

MVVM



android 

Kamil Ptak
Sławomir Szulik
Krzysztof Terelak

Architektura aplikacji mobilnych

Coraz większa liczba programistów zdaje sobie sprawę, jak bardzo istotne jest stosowanie architektury. Jednak oficjalna dokumentacja nie narzuca konkretnego rozwiązania.

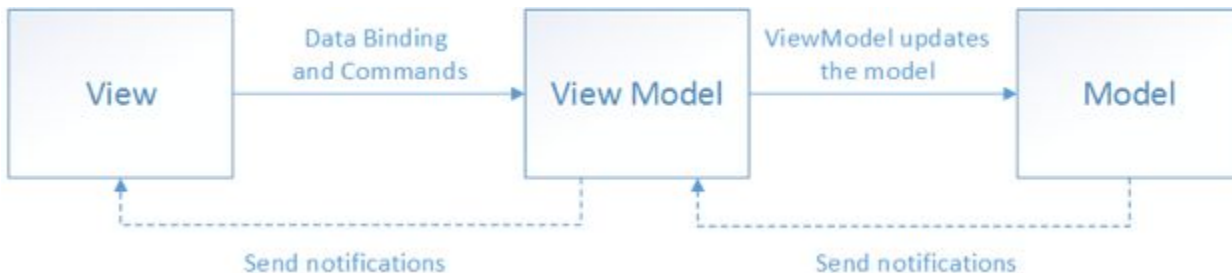
Szukając informacji w sieci można wskazać trzy bardzo popularne podejścia "klasyczne"

