Zadruga

Arhitekturni projekat

Verzija 1.0

Pregled izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 12.04.2021. | 1.0 | Inicijalna verzija | Vuk, Lazar, Teodora, Uroš |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

1. Cilj dokumenta 5

2. Opseg dokumenta 5

3. Reference 5

4. Predstavljanje arhitekture 5

5. Ciljevi i ograničenja arhitekture 5

6. Pogled na slučajeve korišćenja 5

6.1 Dijagrami slučajeva korišćenja 6

6.2 Kratak opis slučajeva korišćenja 7

6.2.1 Registracija na sistem 7

6.2.2 Prijavljivanje na sistem 8

6.2.3 Unos podataka o korisniku 8

6.2.4 Ažuriranje podataka o korisniku 8

6.2.5 Prikaz podataka o korisniku 8

6.2.6 Kreiranje oglasa 8

6.2.7 Prikaz oglasa 8

6.2.8 Ažuriranje postojećeg oglasa 8

6.2.9 Brisanje oglasa 8

6.2.10 Sortiranje oglasa po lokaciji 8

6.2.11 Filtriranje oglasa po lokaciji 8

6.2.12 Filtriranje oglasa po kategorijama poslova 8

6.2.13 Filtriranje oglasa po novčanoj naknadi 8

6.2.14 Preporuka poslovnih oglasa 8

6.2.15 Apliciranje za posao 9

6.2.16 Odabir kandidata i formiranje komunikacionog kanala 9

6.2.17 Ostvarivanje komunikacije između poslodavca i radnika 9

6.2.18 Ocenjivanje radnika i dodela bedževa 9

6.2.19 Ocenjivanje poslodavaca 9

6.2.20 Izbor jezika 9

6.2.21 Kontrola neprikladnog sadržaja 9

7. Pogled na logičku arhitekturu sistema 9

7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve 10

7.1.1 Korisnički interfejs 10

7.1.2 Firebase 10

7.1.3 Aplikaciona logika 10

7.1.4 Pristup podacima 10

7.1.5 Sistem za preporuku 10

7.1.6 XML 11

7.1.7 Android SDK 11

7.1.8 Android Architecture Components 11

7.1.9 Django framework 11

7.1.10 PyTorch 11

7.1.11 MySQL 11

8. Pogled na procese 11

8.1 Procesi 11

8.1.1 Pozadinski proces obaveštenja 12

8.1.2 Server 12

8.1.3 Aplikacija 12

8.1.4 SQL Server 12

9. Pogled na raspoređivanje sistema 12

9.1 Klijent 13

9.2 Server 13

9.3 DBMS server 13

10. Pogled na implementaciju sistema 13

10.1 Model domena 13

10.2 Šema baze podataka 13

10.3 Komponente sistema 14

10.3.1 Komponente za korisnički interfejs 15

10.3.2 Komponente za aplikacionu logiku 17

10.3.3 Komponente za pristup podacima 19

11. Performanse 20

12. Kvalitet 20

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture aplikacije Zadruga.

# Opseg dokumenta

Dokument se odnosi na aplikaciju Zadruga koja će biti razvijena od strane tima Parovi. Namena aplikacije biće kreiranje digitalnog tržišta rada tako da poslodavcima obezbeđuje efikasan i pouzdan način da nađu radnike za male poslove, a studentima brz i lak način da dođu do poslova koji im odgovaraju.

# Reference

Spisak korišćene literature:

1. Zadruga – Predlog projekta, D01\_Predlog\_Projekta, V1.0, 2021, Parovi.
2. Zadruga – Vizija sistema, D02\_Vizija\_Sistema, V1.0, 2021, Parovi.
3. Zadruga – Planirani raspored aktivnosti, D03\_Raspored\_Aktivnosti.mpp, V1.0, 2021, Parovi.
4. Zadruga – Specifikacija zahteva, D04\_Specifikacija\_Zahteva, V1.0, 2021, Parovi

# Predstavljanje arhitekture

Arhitektura sistema u dokumentu je prikazana kao serija pogleda na sistem: pogled na slučajeve korišćenja, pogled na logičku arhitekturu sistema, pogled na procese, pogled na razmeštaj komponenti sistema i pogled na implementaciju. Ovi pogledi su predstavljeni odgovarajućim UML dijagramima.

# Ciljevi i ograničenja arhitekture

Ključni zahtevi i sistemska ograničenja koja imaju značajan uticaj na izbor arhitekture i projektovanje sistema su:

1. Zadruga aplikacija će biti implementiran kao Android aplikacija zasnovana na Java Android Framework-u, Django Framework-u za Python, Firebase Backend-as-a-Service i MySQL bazi podataka
2. Klijentski deo Android aplikacije će biti optimizovan za sledeće Android uređaje od verzije JellyBean
3. Svi zahtevi u pogledu performansi dati u [4] moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema

# Pogled na slučajeve korišćenja

U ovom odeljku je dat pogled na slučajeve korišćenja definisane u specifikaciji zahteva [4].

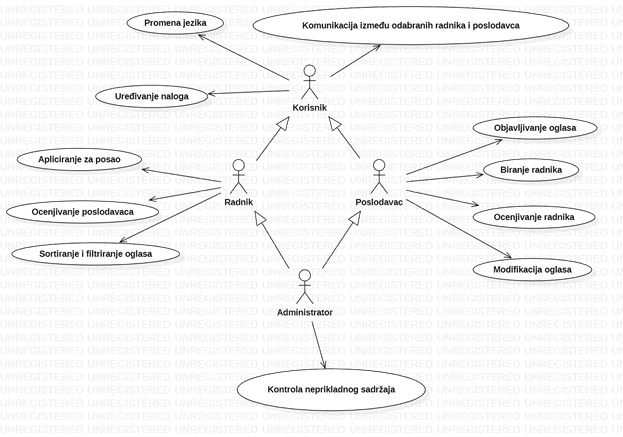
Slučajevi korišćenja aplikacije Zadruga su:

