MATURE DEVELOPMENT

Strukturiranje koda i primjena MVP arhitekturalnog paterna

R.C. Martin (Uncle Bob)

-Agile software development

-Clean code

-The clean coder

-Clean architecture

Što je clean?

- Pomaže strukturirati kod za nas i osobito druge

- Pratiti SOLID principe

- Paziti na imanovanje varijabli/metoda, stavljati deskriptivna imena, jer većinu vremena provodite čitajući kod

SOLID

- Single responsibility principle
- Open-closed principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle

Single responsibility principle

- Svaka stavka treba raditi jednu i samo jednu stvar

- Stavka je svaka komponeneta koju možete kreirati u kodu (varijable/metode)

- Mjenjanjem bilo koje od stavki ne remetimo ostatak koda/aplikacije

Open-closed principle

- Sve treba biti otvoreno za ekstenziju ali ne i modifikaciju

- Kada radimo nekakvo sučelje, treba biti takvo da kada mu dodajemo novo ponašanje, ništa u ostatku koda se ne smije pokidati

- Ako promijenimo jedno od ponašanja ostala bi trebala ostati netaknuta

Liskov substitution principle

- Svaki tip koji koristimo se mora moći zamijeniti s nekim od subtipova ili supertipova bez da se nešto posebno pokida

- Tipičan primjer : klasa Osoba i subtipovi Student i Profesor — trebamo moći koristiti samo Osobu pomoću objekata P i S, kako bi ono radilo za oba slučaja

Interface segregation principle

- Bolje napraviti više manjih interface-a i implementirati ih sve, nego jedan veliki kako bi izbjegli implementaciju metoda koje nam nisu potrebne

- Lakše za čitajne, održavanje i izmjenjivanje, promjene se događaju na manjem broju mjesta

- Ps - DRY - Don't repeat yourself

Dependency inversion principle

- Apstrakcije nebi trebale ovisiti o detaljima

- Detalji ovise o apstrakcijama

- “Low level” stvari ne trebaju ovisiti o “high level stvarima”

- Bolje je ovisiti o interfaceu, nego o konkretnoj klasi

MVVM

- Model - vanjski entitet (baza, networking)

- ViewModel - logika i rad s podatcima, dohvaćanje i spremanje

- View - “UI” sloj, Andoird platforma

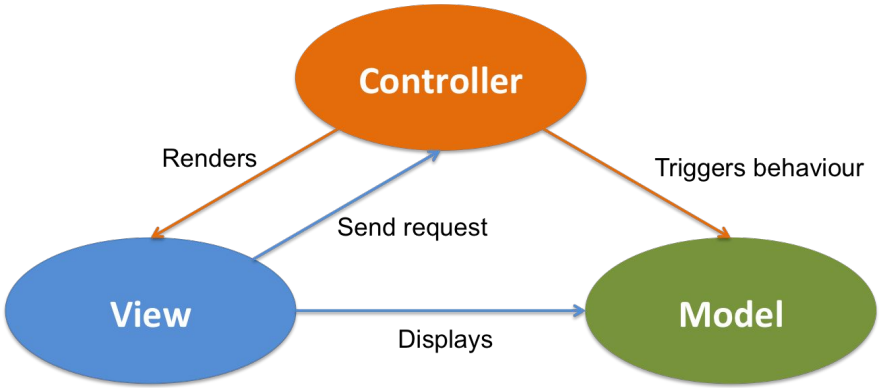
MVP

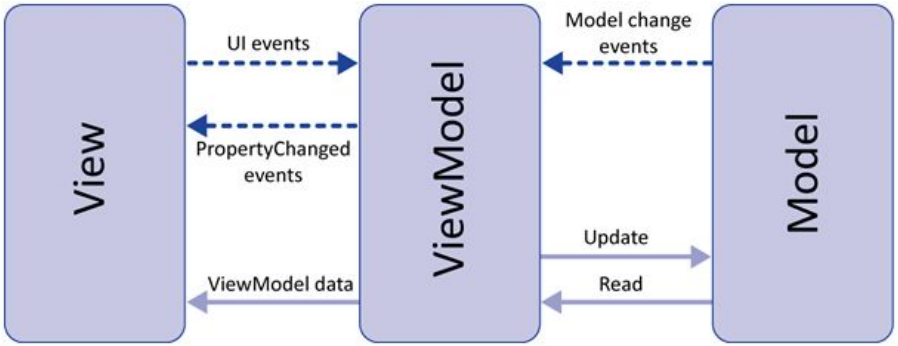
- Model - vanjski entitet (baza, networking)

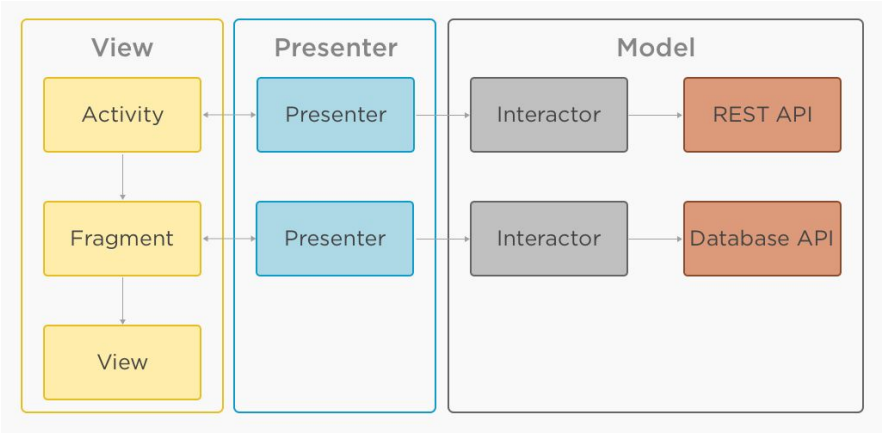
- Presenter - komunikacija između View-a i Modela, sva logika rada nad podatcima

- View - “UI” sloj, android platforma

MVC - MVP - MVVM


Controller
Renders
Triggers behaviour
Send request
View
Displays
Model


View
UI events
PropertyChanged events
ViewModel
Model change events
Update
ViewModel data
Read
Model


View
Activity
Fragment
View
Presenter
Presenter
Presenter
Model
Interactor
REST API
Interactor
Database API

Zašto paterni

- Nemamo God klase koje sadrže 500, 1000 i više linija koda

- Odvajanje koda nam pomaže kod refaktoriranja, testiranja, proširivanja i razumjevanja (netko će poslije tebe raditi na tom projektu, drži kod onakav kakav očekuješ da ćeš dobiti)

- Nek stvari posatju reusable (DRY)

ZADAĆA

- Zadatak s gita(link na moodle-u) prebaciti u MVP
- Dodati mogućnost favoritanja
- Dodati prikaz top filmova i favorite filmova
- Ishendlati sve errore, primijeniti što ste naučili na prošlim predavanjima

- Naravno sve MVP