NEWSMAN

Vukić Uroš 15964

Sreten Šikuljak 15861

Sadržaj

[1. Arhitekturni dizajn 3](#_Toc19721117)

[1. Arhitekturni obrasci 3](#_Toc19721118)

[2. Generalna arhitektura 3](#_Toc19721119)

[3. Strukturni pogledi 3](#_Toc19721120)

[4. Bihevioralni pogledi 4](#_Toc19721121)

[5. Alokacioni dijagram 6](#_Toc19721122)

[6. Implementaciona pitanja – biblioteke, komponente i okviri (framework) koji će biti korišćeni za implementaciju 6](#_Toc19721123)

[2. Projektni obrasci 7](#_Toc19721124)

[1. Observer 7](#_Toc19721125)

[2. Strategy 7](#_Toc19721126)

[3. Composite 8](#_Toc19721127)

[4. Builder 8](#_Toc19721128)

[5. Decorator 9](#_Toc19721129)

[6. Singleton 9](#_Toc19721130)

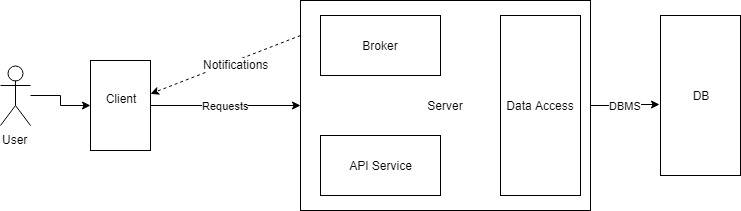
[7. MessageQueue Strategy & Factory 10](#_Toc19721131)

# Arhitekturni dizajn

## Arhitekturni obrasci

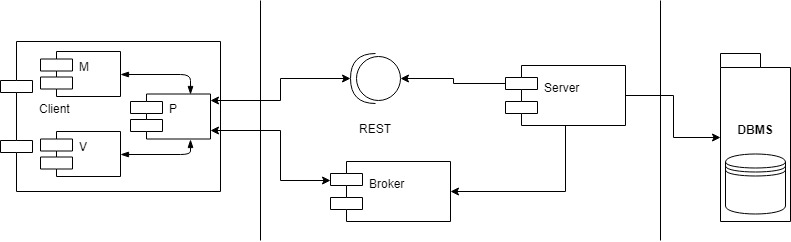
* Slojevita arhitektura ( Layerd architecture) – sistem jerazvijen kao 3-slojna client-server arhitektura radi struktuiranja dizajna i postizanja niskog stepena povezanosti između komponenti (“loosely coupled”)
* MVP (Model-View-Presenter) – ova aritektura je nametnuta od android-studio framework-a
* Publish-Subscribe – model za asinhronu implicitnu komunikaciju između klijenta i srevera, omogućava da se klijenti pretplate na željni sadržaj i dobijaju obaveštenja kada dođe do izmene tog sadržaja ( implemntiran od strane Message Broker-a)
* Skladište (Repository) – svi podatci se čuvaju u bazi podataka
* ServiceLayer – omogućuje jedinstveni pristup podacima od strane raznovrsnih klijenata

## Generalna arhitektura

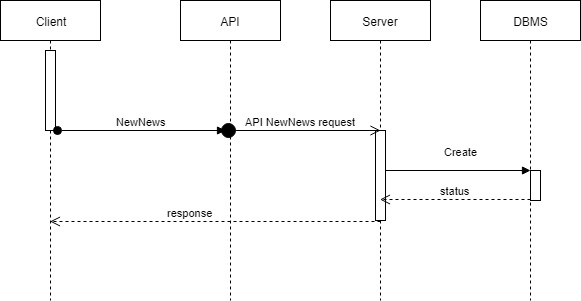


## Strukturni pogledi

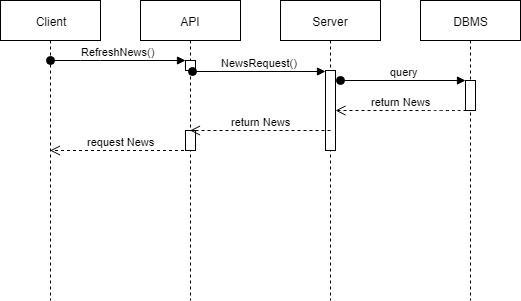
Prikazana je struktura sistema kao i njegove gradivne komponente. Klijentski deo aplikacije je realizovan na osnovu MVP projektnog obrasca. On preko REST servisa sinhrono pribavlja podatke o vestima sa servera. Message Broker služi za asinhronu komunikaciju i obezbeđuje da samo pretplaćeni korisnici dobijaju izmene o vestima. Na Server-u se izvršava API koji komunicira sa bazom podataka.