* *Registracija na sistem*
* *Prijavljivanje na sistem*
* *Unos podataka o korisniku*
* *Ažuriranje podataka o korisniku*
* *Prikaz podataka o korisniku*
* *Kreiranje oglasa*
* *Prikaz oglasa*
* *Ažuriranje postojećeg oglasa*
* *Brisanje oglasa*
* *Sortiranje i filtriranje oglasa*
  + *Sortiranje oglasa po lokaciji*
  + *Filtriranje oglasa po lokaciji*
  + *Filtriranje oglasa po kategorijama poslova*
  + *Filtriranje oglasa po novčanoj naknadi*
* *Preporuka poslovnih oglasa*
* *Apliciranje za posao*
* *Odabir kandidata i formiranje komunikacionog kanala*
* *Ostvarivanje komunikacije između poslodavca i radnika*
* *Ocenjivanje radnika i dodela bedževa*
* *Ocenjivanje poslodavaca*
* *Izbor jezika*
* *Kontrola neprikladnog sadržaja*

Ove slučajeve korišćenja mogu da iniciraju korisnici radnici, korisnici poslodavci i administratori.

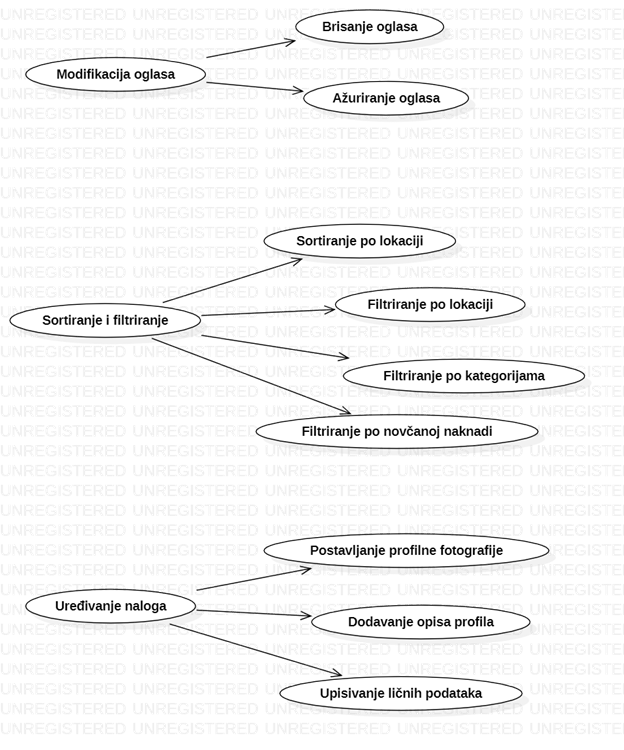
## Dijagrami slučajeva korišćenja

Osnovni UML dijagram koji prikazuje korisnike i slučajeve korišćenja aplikacije Zadruga prikazan je na sledećoj slici:



Slučajevi korišćenja *modifikacija oglasa* i *sortiranje i filtriranje* i *uređenje profila* obuhvataju složenije radnje koje se mogu razložiti dalje razložiti na pojedinačne slučajeve korišćenja.

Detaljni UML dijagrami za gorenavedene slučajeve korišćenja prikazani su na sledećoj slici:



## Kratak opis slučajeva korišćenja

### Registracija na sistem

Kratak opis: Kreiranje novog korisničkog naloga korišćenjem e-mail adrese, korisničkog imena i lozinke. Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Prijavljivanje na sistem

Kratak opis: Korišćenjem već postojećeg naloga (kombinacija e-mail adrese ili korisničkog imena i lozinke), korisnik se prijavljuje na aplikaciju Zadruga.

Akteri koji iniciraju slučaj korišćenja: Korisnik koji je prethodno izvršio registraciju.

### Unos podataka o korisniku

Kratak opis: Novoregistrovani korisnik ima opciju da unese detaljnije podatke o sebi prilikom prvog prijavljivanja.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Ažuriranje podataka o korisniku

Kratak opis: Korisnik može da ažurira sve svoje relevantne podatke, ukoliko je došlo do nekih izmena.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Prikaz podataka o korisniku

Kratak opis: Korisnik može da pregleda svoje podatke, kao i podatke drugih korisnika.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Kreiranje oglasa

Kratak opis: Poslodavci imaju opciju da kreiraju nove oglase za posao koji nude.

Akteri koji iniciraju slučaj korišćenja: Poslodavac.

### Prikaz oglasa

Kratak opis: Radnici (studenti) imaju mogućnost da pregledaju postavljene oglase.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Ažuriranje postojećeg oglasa

Kratak opis: Poslodavac može da promeni pojedine podatke vezane za oglas koje je on sam ranije postavio. Pritom se svi već prijavljeni radnici obaveštavaju o promenama.

Akteri koji iniciraju slučaj korišćenja: Korisnik prijavljen kao poslodavac sa tekućim oglasom za posao.

### Brisanje oglasa

Kratak opis: Poslodavac ima mogućnost da obriše oglas koji je prethodno postavio.

Akteri koji iniciraju slučaj korišćenja: Korisnik prijavljen kao poslodavac sa tekućim oglasom za posao.

### Sortiranje oglasa po lokaciji

Kratak opis: Prikaz svih trenutno dostupnih oglasa sortiranih po lokaciji.

Akteri koji iniciraju slučaj korišćenja: Radnik, Administrator.

### Filtriranje oglasa po lokaciji

Kratak opis: Prikaz trenutno dostupnih oglasa razvrstanih po lokaciji.

Akteri koji iniciraju slučaj korišćenja: Radnik, Administrator

### Filtriranje oglasa po kategorijama poslova

Kratak opis: Prikaz trenutno dostupnih oglasa razvrstanih po odabranim kategorijama.

Akteri koji iniciraju slučaj korišćenja: Radnik, Administrator

### Filtriranje oglasa po novčanoj naknadi

Kratak opis Prikaz trenutno dostupnih oglasa razvrstanih po novčanoj naknadi koja može da se ostvari obavljanjem datog posla.

Akteri koji iniciraju slučaj korišćenja: Radnik, Administrator.

### Preporuka poslovnih oglasa

Kratak opis: Prikaz trenutno dostupnih oglasa na osnovu prethodnih poslova za koje sa korisnik prijavljivao.

