MVP

MODEL-VIEW-PRESENTER

Vad är MVP?

- Ett pattern som tillåter oss att separera hur våra vyer fungerar från hur de ser ut.

- Vi låter alltså våra fragments/activities att delegera funktionalitet till en “presenter”.

MVP

Model View Presenter

VIEW

UI: Activity,
Fragments

MODEL

Data: Models, DB
e business logic

PRESENTER

Mediator

Varför ska vi använda MVP?

- Activities är ofta hårt kopplade till både interface och data.
- För att vår applikation ska vara enkel att skala och förvalta så behövs tydliga lager.
- Vyerna blir bortkopplade från datalager.
- Flytta ut logik från activities.
- Lättare att testa.

Implementation

- Finns många olika tolkningar. Min tolkning baserad på läsning av olika artiklar.

- Många gråzoner, vem är egentligen ansvarig för vad?

- Från situation till situation.

Presenter

- Interface
- Middleman mellan model och vy.
- Får data från modellen och ger det fint formaterat till vyn.
- Ansvar för vad som händer vid interaktion med vyn.

View

- Interface
- Ansvarig för att skapa Presenter.
- Delegerar all interaktion till Presentern.
- Ansvarig för hur saker ser ut.

Model

- Ansvarig för att leverera data.

- I en väl strukturerad arkitektur så är modellen endast en gateway för data.

KOD DEMO