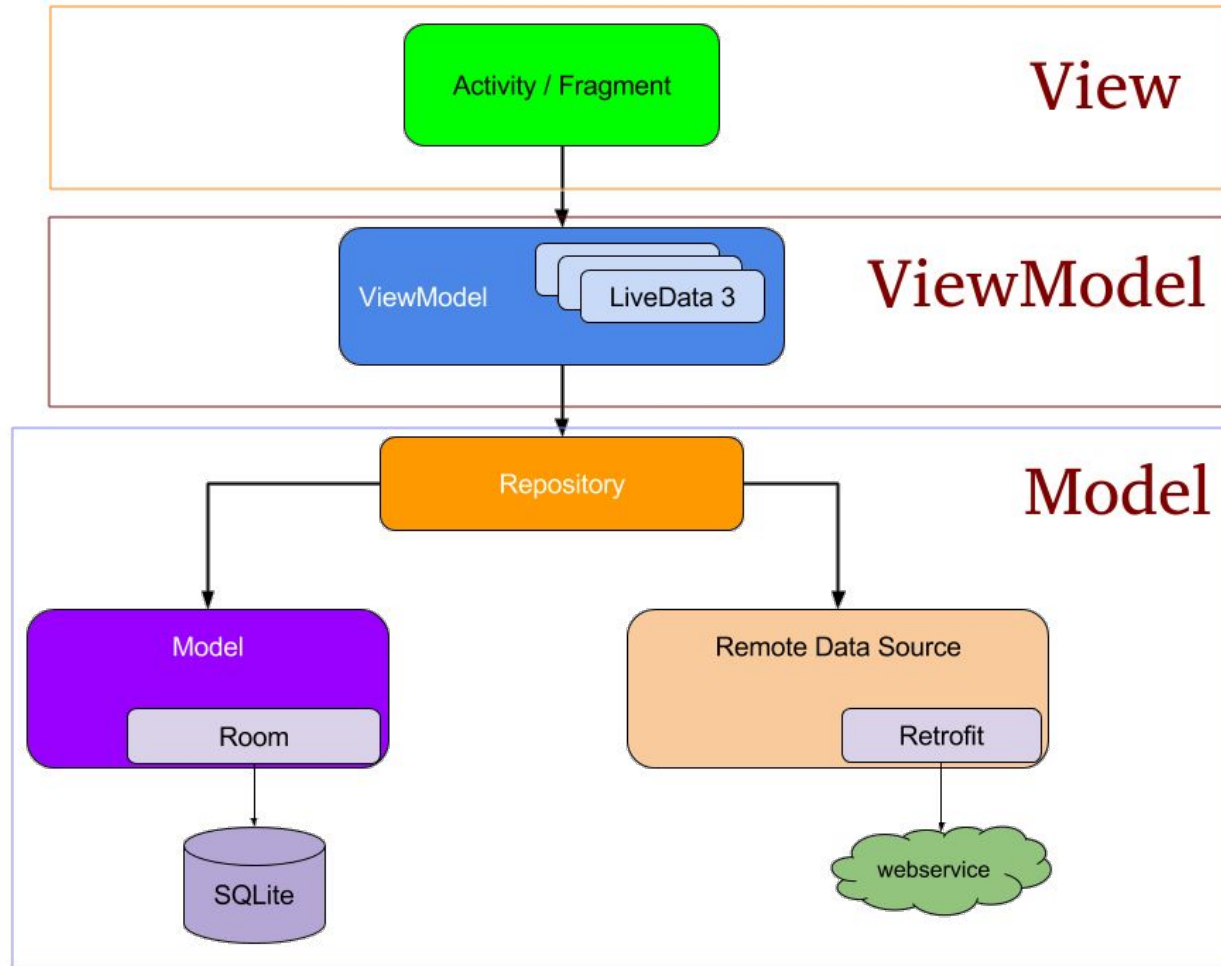
- MVP
- MVVM
- MVI

Często są one realizowane z udziałem dodatkowych bibliotek np. Dagger (Dependency Injection)

Co to jest MVVM

Jego pełna nazwa brzmi Model – View – ViewModel. Całość opiera się na wydzieleniu trzech logicznych warstw w systemie. Chodzi o solidny podział odpowiedzialności, czyli pomiędzy klasami. Co w konsekwencji prowadzi do tego, że każda warstwa ma swoje ograniczenia. I te właśnie ograniczenia pozwalają nam na pisanie lepszego kodu.



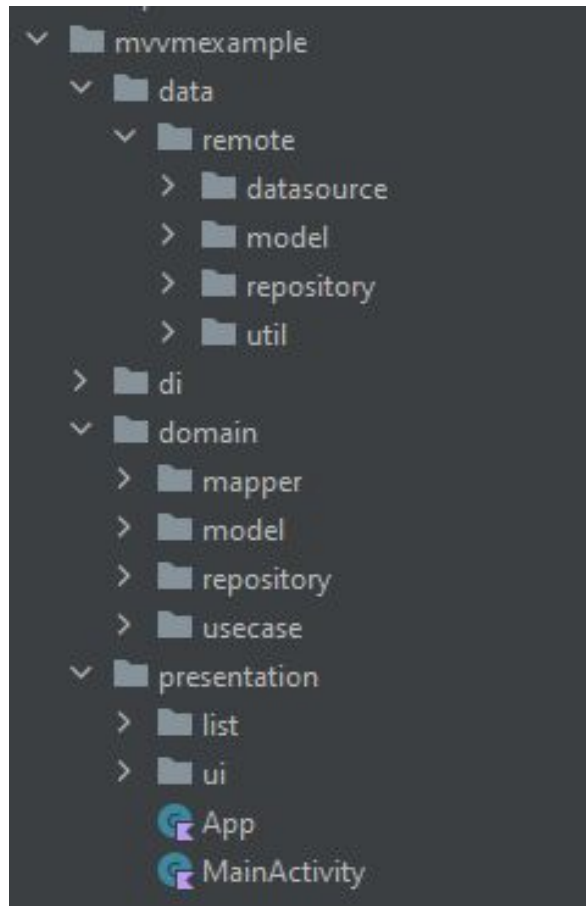


Struktura projektu

Warstwa Danych: Warstwa danych obejmuje interfejsy API, bazy danych i implementację Repozytorium. Jej rola to zarządzanie przepływem danych, trwałością i ułatwianie dostępu, a także pełni funkcję interfejsu abstrakcyjnego dla innych warstw aplikacji.

Warstwa Domeny: W tej warstwie znajdują się modele i interfejsy Repozytorium, a także biznesowa logika projektu wraz z przypadkami użycia. Warstwa domeny jest centralnym punktem dla logiki biznesowej, transformując surowe dane z warstwy danych w przetworzony format dla warstwy prezentacji.

Warstwa Prezentacji: Warstwa prezentacji obsługuje interakcję użytkownika i wyświetlanie rezultatów. Zawiera interfejs graficzny, a w architekturze MVVM składa się z obiektów widoku i widoku modelu. Widoki powinny być proste i pozbawione logiki biznesowej, z koniecznością refaktoryzacji, jeśli zawierają taką logikę.




Model

Reprezentuje dane i logikę biznesową aplikacji. Model jest odpowiedzialny za przechowywanie danych oraz logikę związaną z ich przetwarzaniem.

ViewModel

ViewModel obsługuje zdarzenia i polecenia związane z interakcją użytkownika. Wzorzec ten często wykorzystuje mechanizmy powiązań danych, które automatycznie synchronizują dane między Modelem a Widokiem.



```
@HiltViewModel
class ListViewModel @Inject constructor(
    private val getCharactersUseCase: GetCharactersUseCase
) : ViewModel() {

    // Expose screen UI state
    var viewState by mutableStateOf(ListScreenViewState())
    private set

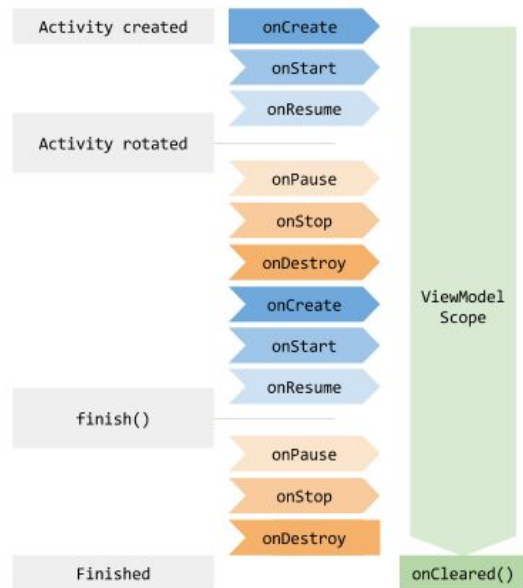
    // Handle business logic
    init { ... }

    fun changeColor(id: Int?, option: CountOptions) { ... }
```

Cykl życia: ViewModel

Cykl życia ViewModelu w architekturze MVVM w Androidzie obejmuje jego tworzenie podczas utworzenia związanej z nim komponenty, takiej jak Activity czy Fragment, oraz niszczenie wraz z zakończeniem tego komponentu.

ViewModel jest odpowiedzialny za przechowywanie i zarządzanie danymi w trakcie zmian konfiguracji, takich jak obrót ekranu. Dzięki temu, dane przeżywają zmiany cyklu życia związane z komponentem interfejsu użytkownika.



View

Odpowiada za prezentację danych użytkownikowi i interakcję z użytkownikiem. To jest warstwa, którą użytkownik widzi i z którą może bezpośrednio integrować.

```

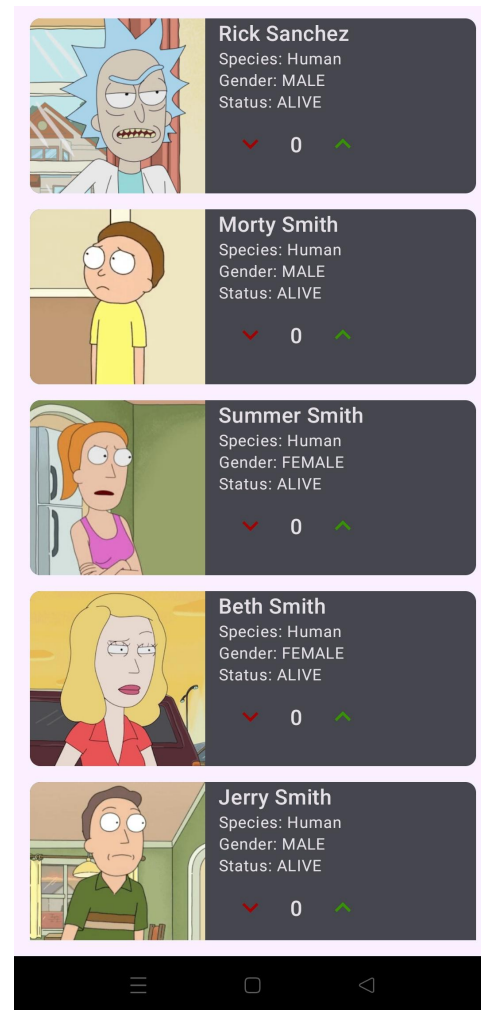
@Composable
fun ListScreen(
    listViewModel: ListViewModel = hiltViewModel()
) {

    // Consume view state
    val state = listViewModel.viewState

    // Display list
    LazyColumn {
        items(state.characters) { character ->
            CharacterItem(
                character = character,
                onCountChange = listViewModel::changeColor
            )
        }
    }
}

```

Wygląd naszej aplikacji



Dziękujemy za uwagę!