Akteri koji iniciraju slučaj korišćenja: Radnik.

### Apliciranje za posao

Kratak opis: Prijavljivanje radnika za obavljanje posla.

Akteri koji iniciraju slučaj korišćenja: Radnik.

### Odabir kandidata i formiranje komunikacionog kanala

Kratak opis: Poslodavac na osnovu liste prijavljenih radnika bira najadekvatnije za posao.

Akteri koji iniciraju slučaj korišćenja: Poslodavac.

### Ostvarivanje komunikacije između poslodavca i radnika

Kratak opis: Komunikacija između poslodavca i radnika putem komunikacionog kanala.

Akteri koji iniciraju slučaj korišćenja: Poslodavac.

### Ocenjivanje radnika i dodela bedževa

Kratak opis: Poslodavci imaju mogućnost da ocene radnike (studente) nakon obavljenog posla. Na osnovu ocena, aplikacija će pojedinim radnicima dodeliti bedževe.

Akteri koji iniciraju slučaj korišćenja: Poslodavac.

### Ocenjivanje poslodavaca

Kratak opis: Radnici imaju mogućnost da ocene poslodavce nakon obavljenog posla.

Akteri koji iniciraju slučaj korišćenja: Radnik, Administrator.

### Izbor jezika

Kratak opis: Izbor između srpskog i engleskog jezika za prikaz informacija u okviru aplikacije.

Akteri koji iniciraju slučaj korišćenja: Korisnik, Administrator.

### Kontrola neprikladnog sadržaja

Kratak opis: Prijavljivanje neprikladnog sadržaja i njegovo eventualno uklanjanje od strane administratora.

Akteri koji iniciraju slučaj korišćenja: Administrator.

# Pogled na logičku arhitekturu sistema

U ovom odeljku je dat pregled logičke arhitekture sistema. Ovaj pogled sadrži opis najznačajnijih klasa, njihove organizacije u pakete i podsisteme, i organizacija podsistema u slojeve. U cilju opisivanja dinamičkih aspekata arhitekture, ovaj odeljak može da uključi opise realizacije najznačajnijih slučajeva korišćenja. Da bi se ilustrovala veza između arhitekturno značajnih klasa, podsistema, paketa ili slojeva moguće je uključiti i odgovarajuće dijagrame klasa.

Logički pogled na Zadruga aplikaciju obuhvata 2 glavna podsistema (aplikacija i serve) unutar kojih imamo 4 zasebna paketa (korisnički interfejs, aplikaciona logika, pristup podacima i sistem za preporuku)

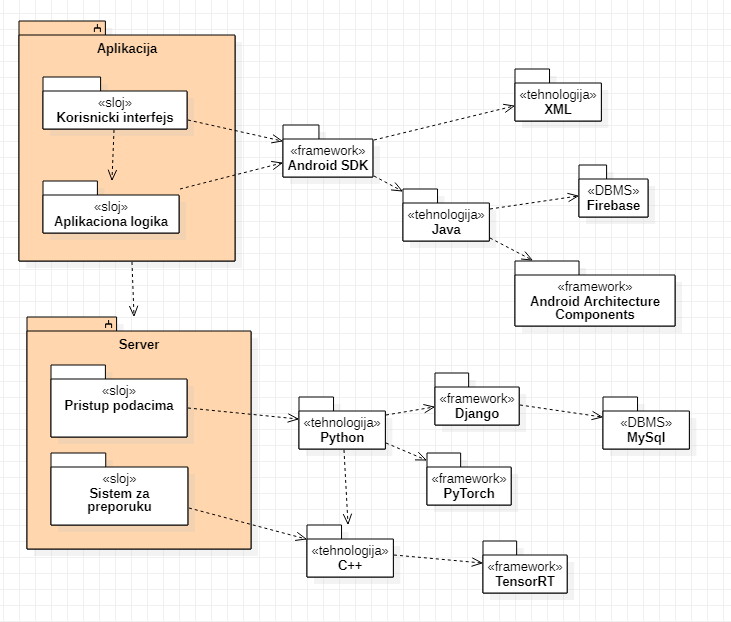
Paket *Korisnički interfejs* sadrži XML fajlove preko kojih će se realizovati grafički dizajn aplikacije kao i forme preko kojih korisnici sistema komuniciraju sa sistemom i resurse u vidu slika i teksta koji će se prikazivati u aplikaciji (koja će biti dvojezična).

Paket *Aplikaciona logika* predstavlja srednji sloj sistema koji sadrži Java klase zadužene za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži Python skripte koje predstavljaju interfejs za pristup, dodavanje i ažuriranje podataka koji se čuvaju u bazi podataka.

Paket S*istem za preporuku* sadrži C++ skripte koje služe da odrede koji će se oglasi prikazati korisniku na osnovu njegovih interesovanja i to na optimizovan i efektivan način.

## Pregled arhitekture – organizacija paketa i podsistema u slojeve



### Korisnički interfejs

Ovaj sloj realizuje korisnički interfejs aplikacije. U njemu su sadržani svi XML fajlovi, multimedijalni sadržaji i ostali resursi potrebni za realizaciju aplikacije. Sloj korisničkog interfejsa zavisi od sloja aplikacione logike, kao i Android SDK framework-a.

### Firebase

Firebase sloj ćemo koristiti da na efikasan način, korišćenjem JSON baze podataka, pamtimo, dodajemo i “povlačimo” podatke vezane za notifikacije i poruke, koje će korisnici unutar komunikacinog kanala međusobno razmenjivati. Ovaj sloj ne zavisi od drugih slojeva.

### Aplikaciona logika

Sloj aplikacione logike predstavlja vezu između korisničkog interfejsa i serverske strane aplikacije. Sadrži Java fajlove čijim se izvršavanjem implementiraju sve funkcionalnosti vidljive od strane korisnika. Ovaj sloj zavisi od serverskog sloja i Android SDK framework-a.

### Pristup podacima

Sloj za pristup podacima sadrži Python skripte koje koriste Django framework i koje su zadužene za pribavljanje, dodavanje i ažuriranje podataka koji se čuvaju u MySQL bazi podataka.

Ovaj sloj ne zavisi od drugih slojeva.

### Sistem za preporuku

Sistem za preporuku predstavlja sloj na serverskoj strani koji je zadužen za generisanje personalizovanih preporuka poslova korisnicima-radnicima na osnovu informacija o njihovim afinitetima i prethodnim poslovima. Sistem će se zasnivati na standardnim rešenjima iz oblasti Mašinskog učenja i biće realizovan u C++ framework-u TensorRT kako bi se potrebna računanja obavljala što brže. Ovaj sloj je zavisan od sloja za pristup podacima.

### XML

Tehnologija XML-a definiše gradivne elemente stranica koje se prikazuju na ekranu uređaja i koje omogućavaju prikaz informacija, dugmića i ostalih sastavnih delova korisničkog interfejsa neophodnih za interakciju sa slojem aplikacione logike.

### Android SDK

Android Software Development Kit predstavlja širok skup alata koji su neophodni za razvoj aplikacija za Android operativni sistem.

### Android Architecture Components

Android Architecture Components jeste skup biblioteka koje pomažu dizajniranju robusnih i održivih Android aplikacija. Kao što ime kaže, one predstavljaju implementaciju standardnih i oprobanih arhitekturnih rešenja koja podržavaju modularnost i time značajno olakšavaju razvoj i održavanje koda.

### Django framework

Django je framework pisan u Python-u koji omogućava lakše povezivanje aplikacije sa bazom podataka. Sadrži Objektno-relacioni maper koji posreduje između modela podataka izraženog kroz Python klase i relacionog modela baze podataka, sistem za obradu HTTP zahteva, kao i razne druge servise koji olakšavaju programiranje.

### PyTorch

PyTorch je framework pisan u Python-u koji omogućava brži i lakši razvoj i treniranje Machine Learning modela. Biće korišćen za treniranje modela sistema za preporuku.

### MySQL

MySQL predstavlja sistem za upravljanje bazama podataka koji će se koristiti za čuvanje podataka neophodnih za rad aplikacije.

# Pogled na procese

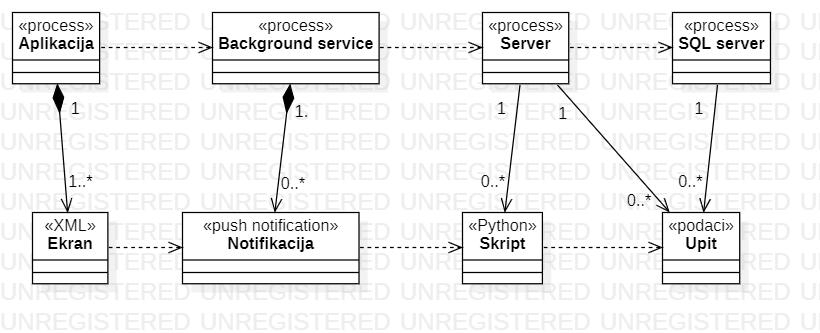
U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe spada u opis procesne arhitekture.

Aplikacija zasnovane na Android tehnologiji imaju relativno jednostavan procesni model koji delom u kontroli aplikacije na samom uređaju, a delom u kontroli servera. Sa stanovišta projektanta nije potrebno voditi računa o načinu rada servera i načinu izvršavanja skripti na njemu.

Ilustracije radi u nastavku je dat opis procesa uključenih u izvršenje Zadruga aplikacije kao Android aplikacije.

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju Zadruga aplikacije. Dijagram je opšteg tipa i može se primeniti na bilo koju Android aplikaciju koja obuhvata backend u Python-u i SQL bazu podataka.



### Pozadinski proces obaveštenja

Pozadinski proces obaveštenja je proces koji se izvršava na uređaju čak i kada Android aplikacija nije aktivna. Funkcioniše po push principu, dakle uređaj ne šalje konstantno zahteve serveru kako bi saznao da li ima novih obaveštenja, već ga server o tome direktno obaveštava. Ovo je ključna osobina ovog servisa koja kao rezultat ima uštedu mobilnih podataka, kao i sprečavanje zagušenja komunikacionog kanala.

### Server

Server je proces koji se izvršava na udaljenom računaru i opslužuje sve zahteve korisnika koje oni (implicitno) šalju putem Android aplikacije. Pre svega služi za smeštanje skripti koje sadrže logiku za pristup centralizovanoj bazi podataka, ali takođe izvršava i neke operacije koje su previše računski zahtevne, kako iste ne bi morale da se izvršavaju na samom mobilnom uređaju, koji po definiciji ima slabiju konfiguraciju, pa samim tim utičemo i na performanse i potrošnju baterije.

### Aplikacija

Aplikacija je proces koji se odnosi na samu Android aplikaciju koju korisnici pokreću na svom uređaju. Sastoji se od većeg broja ekrana koji su logički međusobno povezani putem dugmića, menija i slično. Svaki ekran se sastoji od svoje grafičke reprezentacije (u .xml fajlovima), kao i deo biznis logike koji je neophodno izvršavati na samom uređaju.

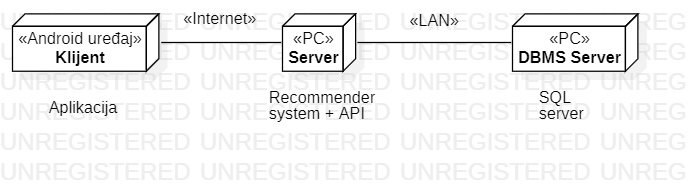
### SQL Server

MySQL Server je proces koji izvršava funkcionalnost MySQL sistema za upravljanje bazama podataka. Ovaj proces može konkurentno da prihvati određen broj upita, izvrši ih nad bazom podataka i vrati rezultate procesu koji je upite postavio.

# Pogled na raspoređivanje sistema

Pogled na raspoređivanje sistema prikazuje različite fizičke čvorove za najopštiju konfiguraciju sistema. Fizičkim čvorovima koji predstavljaju procesore vrši se dodeljivanje identifikovanih procesa.

Na sledećoj slici dat je UML dijagram raspoređivanja Zadruga aplikacije.



## Klijent

Android uređaj na kojem se izvršava aplikacija Zadruga. Aplikacija zavisi od Internet konekcije u pogledu funkcionalnosti jer na taj način pribavlja informacije od servera.

## Server

Računar na kome se izvršava server istovremeno može da opslužuje veći broj klijenata. Server može pristupati podacima koji se nalaze u bazi podataka korišćenjem API poziva koji su mu prosleđeni putem HTTP zahteva. Pored ovih funkcionalnosti na serveru se realizuje i funkcionalnost aplikacije - Recommender system. U najopštioj konfiguraciji DBMS se izvršava na posebnoj mašini koja je sa datim serverom u lokalnoj mreži (LAN).

## DBMS server

DBMS server je računar na kome se izvršava MySQL Server proces koji realizuje funkcionalnost sistema za upravljanje bazama podataka. Zbog sigurnosti podataka koji se na ovom računaru čuvaju pristup bazi je ograničen samo na računare iz lokalne mreže (LAN).

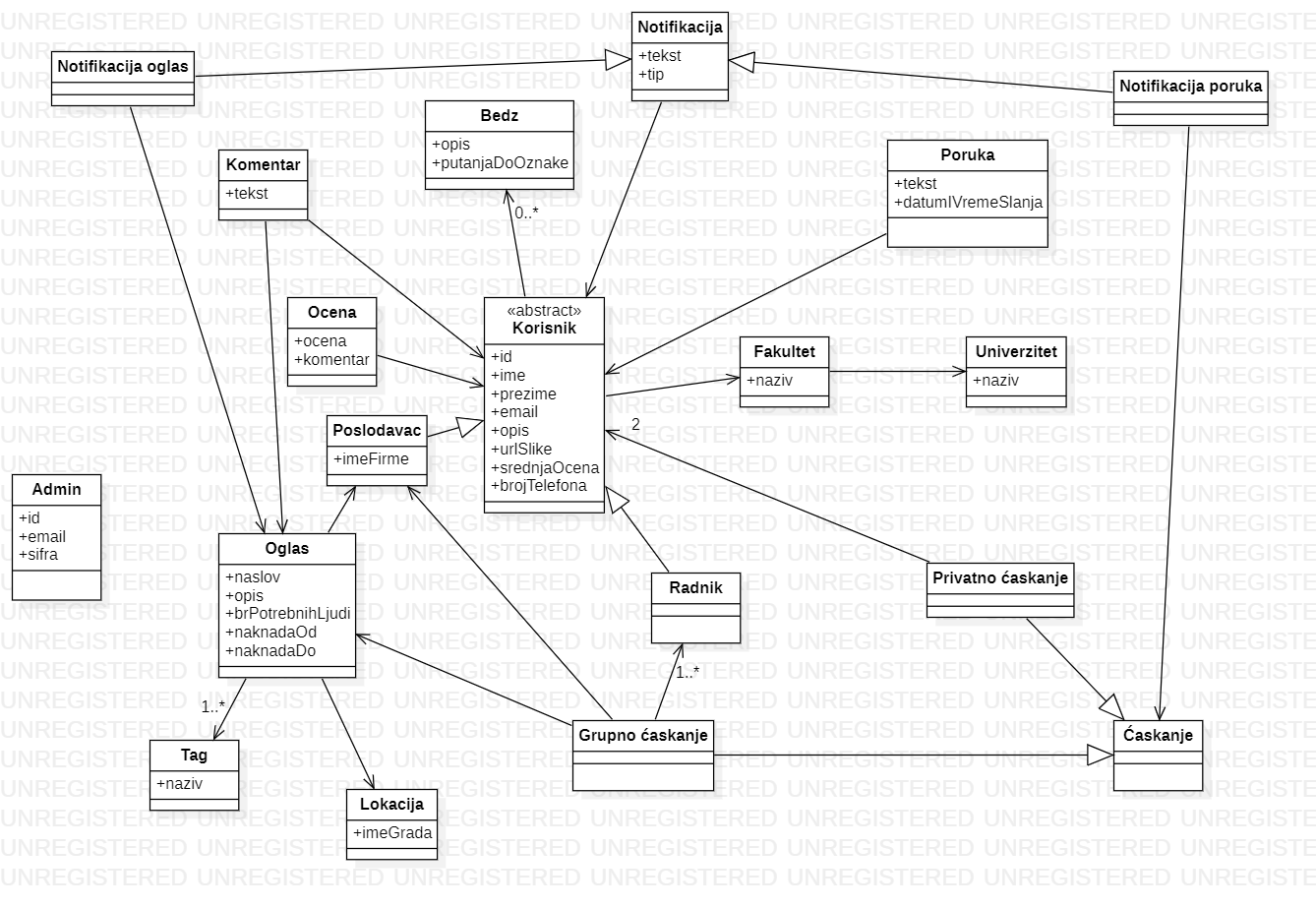
# Pogled na implementaciju sistema

Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju Zadruga aplikacije ovaj odeljak sadrži model domena, šemu baze podataka i prikaz komponenti sistema razvrstanih u ranije identifikovane pakete.

## Model domena

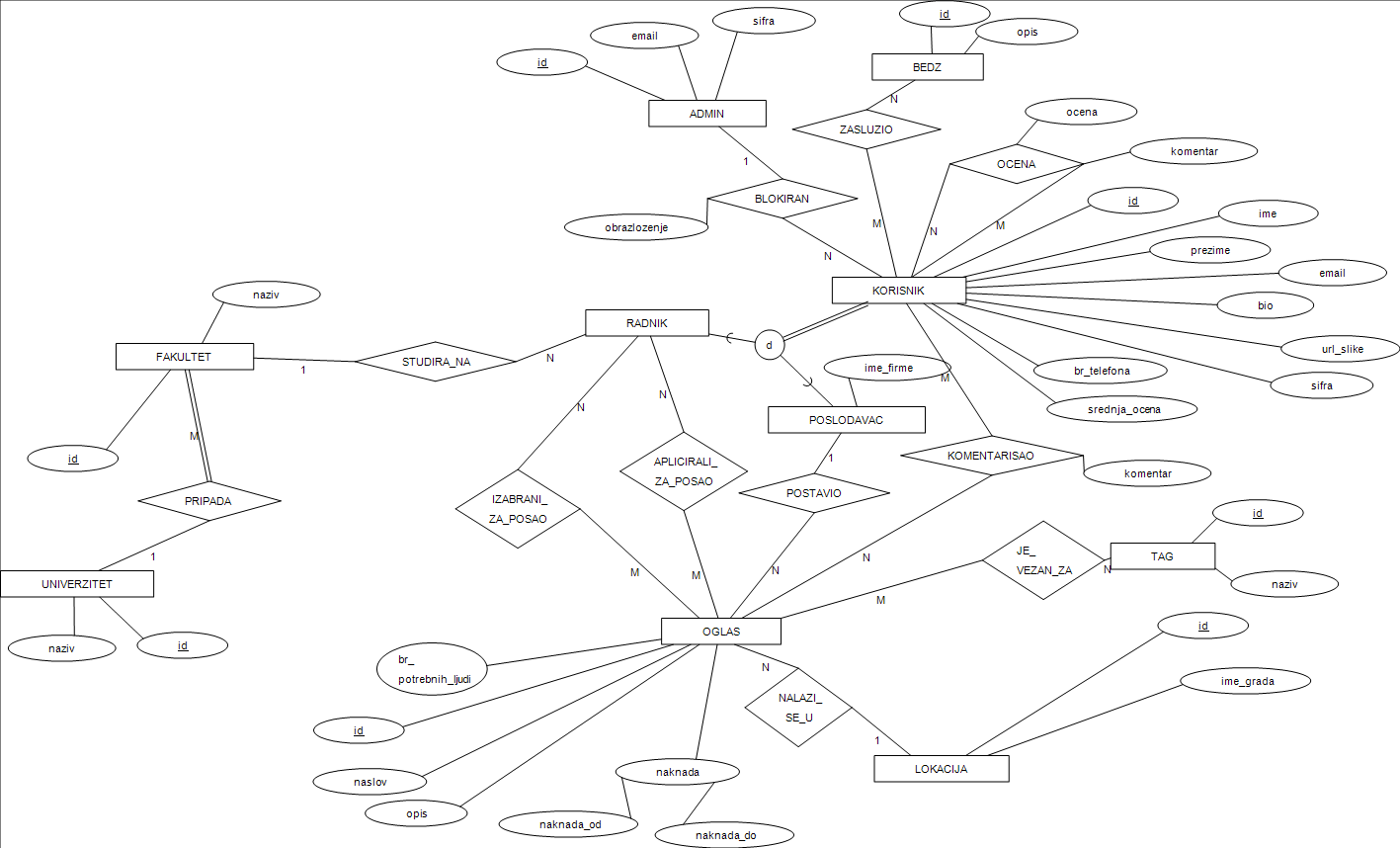
Model domena za koji se aplikacija Zadruga projektuje je ilustrovan UML dijagramom klasa. U njemu su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.

Model domena predstavlja osnovu za projektovanje baze podataka, ali i identifikaciju nekih od komponenti (Java klasa) koje će biti implementirane.



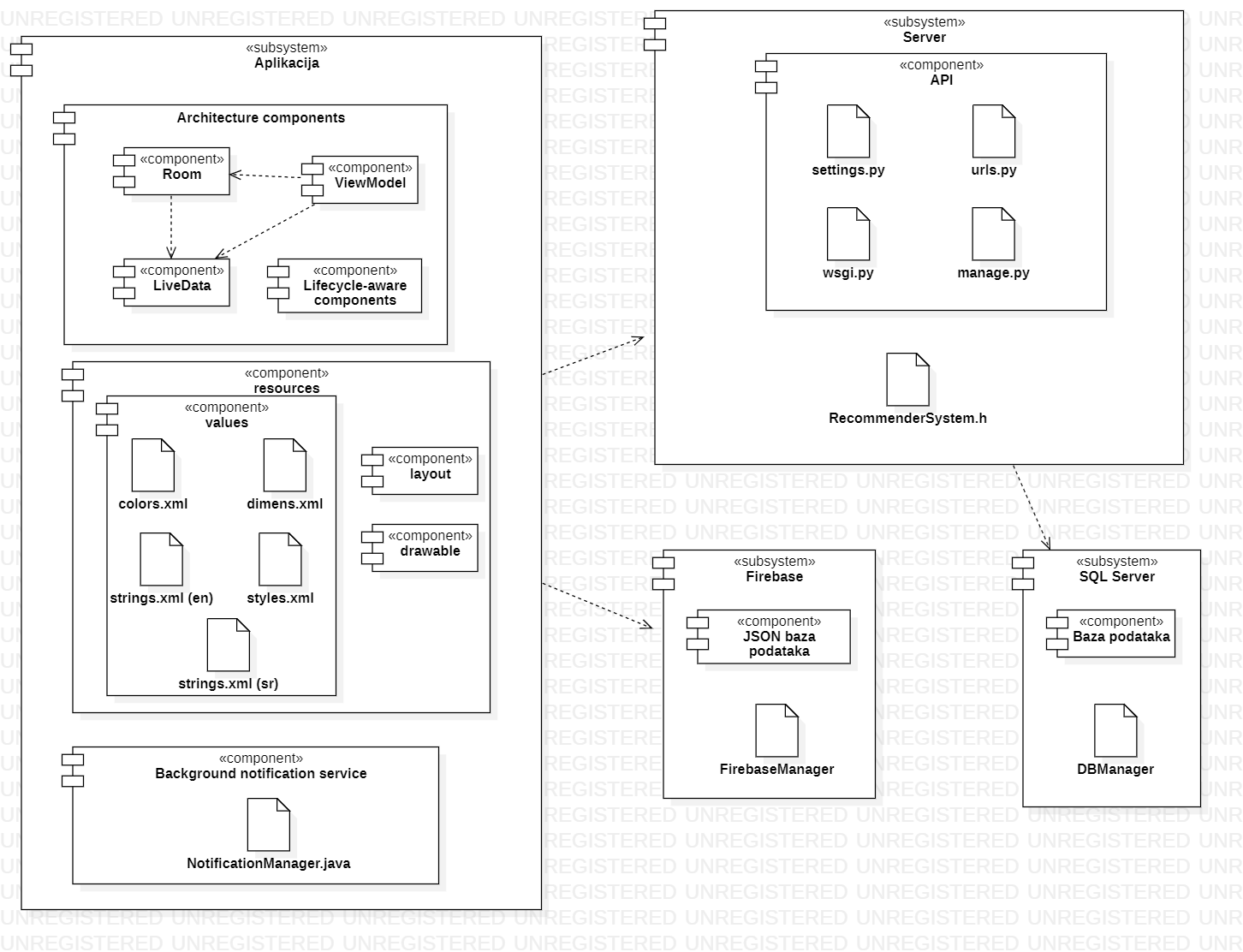
## Šema baze podataka

Detaljna šema baze podataka je prikazana na sledećem dijagramu. Baza podataka i dijagram su kreirani korišćenjem *#EER ,* program koji nam je fakultet omogućio da koristimo za iscrtavanje EER dijagrama, dok je implementacija baze odrađena u SQL-u.



## Komponente sistema

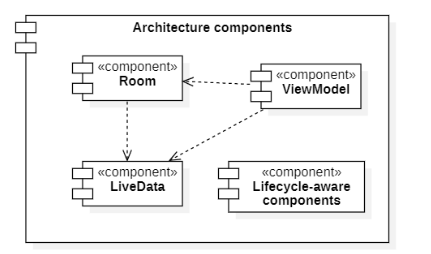
Komponente sistema aplikacije Zadruzga su podsistemi: Aplikacija, Zadruga API Server, Firebase i SQL Server, čije će komponente biti date po arhitekturnim slojevima. Za ilustraciju će biti korišćeni UML dijagrami komponenti.



### Komponente za korisnički interfejs

Komponente korisničkog interfejsa podeljenje su u dve veće kategorije: Arhitekturne komponente i Resursi.

#### Arhitekturne komponente



Ove komponente predstavljaju kolekciju biblioteka koje se koriste u svrhu kreiranja aplikacija kojima se može lako upravljati i koje poseduju mogućnost testiranja i podložne su lakim promenama ukoliko dođe do neke grešake.

U okviru komponenti koje čine arhitekturu aplikacije sadržano je još četiri potkomponente: Room, ViewModel, LiveData i Lifecycle-aware components.

###### Lifecycle-aware components

Ova komponenta reaguje na promene lifecycle statusa nekih drugih komponenti unutar same aplikacije, kao što su activities i fragments. Ona pospešuje bolju arhitekturu aplikacije u pogledu bolje organizacije i smanjenja iskorišćenosti memorijskog prostora. Ova komponenta ne utiče direktno ni na jednu drugu komponentu u okviru Architecture Components.

###### Room

Komponenta Room se koristi za lokalno skladištenje podataka. Funkcionalnost ove komponente se ogleda u tome što bitne podatke kešira, tako da ukoliko korisnik nema trenutno Internet konekciju još uvek može da pretražuje ili pristupa keširanim podacima. Room “dodaje” aplikaciji još jedan apstraktni sloj korišćenjem SQLite-a, tj. Room predstavlja SQLite objektno-mapiranu biblioteku, koja prevodi podatke iz SQLite tabela u Java objekte.

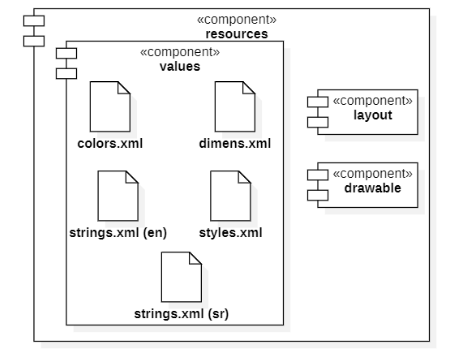
###### LiveData

LiveData je komponenta koja prati gorenavedeni lifecycle status ostalih komponenti aplikacije i ukoliko dođe do neke promene obaveštava samo one komponente koje poseduju “aktivni” lifecycle status.

###### ViewModel

Ova komponenta upravlja i skladišti UI podatke koji opstaju i nakon nekih promena u konfiguraciji, kao što je npr. rotacija ekrana.

#### Resursi



Resurse predstavljaju dodatni fajlovi koji se koriste za odredjivanje razmeštaja i sadržaja ekrana aplikacije uključujući palete, tekst, slike i sl. Kako su nezavisni od same logike aplikacije, izdvajaju se u posebnu komponentu kako bi se lakše pratili, održavali i koristili.

###### Drawables

Ovoj grupi pripadaju svi grafički resursi odnosno oni koji se mogu iscrtati na ekranu. To su mahom geometrijski oblici, bitmape, slike i sl.

###### Layout

Layout resursi, kao što samo ime kaže, služe za određivanje razmeštaja ekrana, dakle skladište informacije o arhitekturi stranica koje će se prikazivati unutar aplikacije.

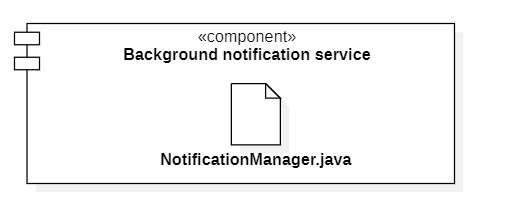
###### Values

* Colors.xml – sadrži definicije boja koje će biti deo korisničkog interfejsa
* Style.xml – sadrži definicije formata i izgleda korisničkog interfejsa
* Dimens.xml – sadrži definicije dimenzija koje se odnose na veličinu fonta ili veličinu različitih elemenata korisničkog interfejsa.
* Strings.xml (en) – čuva sadržaj tekstualnih elemenata korisničkog interfejsa na engleskom jeziku
* Strings.xml (sr) – čuva sadržaj tekstualnih elemenata korisničkog interfejsa na srpskom jeziku

### Komponente za aplikacionu logiku

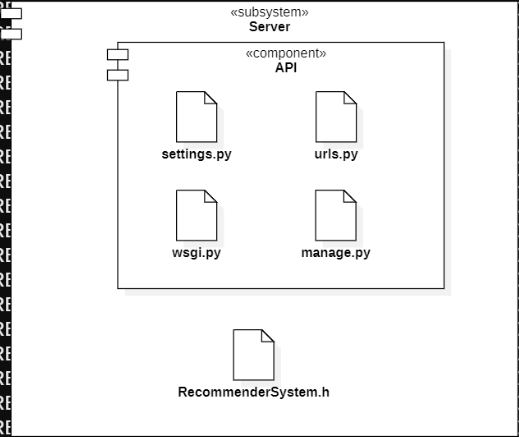
Komponente za aplikacionu logiku podeljenje su u dve veće kategorije: Servis za pozadinske notifikacije i API Server.

#### Servis za pozadinske notifikacije



Notifikacije predstavljaju poruke koje Android prikazuje izvan korisničkog interfejsa aplikacije kako bi obavestio korisnika o nekim događajima koji su mu od interesa. Korisnik pritiskom na notifikaciju da otvori aplikaciju ili čak preduzme akciju određenu tom notifikacijom. Naša aplikacija će, konkretno, korisnike obaveštavati o komunikaciji sa drugim korisnicima, a povremeno će radnicima slati obaveštenja o poslovima koji bi ih možda zanimali.

#### Zadruga API Server



Zadruga API Server je komponenta koja sadrži ključne funkcionalnosti vezane za samu biznis logiku naše aplikacije. Pre svega, treba napomenuti da je glavni deo rađen u Python framework-u Django. Unutar bloka API su prikazani osnovni moduli koji čine srž jedne Django aplikacije.

* settings.py - glavni modul za podešavanja aplikacije – instaliranje dodatnih biblioteka, podešavanje dozvoljenih hostova (CORS), pamćenje parametara za konekciju sa bazom...
* urls.py - modul čija je glavna uloga da obezbedi mapiranje između pojedinih URL adresa i odgovarajućih fajlova koji treba da opsluže zahteve sa tih adresa
* wsgi.py - modul koji sadrži podešavanja vezana za deployment
* manage.py - ovaj modul pruža mogućnost pozivanja čestih komandi putem komandne linije – startovanje i gašenje servera, migracije...

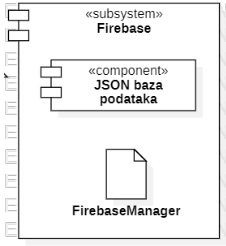
Zbog ograničenog prostora, prikazane su samo najbitniji moduli, koji pre svega omogućavaju komunikaciju našeg API-ja sa spoljašnjim svetom, odnosno njegovim konzumentima, što je u ovom slučaju Android aplikacija. Osim ovih, prisutan je još veliki broj modula koji obezbeđuje biznis logiku - modeli, kontroleri, klase za komunkaciju sa bazom...

Jedan delić je ipak izdvojen iz ovog paketa, a to je sistem za preporuku. Korisnicima je najbitniji brz odziv aplikacije i ne bi voleli da predugo čekaju da im se učitaju pojedini ekrani. Zbog toga se trudimo da vreme odziva servera bude što moguće kraće. Najveći deo ovog vremena odlazi na putovanje podataka do servera i nazad do Android uređaja. Međutim, ukoliko je kod koji treba izvršiti prilikom svakog zahteva suviše komplikovan, može se desiti da i taj deo drastično utiče na ovo vreme. U našem slučaju, to usko grlo je sistem za preporuku. Kako je Python interpreterski jezik, on se mnogo sporije izvršava nego, recimo, C++, koji je kompajlerski jezik i koji ćemo koristiti kako bismo donekle doskočili ovom problemu.

### Komponente za pristup podacima

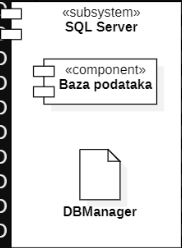
U komponente za pristup podacima vezane za aplikaciju Zadruga razmatraćemo dva podsistema: SQL server i Firebase.

#### Firebase



Firebase je Backend-as-a-Service Google-ova platforma koja se koristi za pravljenje mobilnih i web aplikacija. Firebase podsistem sadrži još jednu komponentu - JSON bazu podataka pomoću koje pamtimo, dodajemo i “povlačimo” podatke vezane za notifikacije i poruke koje će korisnici razmenjivati. Ovaj podsistem ne zavisi od servera i SQL servera.

#### SQL Server



SQL Server je komponenta koja služi za pamćenje i manipulaciju svim neophodnim podacima, osim notifikacijama i porukama. Ova komponenta može da bude pokrenuta na istom računaru kao i Zadruga API server, ali ne obavezno, ukoliko to iz bilo kog razloga nije moguće - zbog toga ih razmatramo odvojeno.

Sastoji se od dve potkomponente:

* + Baza podataka – ova komponenta se odnosi na samo skladištenje podataka, tj. skup fajlova čiji su sadržaj podaci koje pamtimo
  + DBManager – DBMS koji pruža sve osnovne usluge za kreiranje strukture, unos i manipulaciju podacima na efikasan način.

Kao što je već naglašeno, Zadruga API server se oslanja na ovu komponentu kako bi pouzdano i efikasno opslužio svoje klijente.

# Performanse

Izabrana arhitektura softvera podržava zahteve u pogledu broja korisnika koji mogu simultano pristupati sistemu i vremena odziva za pristup bazi podataka specificirane u zahtevima u pogledu performansi [4]:

1. Sistem će da podrži do 10000 simultanih pristupa korisnika aplikaciji.
2. Vreme potrebno za pristupanje bazi podataka u cilju izvršenje nekog upita ne sme da bude veće od 5 sekundi.

Zahtevane performanse su zadovoljene izborom tehnologija na kojima će sistem biti razvijen i definisane hardverske platforme [4].

# Kvalitet

Izabrana arhitektura softvera podržava zahteve u pogledu dostupnosti i srednjeg vremena između otkaza specificirane u zahtevima u pogledu pouzdanosti [4]:

1. Zadruga aplikacija će biti dostupan 24 časa dnevno, 7 dana u nedelji. Vreme kada aplikacija nije dostupan ne sme da pređe 10%.
2. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 120 sati.