NEWSMAN

Vukić Uroš 15964

Sreten Šikuljak 15861

Sadržaj

[1. Kontekst i cilj projekta 3](#_Toc531864647)

[2. Arhitekturni zahtevi 3](#_Toc531864648)

[1. Arhitekturno značajni slučajevi korišćenja (glavni funkcionalni zahtevi) 3](#_Toc531864649)

[2. Ne-funkcionalni zahtevi 3](#_Toc531864650)

[3. Tehnička i poslovna ograničenja 3](#_Toc531864651)

[3. Arhitekturni dizajn 3](#_Toc531864652)

[1. Arhitekturni obrasci 3](#_Toc531864653)

[2. Generalna arhitektura 4](#_Toc531864654)

[3. Strukturni pogledi 4](#_Toc531864655)

[4. Bihevioralni pogledi 5](#_Toc531864656)

[5. Alokacioni dijagram 6](#_Toc531864657)

[6. Implementaciona pitanja – biblioteke, komponente i okviri (framework) koji će biti korišćeni za implementaciju 7](#_Toc531864658)

# Kontekst i cilj projekta

Newsmen je aplikacija za kolaborativno kreiranje novina od strane grupe korisnika sa ciljem smanjenja cenzure prilikom izveštavanja o događajima i novostima. Svaki korisnik može da kreira neku novu vest, pri čemu ostali korisnici mogu da pregledaju datu vest i po potrebi izmene i na taj način dodaju svoje vidjenje događaja. Svaka vest pored teksta koji opisuje odgađaj može sadržati i dodatne multimedijalne sadržaje koji je bolje opisuju (fotografija, video, zvuk ...).

# Arhitekturni zahtevi

## Arhitekturno značajni slučajevi korišćenja (glavni funkcionalni zahtevi)

* Pregled vesti u svakom trenutku
* Kreiranje nove vesti
* Izmena već postojeće vesti
* Mogućnost prijavljivanja za praćenje vesti
* Vest može sadržati bilo koji multimedijalni sadržaj

## Ne-funkcionalni zahtevi

* Skalablnost – aplikacija mora biti skalabilna jer je cilj aplikacije kolaborativno kreiranje vesti i samim tim postoji mogućnost velikog broja korisnika sistema
* Proširljivost – aplikacija treba da bude lako proširljiva za dodavanje novih funkcionalnosti, kreiranja novih tipova klijenata kao i novih tipova sadržaja za opisivanje vesti, ocenjivanje verodostojnosti sadržaja vesti, portabilnost na druge platforme
* Dostupnost – sistem mora biti aktivan 24h/7 dana u nedelji i biti dostupan što većem broju korisnika („ on the go“ – u svakom trenutku)
* Modifikabilnost – laka izmena funkcionalnosti sistema
* Pouzdanost – nametnuto od strane tržišta i konkurencije
* Upotrebljivost – aplikacija treba da ima intuitivan korisnicki interface

## Tehnička i poslovna ograničenja

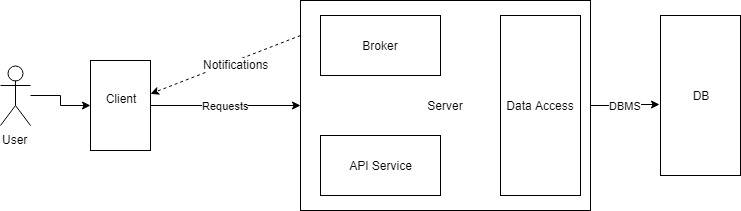
Za izradu sistema potrebno je koristiti open-source biblioteke i okruženja.

# Arhitekturni dizajn

## Arhitekturni obrasci

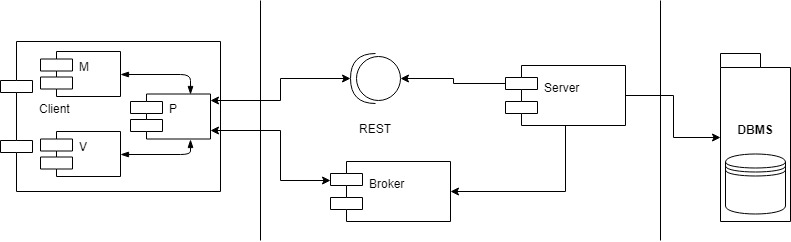
* Slojevita arhitektura ( Layerd architecture) – sistem jerazvijen kao 3-slojna client-server arhitektura radi struktuiranja dizajna i postizanja niskog stepena povezanosti između komponenti (“loosely coupled”)
* MVP (Model-View-Presenter) – ova aritektura je nametnuta od android-studio framework-a
* 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

## Generalna arhitektura

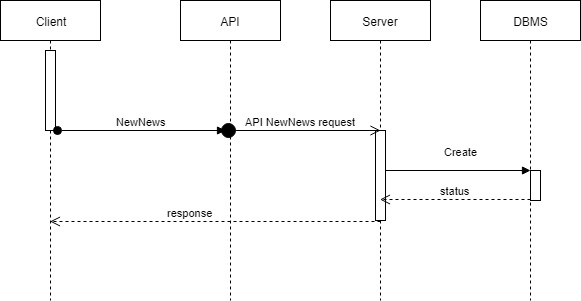


## Strukturni pogledi

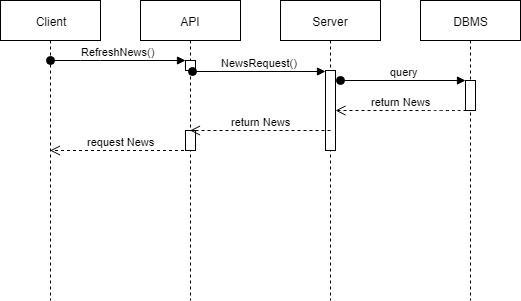
Prikazana je struktura sistema kao i njegove gradivne komponente. Klijentski deo aplikacije je realizovan na osnovu MVP projektnog obrasca. On preko REST servisa sinhrono pribavlja podatke o vestima sa servera. Message Broker služi za asinhronu komunikaciju i obezbeđuje da samo pretplaćeni korisnici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