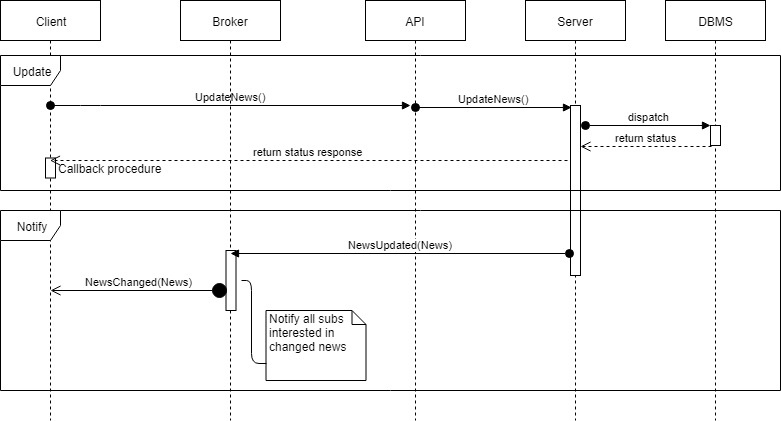
## Bihevioralni pogledi



1. Prikazuje komunikaciju pri kreiranju nove vesti

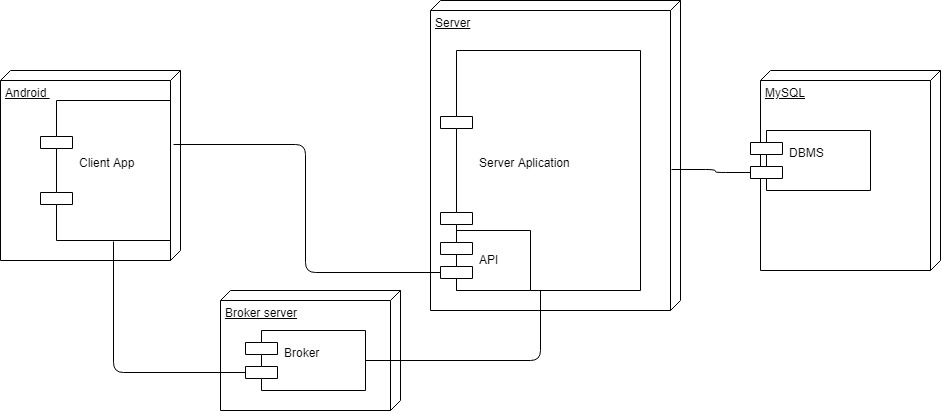


2. Prikazuje pribavljanje vesti od strane klijenta



3. Prikazuje komunikaciju prilikom ažuriranja vec postojeće vesti

## Alokacioni dijagram



## Implementaciona pitanja – biblioteke, komponente i okviri (framework) koji će biti korišćeni za implementaciju

Android-studio framework - za izradu klijentske aplikacije

ASP.NET framework - za izradu web API servera

Fluent NHybernate – ORM maper

MySQL DBMS – baza podataka

RabbitMQ – Message Broker server

RabbitMQ.NET – biblioteka za Message Broker servera

RabbitMQ-Client.jar – biblioteka za Message Broker klijenta

Retrofit – biblioteka za realizaciju REST konekcije sa serverom

Room – ORM mapper za mapiranje na SQL lite

LiveData – biblioteka koja omogućava praćenje promena u podacima

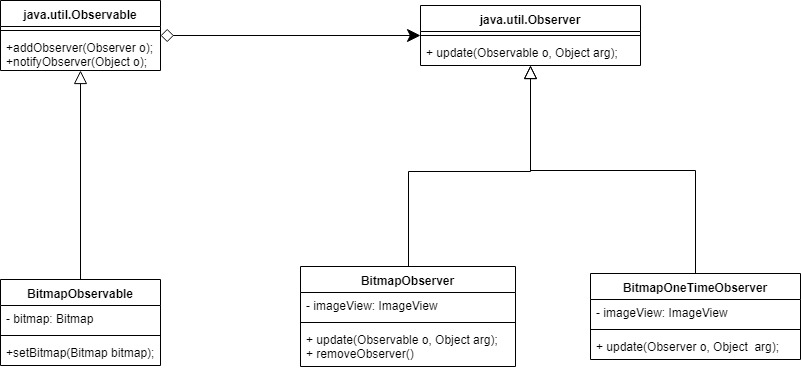
Gson – Google JSon serializator

Dagger – android biblioteka za implementaciju Dependency Injection

# Projektni obrasci

## Observer

Zbog performansi prikaza vesti zahtevi za slikama su odvojeni od zahteva za vesti. Time se postiže da se kontrola vrati UI threadu a pribavljanje velikih binarnih podataka se obavlja u pozadinskoj niti. Kao placeholder se koristi default-na slika. Kada se pribavi zahtevana slika ImageView se azurira. Ovo se postiže korišćenjem Observer projektnog obrasca.



