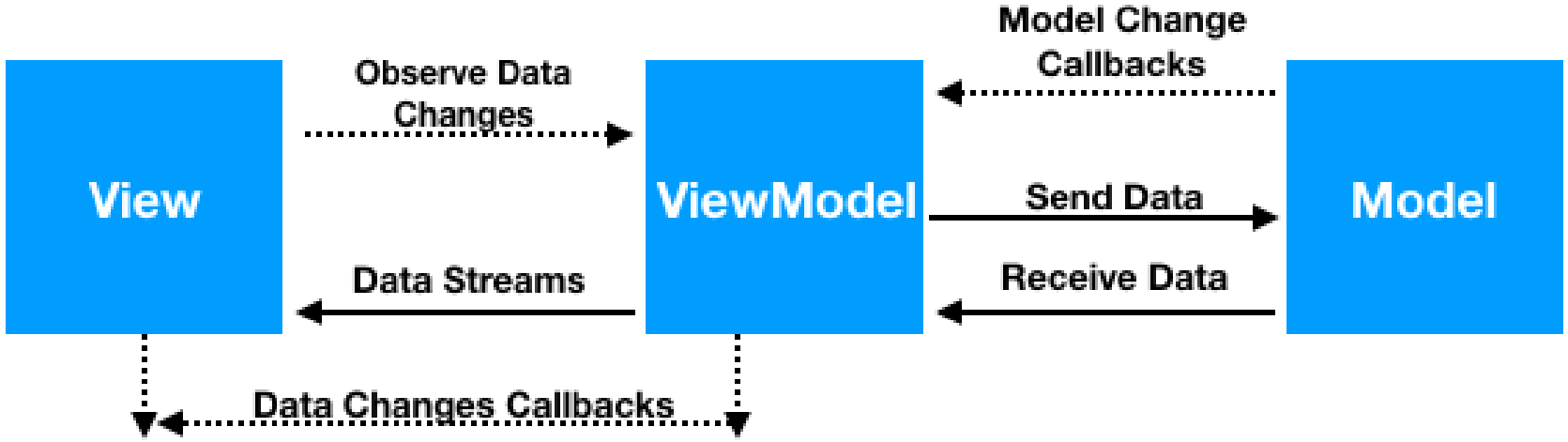
Przypisanie w ten sposób nadmiernej odpowiedzialności kontrolerom interfejsu użytkownika również znacznie utrudnia testowanie.

Łatwiej i wydajniej jest oddzielić własność danych widoku od logiki kontrolera interfejsu użytkownika.

MVVM (Model-View-ViewModel)

Wzorzec architektoniczny MVVM (Model-View-ViewModel) ma za zadanie ułatwić tworzenie ekranów aplikacji poprzez zastosowanie podziału odpowiedzialności na trzy różne warstwy: widoku (View), widoku modelu (ViewModel) oraz modelu (Model).

Google mianował MVVM podstawowym wzorcem w architekturze aplikacji Androidowych.



MVVM (Model-View-ViewModel)

Warstwa widoku odpowiedzialna jest za prezentacje danych, stanu systemu i bieżących operacji w interfejsie graficznym, a także za inicjalizację i wiązanie ViewModel z elementami widoku.

Warstwa widoku modelu zajmuje się dostarczaniem danych modelu dla warstwy widoku oraz podejmowaniem akcji na rzecz wywołanego zdarzenia z widoku.

Natomiast warstwa modelu odpowiada za logikę biznesową, czyli przetwarzanie, przechowywanie, modyfikacje oraz dostarczanie oczekiwanych danych do widoku modelu.

MVVM (Model-View-ViewModel)

Wzorzec MVVM to wzorzec wywodzący się z wzorca MVP zdefiniowanego przez Martina Fowlera w latach 90. XX wieku.

Wzorzec MVVM pomaga całkowicie oddzielić logikę biznesową i prezentacyjną od interfejsu użytkownika, a logikę biznesową i interfejs użytkownika można wyraźnie oddzielić w celu łatwiejszego testowania i łatwiejszej konserwacji.

MVVM (Model-View-ViewModel)

Dzięki zastosowaniu strategii wiązania danych (data binding) w warstwie widoku minimalizowana jest jego logika, kod staje się bardziej uporządkowany i otwarty na modyfikacje, a przeprowadzenie testów łatwiejsze.

Idea wzorca MVVM opiera się przede wszystkim na obserwowaniu przez warstwę widoku zmieniających się danych w warstwie widoku modelu i reagowanie na zmiany poprzez mechanizm wiązania danych.

Ze względu na różnorodność środowisk i technologii realizacja wzorca MVVM może zostać uzyskana na wiele sposobów.

Implementacja

Warstwa widoku View tworzy instancje widoku modelu ViewModel oraz wiąże go z widokiem za pomocą mechanizmu data binding.

Klasa ViewModel przechowuje wszystkie niezbędne dane dla warstwy widoku, które otrzymuje z modelu.

Dzięki zastosowaniu wzorca Obserwator na polach widoku modelu każda zmiana stanu jest odnotowana przez widok.

Warstwa modelu Model dostarcza implementacje pobierania, modyfikacji i przetwarzania danych z repozytoriów.

Zadanie

1. https://developer.android.com/codelabs/kotlin-android-training-view-model?index=..%2F..android-kotlin-fundamentals#0
2. https://developer.android.com/codelabs/kotlin-android-training-live-data#13
3. https://developer.android.com/codelabs/kotlin-android-training-live-data-data-binding#9
4. https://developer.android.com/codelabs/kotlin-android-training-live-data-transformations#0

Jest to jedno zadanie, które składa się z 4 części. Dzięki niemu poznasz zasady tworzenia aplikacji w modelu MVVM.