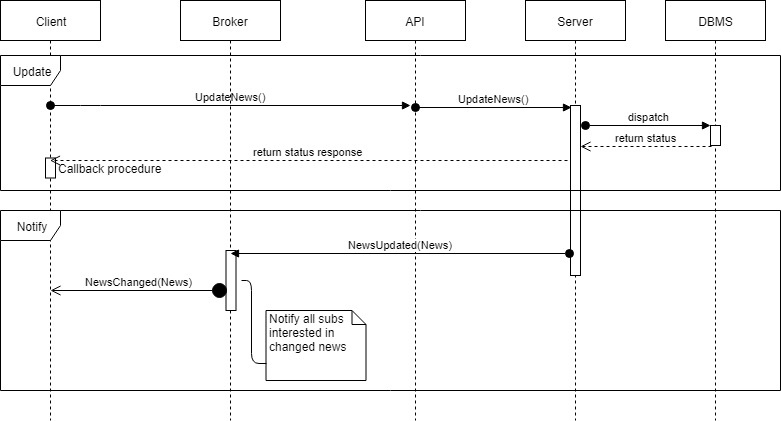
## Bihevioralni pogledi



1. Prikazuje komunikaciju pri kreiranju nove vesti

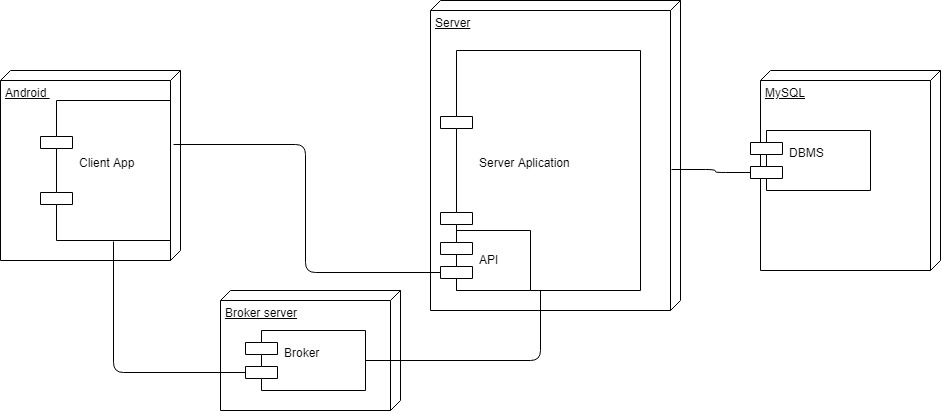


2. Prikazuje pribavljanje vesti od strane klijenta



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

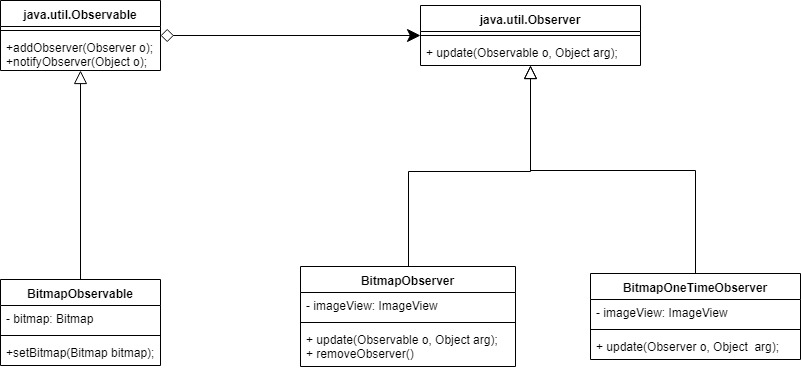
RabbitMQ.NET – biblioteka za Message Broker servera

RabbitMQ-Client.jar – biblioteka za Message Broker klijenta

# 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 asinhrono. Kao placeholder se koristi default-na slika. Kada se pribavi zahtevana slika ImageView se azurira. Ovo se postiže korišćenjem Observer projektnog obrasca.

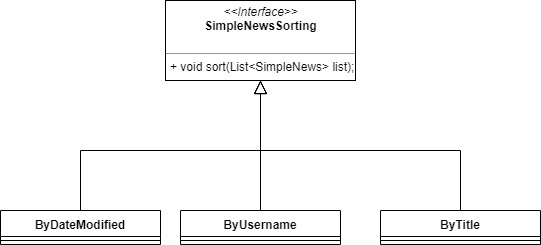


- **BitmapObservable** – služi da wrap-uje sliku da bi mogla da se posmatra. Asinhroni pozivi za dobavljanje slike sa servera pozivaju funkciju *setBitmap(Bitmap bitmap)* koja poziva *notifyObservers(Object o)*. Notifikacija se izvršava na glavnoj niti.

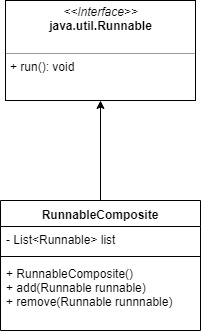
- **BitmapObserver** – klasični observer. Sadrži referencu na ImageView klasu. Pri svakom pozivu *update(Obesrvable o, Object arg)* se menja slika koja se prikazuje na ImageView.

- **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 kompozita 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

Ovo je singleton.

## 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.