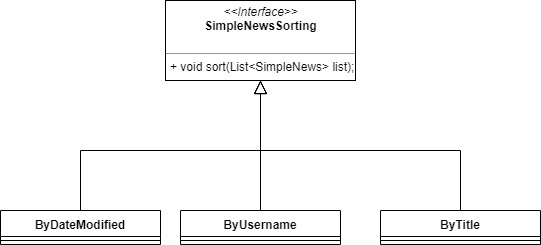
- **BitmapObservable** – služi da wrap-uje sliku da bi mogla da se posmatra. Asinhroni pozivi za dobavljanje slike sa servera pozivaju funkciju *setBitmap(Bitmap bitmap)* koja poziva *notifyObservers(Object o)*. Notifikacija se izvršava na glavnoj niti.

- **BitmapObserver** – klasični observer. Sadrži referencu na ImageView klasu. Pri svakom pozivu *update(Obesrvable o, Object arg)* se menja slika koja se prikazuje na ImageView.

- **BitmapOneTimeObserver** – ovaj observer se koristi pri ažuriranju vesti. Ima mogucnost da pribavi sliku i pri prvom dostavljanju slike se skida sa liste observera. Na taj način ne može da dodje do promene slike koja se ažurira od strane trećeg lica.

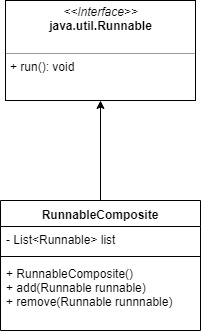
## Strategy

Prilikom prikazivanja liste vesti postoji potreba da se sortiraju po odredjenom kriterijumu. U tu svrhu smo iskoritili Strategy projektni obrazac gde se algoritam za sortiranje enkapsulira u interface-u a u konkretnim klasama su implementirani algoritmi.



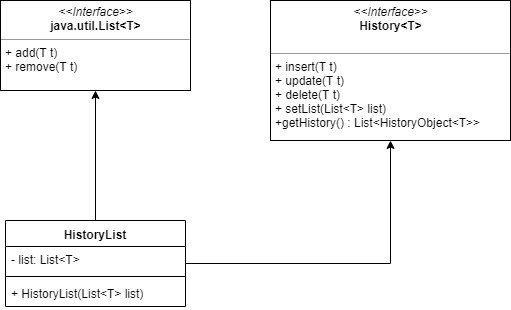
## Composite

Da bi se grupa ažuriranja posmatrala isto kao i jedano ažuriranje kreirali smo composite obrazac.



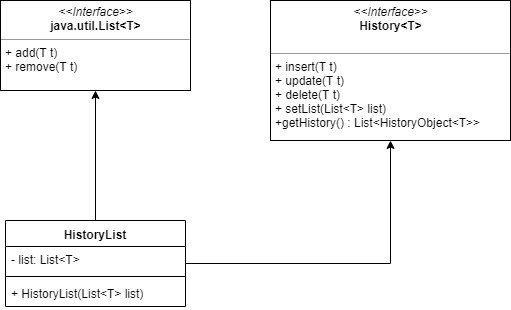
## Builder

Da bi smo sakrlili kompleksnost kreiranja composit-a sakrili smog a iza Builder obrasca.



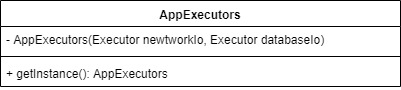
## Decorator

HistoryList je Decorator oko javine liste koji nam dozvoljava da pratimo promene koje su napravljene u dekorisanoj listi. To koristimo u modulu za ažuriranje da bi pratili izmene na ulaznim podacima.



## Singleton

Singleton obrascem obezbedili smo da postoji jedna instanca AppExecutor klase i obezbedjen je globalni pristup klasi.



## MessageQueue Strategy & Factory

Radi smanjenja spregnutosti komponenti za obradjivanje publish-subscribe mehanizma klasa koja obrađuje zahteve ne zna na koj način se obradjuje zahtev. Koristeći factory pribavlja interfejs obradjivača i poziva metodu za obradu. Vredno je napomenuti da postoji i NullUpdate klasa koja je NullObject pattern.

