**Univerzitet u Beogradu**

**Matematički fakultet**

**Komparativna analiza modernih tehnologija u razvoju Android aplikacija**

Master rad

**Kandidat: Mentor:**

Stefan Bačević Prof. dr Vladimir Filipović

**Uvod**

Android je operativni sistem zasnovan na Linux jezgru, razvijen od strane kompanije Google primarno za mobilne uređaje. Od inicijalnog predstavljanja 2008. godine stekao je ogromnu popularnost i danas je najzastupljeniji mobilni operativni sistem na tržištu. Sa sve većom upotrebom različitih mobilnih uređaja kao i nastojanjem da Android bude operativni sistem koji će pokretati najrazličitije uređaje u okviru interneta stvari (Internet of things) možemo očekivati da se trend rasta udela Androida na tržištu nastavi.

Uprkos svemu tome razvijanje aplikacija za Android je i dalje mlada oblast računarstva koja se razvija veoma brzo. Programeri su “u hodu” otkrivali kako najbolje da primene postojeće šablone u programiranju na novi kontekst i uporedo sa tim su se razvijali alati i biblioteke koji pojednostavljuju i ubrzavaju programiranje Android aplikacija. Vremenom je pređen put od aplikacija čija je sva logika sadržana u aktivnostima aplikacije (eng. Activity) preko korišćenja Model-pogled-kontroler šablona (eng. Model-View-Controller, u daljem tekstu MVC) sve do danas aktuelnih šablona Model-kontroler-prezenter (eng. Model-View-Presenter, u daljem tekstu MVP) i Model-Pogled-Pogled-Model (eng. Model-View-View-Model, u daljem tekstu MVVM). Zbog sve složenijih projekata a prateći SOLID princip inverzije kontrole započeto je korišćenje bilblioteka koje omogućavaju umetanje zavisnosti kao što su RoboGuice i Dagger.

U okviru rada biće kreirana i aplikacija otvorenog koda “Movie Master“ koja će pored pomenutih arhitekturalnih šablona i biblioteka predstaviti još i reaktivni način programiranja i njegovu podršku u Androidu kroz RxAndroid kao i podršku za Javu 8. Takođe će biti predstavljene i biblioteke “Butter Knife” za povezivanje promenljivih u Java kodu sa komponentama u xml-u preko anotacija, “Stetho“ koja olakšava pronalaženje i otklanjanje grešaka i „Fabric“ platforma za praćenje i analitiku. Aplikacija će služiti za pregled informacija o filmovima, korisniku će se predstaviti lista filmova koju može da pretražuje a klikom na željeni film prikazaće mu se dodatne informacije i imaće mogućnost da ga sačuva u personalizovanu listu, podeli i oceni. Na taj način će, pored teorijske analize i komparacije različitih tehnologija, biti prikazana i njihova implementacija u realnom okruženju i praktična primena u rešavanju najčešće viđenih problema u razvoju Android aplikacija. Aplikacija će težiti da bude razvijena u skladu sa dobrim praksama i da samim tim bude testabilna, modularna i jednostavna za buduće održavanje i eventualne izmene.

Cilj rada je da svaka od ovih tehnologija će biti sagledana i analizirana iz različitih perspektiva pri čemu će biti istaknute pozitivne i negativne strane i napravljeno kritičko poređenje sa osvrtom na neka ranija rešenja. Shodno tome rad će predstavljati svojevrsno istraživanje najkorišćenijih modernih tehnologija u razvoju Android aplikacija.

**Model-Pogled-Prezenter**

Model-Pogled-Prezenter je arhitekturalni šablon, nastao iz Model-Pogled-Kontroler šablona, koji se najviše koristi za kreiranje korisničkog interfejsa. MVP šablon se pojavljuje prvi put početkom 90-ih godina u kompaniji Taligent kao osnova za razvoj aplikacija u njihovom objektno orijentisanom CommonPoint razvojnom okruženju. Kasnije je ovaj šablon upotrebljen za kreiranje osnove za Dolphin Smalltak korisnički interfejs dok je 2006. Microsoft uveo MVP u dokumentaciju i primere za programiranje korisničkih interfejsa u .NET razvojnom okviru.

