

Prolećni semestar 2018/2019

Implementacija HTTP protokola u aplikaciji za razmenu poruka

CS102

Objekti i apstrakcija podataka

Projektna dokumentacija

Student:

Nikola Tasić 3698

Asistent:

Jovana Jovic

**Sadržaj**

[1.](#_gjdgxs) Struktura 3

[2.](#_30j0zll) Uvod 4

3[. Analiza](#_1fob9te) 4

4[. Zaključak](#_1fob9te)  5

5[. Reference](#_3znysh7) 5

# Struktura projkekta

├── src

│ ├── client

│ │ ├── Client.java

│ │ ├── components

│ │ │ ├── ComposeComponent.java

│ │ │ ├── MessagesComponent.java

│ │ │ ├── SidebarComponent.java

│ │ │ └── TopControls.java

│ │ ├── scenes

│ │ │ ├── HomeScene.java

│ │ │ ├── LoginScene.java

│ │ │ └── RegisterScene.java

│ │ └── state

│ │ └── State.java

│ ├── http

│ │ ├── HttpStatus.java

│ │ ├── Request.java

│ │ ├── RequestTypes.java

│ │ ├── Response.java

│ │ └── StatusCodes.java

│ ├── server

│ │ ├── database

│ │ │ └── DBController.java

│ │ ├── handler

│ │ │ ├── Handler.java

│ │ │ └── HandlerUtils.java

│ │ ├── message

│ │ │ └── Message.java

│ │ ├── Server.java

│ │ └── user

│ │ ├── User.java

│ │ └── UserUtils.java

│ └── utils

│ └── Encryption.java

└── tests

├── RequestTest.java

├── ResponseTest.java

├── StateTest.java

└── UserTest.java

# Uvod

# Aplikacija sluzi za razmenjivanje poruka izmedju vise klijenata preko zajednicke server aplikacije. Koristi SQLite3 bazu u kojoj se cuvaju podaci o registrovanim korisnicima i razmenjenim porukama. Korisnici su autentikovani preko veoma prosle implementacije authentikacionih tokena.

# Analiza

Sistem se sastoji iz dve podsistema. Multi-threaded server aplikacije i vise mogucih klijentskih aplikacija koje komuniciraju sa serverom preko HTTP protokola. Ideja o komunikaciji izmedju klijenata i servera je bila da se na najprostiji nacin razmenjuju poruke i informacije o autentikovanim korisnicima. Obzirom na to da je svaki pokusaj ove implementacije licio na HTTP prosto sam implementirao osnove tog protokola za potrebe ovog sistema. Pri pokretanju aplikacije korsniku se prikazuje Login screen i opcija da se registruje. Posle unosenja dosta nepotrebnih informacija (ideja je takodje bila da implementiram napredne SQL upite preko tih “nepotrebih informacija” ) korisnik ce uspesno biti registrovan ako ne postoji jos jedan korisnik sa takvim korisnickim imenom ili email-om u bazi. Posle uspesnog registrovanja korisnik je prebacen na glavnu scenu gde se nalaze opcije za slanje i pregled poruka zajedno sa listom korisnika na serveru. Server funkcionise tako sto pokrece loop koji ceka nove konekcije i cim se nova konekcija ostvari on pokrece novi thread koji u komene se vrsi dalje procesuiranje. Parsiranje podataka iz zahteva se vrsi preko veoma korisne klase Request koja u sebi sadrzi sve potrebne funkcije za ucitavanje objekata koji reprezentuju zahtev iz samog reader-a koji nam daje Socket objekat. Iz tog Request objekta se parsira path na koji je upucen zahtev i u zavisnosti od toga da li server ima podrzan taj path obradjuju se dalje podaci ili se salje HTTP status kod 404 “NOT FOUND”. Takodje postoji enum svih standardnih HTTP status kodova i odgovarajucih propratnih poruka. Posle potvrdjenog path-a zahteva poziva se odgovarajuca handler funkcija iz Handler klase. U svakom slucaju (obzirom na to da je u pitanju aplikacija za komunikaciju) se proverava da li je korisnik koji vrsi zahtev autentikovan. Svaki zahtev MORA da sadrzi dva HTTP headera: Token i User. Token header zadrzi enkriptovan string koji predstavlja neku vrstu korisnickog identiteta na serveru, dok je u User headeru njegovo korisnicko ime. U Token headeru se takodje posle dekripcije moze naci isto to korisnicko ime i kao i datum dokle vazi taj token (posle isteka tog vremena korisnik mora da se ponovo uloguje). Prlikom svakog neuspesnog pokusaja autentikacije salje se odgovarajuci HTTP header. Takodje prilikom uspesne autentikacije salje “Set-Cookie” header koji moze da se koristi za postavljanje Tokena u web pretrazivacu. Sto se tice slanja i pretrazivanja poruka postoji API koji omogucava korisniku da dobije samo poruke koje su namenjene njemu.

# Zakljucak

Projekat predstavlja zanimljivu osnovu za razumevanje low-level HTTP protokola i autentikacije korisnika na web serverima. Fokus projekta je vise bio na modularnosti serverske aplikacije tako da se lako dodaju nove funkcionalnosti a manje na interakciji sa bazom i korisnickom interfejsu.

# Reference

*[1]* StackOverflow, [*https://stackoverflow.com/*](https://stackoverflow.com/)

*[2]* Github, [*https://github.com/*](https://github.com/)

*[3]* LAMS, [*http://lams.metropolitan.ac.rs:8080/lams/*](http://lams.metropolitan.ac.rs:8080/lams/)

*[4]* RFC 2616, [*https://tools.ietf.org/html/rfc2616*](https://tools.ietf.org/html/rfc2616)*/*