Glavne motivacije za uvođenje MVP šablona je podela odgovornosti u prezentacionoj logici, viši stepen modularnosti koda i mogućnost automatskog testiranja koda. Sa početkom razvoja Android operativnog sistema većina programera je gotovo celu logiku aplikacije pisala u aktivnostima aplikacije. Pokazalo se da su takve aplikacije jako teške za održavanje jer je prezentaciona logika usko povezana sa biznis logikom i modelom, pored toga automatsko testiranje aplikacija je gotovo nemoguće. Nakon toga je popularnost stekao Model-Pogled-Kontroler (MVC) šablon. Kod MVC šablona imamo tri komponente:

* Statičan pogled koji u sebi nema nikakvu logiku i koji ne čuva stanje. Pogled je u Androidu uglavnom implementiran u okviru statičnih resursnih xml datoteka.
* Model koji je zadužen za dohvatanje i čuvanje podataka. U konkretnoj implementaciji u modelu može biti komunikacija sa bazom ili pozivanje eksternog servisa, takođe u modelu je implementirana i biznis logika aplikacije.
* Kontroler implementira neophodne transformacije podataka za prikaz, pored toga kontroler može biti deljen između više pogleda i kontroliše koji će pogled biti prikazan. U Androidu kontroler je uglavnom implementiran u okviru aktivnosti.

Iako je ovaj šablon doneo izvesnu modularnost i podelu odgovornosti i dalje je u određenoj meri postojala spregnutost između prikaza i modela i biznis logike što se odražavalo na otežano pisanje automatskih testova.

Kod Model-Pogled-Prezenter šablona slično kao kod MVC šablona imamo tri komponente:

* Model – model ima potpuno istu ulogu i implementaciju kao kod MVC šablona. On služi za dohvatanje podataka i biznis logiku aplikacije, za razliku od MVC šablona, model je potpuno nezavisan od prikaza.
* Pogled – pogled je odgovoran za predstavljanje podataka i za prihvatanje interakcije korisnika. Poželjno je da pogled služi samo za jednostavan prikaz podataka koje dobije od prezentera kao i prosleđivanju korisničkih interakcija prezenteru i da u sebi sadrži što manje logike. Pogled najčešće implementiraju aktivnosti i fragmenti aplikacije.
* Prezenter – prezenter zahteva podatke od modela, vrši transformacije nad podacima tako da budu pogodni za prikaz, kontroliše prikaz i reaguje na korisničku interakciju. Za svaki pogled postoji poseban prezenter.

Pošto su pogled i prezenter usko povezani oni moraju da imaju referencu jedan ka drugom a sa druge strane je potrebno da budu modularni radi automatskog testiranja ili potpune zamene jednog od ta dva sloja. Iz tog razloga oni su abstrahovani i implementiraju interfejse koji služe kao svojevrsni “dogovor” između ta dva sloja i čine kod čitljivijim.



Model-Pogled-Prezenter shema

**Implementacija**

Postoji više načina za implentaciju MVP šablona, iako je ideja u osnovi identična, za određene slučajeve je bolje koristiti implementaciju ovog šablona sa izvesnim manjim izmenama. Implementacija koja će biti predstavljena je zasnovana na preporukama kompanije Google i prilagođena potrebama aplikacije.

Uzmimo sledeći scenario za primer na kome će biti pokazana implementacija MVP šablona. U aplikaciji koju razvijamo potrebno je dohvatiti podatke preko mreže sa REST servisa i prikazati podatke u listi na android uređaju. Na osnovu te liste korisnik će kasnije moći da pregleda detalje o filmovima, da favorizuje određene filmove, pretraži listu itd. Međutim za početak želimo samo da predstavimo listu gde će za svaki film biti prikazan naslov, godina premijere, ocena i poster. U skladu sa realističnim produkcionim scenariom potrebno je i da se omogući relativno jednostavno i brzo menjanje servisa za podatke kao i celokupnog prikaza liste. Takođe je potrebno da arhitektura aplikacije omogućava automatsko testiranje.

Za potrebe aplikacije razvijen je REST server u NodeJS-u radi nezavisnosti od spoljnih servisa kao i zbog veće kontrole nad resursima i boljeg predstavljanja bitnih koncepta. NodeJS je izabran kao jezik servera prevashodno zbog brzine i lakoće razvijanja jednostavnih servisa. Kada klijent uputi GET zahtev servisu na putanji /all servis će vratiti listu JSON objekata u sledećem formatu: