**Факултет техничких наука у Чачку**

**Универзитета у Крагујевцу**

**Никола Радовановић**

**Развој апликације применом Spring boot-а и мрежног програмирања**

**дипломски рад**

**Чачак, октобар 2022.**

**Факултет техничких наука у Чачку**

**Универзитета у Крагујевцу**



**Развој апликације применом Spring boot-a и мрежног програмирања**

**Дипломски рад**

**Врста студија: Основне академске студије**

**Назив студијског програма: Информационе технологије**

**Предмет: Напредно објектно-оријентисано програмирање**

**Студент: Руководилац рада:**

**Никола Радовановић 54-2017 Др Олга Ристић, ванр. проф.**

**Чачак, октобар 2022.**

**Резиме**

Програмирање је делатност која постоји преко седамдесет година и захтева константо напредовање технологија које се користе. Иако технологије константно напредују, основе програмирања су махом остале исте откако се појавио први објектно оријентисани језик.

Java програмски језик је своју популарност стекао захваљујући својој портабилности, односно не завишношћу од система који је покреће. Због тога се разноврсност примене Jave креће од једноставних десктоп апликација, преко мобилних апликација до сложених веб сервера.

Због популарности Jave и зато што постоје програмска решења за проблеме са којима се програмери често сусрећу, појављује се Spring фрејморк. Spring je фрејмворк за јава програмски језик који помаже да се програмери фокусирају на нова решења тако што обезбеђује комплетну инфраструктуру апликације. Другим речима програмери се могу фокусирати на проблемске захтеве саме апликације уместо да се прво баве израдом инфраструктуре.

Инфрастуктура коју Spring обезбеђује укључује и конекцију путем интернета са другим апликацијама кроз веб сокет. Као и протокол за слање порука

Кључне речи:

* Java
* Spring
* Spring Boot
* Фрејмворк
* Веб сокет

**Summary**

Keywords:

* Java
* Spring
* Spring Boot
* Framework
* Web socket
* Application

# Садржај

[1. Садржај 4](#_Toc115038788)

[1. Увод 5](#_Toc115038789)

[2. Spring Framework 7](#_Toc115038790)

[2.1 Разлика између спринг и спринг боот фрејмворка 7](#_Toc115038791)

[2.2 Креирање спринг боот пројекта 7](#_Toc115038792)

[3. План развоја апликације 9](#_Toc115038793)

[3.1 Коришћени алати и програмска подршка 9](#_Toc115038794)

[3.1.1 Spring Boot 9](#_Toc115038795)

[3.1.2 Maven 9](#_Toc115038796)

[3.1.3 Websocket 10](#_Toc115038797)

[3.1.4 STOMP 10](#_Toc115038798)

[3.1.5 IntelliJ IDEA 10](#_Toc115038799)

[3.1.6 JavaScript 11](#_Toc115038800)

[3.1.7 JPA 11](#_Toc115038801)

[3.2Коришћене зависности 11](#_Toc115038802)

[3.3Глобални опис 11](#_Toc115038803)

[3.1.1 Перспектива система 11](#_Toc115038804)

[3.1.2 Карактеристике система 11](#_Toc115038805)

[3.4 Дефинисање захтева апликације 11](#_Toc115038806)

[3.4.1 Функционални захтеви 11](#_Toc115038807)

[1.1.1 Нефункционални захтеви 13](#_Toc115038808)

[1.1.2 Процесни захтеви 14](#_Toc115038809)

[2. Архитектура апликације 15](#_Toc115038810)

[2.1 Spring Boot архитектура 15](#_Toc115038811)

[2.2 Архитектура веб сокета 16](#_Toc115038812)

[2.3 STOMP архитектура 16](#_Toc115038813)

[2.4 База података 16](#_Toc115038814)

[2.5 Генерална архитектура 16](#_Toc115038815)

[3. Изглед апликације 17](#_Toc115038816)

[4. Закључак 20](#_Toc115038817)

[5. Закључак 21](#_Toc115038818)

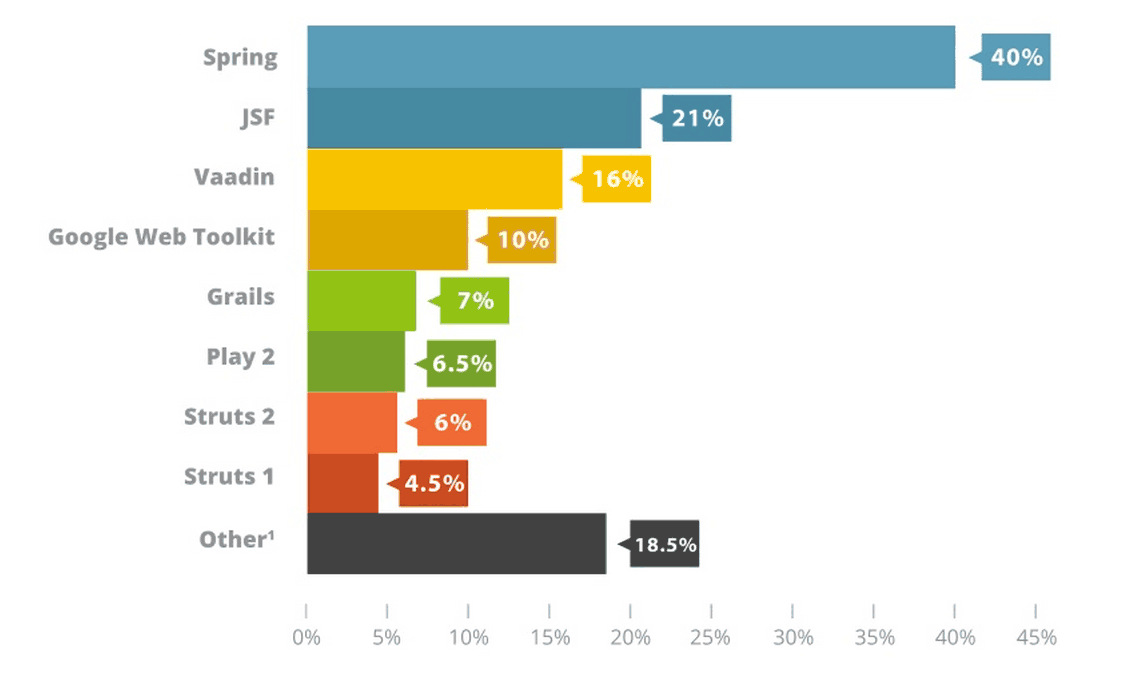
[6. Списак слика 22](#_Toc115038819)

# Увод

За развијање комплексних софтвера потребно је увести доста компоненти у саму структуру апликације, као што су на пример: База података, модул који се повезује са осталим апликацијама путем интернета, контролери... Како би се смањило време израде оваквих апликација појавила се потреба за софтвером који ће олакшати повезивање свих делова апликације. Готово за сваки програмски језик постоји фрејмворк. У листи испод приказани су неки од најпопуларнијих фрејмворка као и њихови програмски језици.

1. ASP .NET CORE (C#)
2. Bootstrap (HTML, CSS)
3. Rails (Ruby)
4. Laravel (PHP)
5. Angular (JavaScript)
6. Spring (Java)

За потребе овог рада креирана је апликација у Јава програмском језику уз помоћ Spring фрејмворка. Spring није једини фрејмворк за Јаву, али је убедљиво најпопуларнији због своје флексибилности, скалабилности и лакоће коришћења. На слици испод приказанa је популарност јава фрејмворка према броју корисника у процентима (<https://www.jrebel.com/blog/java-web-framework-usage-stats>).



Као што се може приметити, спринг је тренутно најпопуларнији јава фрејмворк. Због својих предности предвиђа се још већи раст његове популарности. Неке од ових предности су:

* Смањује време израде софтвера, а повећава ефикасност.
* Лако повезивање са базама података
* Интеграција са Spring екосистемом што омогућава коришћење Spring Data, Security, ORM и JDBC
* Уштеда времена при писању кода

Циљ дипломског рада је да прикаже архитектуре спринг боот фрејмворка, веб сокета, стомп протокола и генералну архитектуру апликације, на примеру апликације за групно дописивање. Апликација је поприлично једноставне функционалности, али се у позадини дешава много више ствари.

# Spring Framework

Спринг се први пут помиње у књизи Рода Џонсона ,,*Expert One-on-One J2EE Design and* 2002. године. Род је развио Спринг како би олакшао развој серверских јава апликација. Спринг је пуштен у рад 2003. године под лиценцом Apache 2.0.

Спринг боот настаје десетак година касније тако што је Мајк Јангстром на форуму предложио унапређење спринг-а кроз обједињење микросервиса. Глсвни покретач овог пројекта је Фил Веб из Спринг развојног тима, који прву верзију спринг боот-а објављује 2014. године

## Разлика између спринг и спринг боот фрејмворка

Спринг је најпопуларнији фрејмворк за развој апликација у Јави. Међутим, током последњих неколико година, Спринг је постао све сложенији због додатних функционалности. Започињање новог пројекта обично захтева дуготрајан процес и одузима много времена.

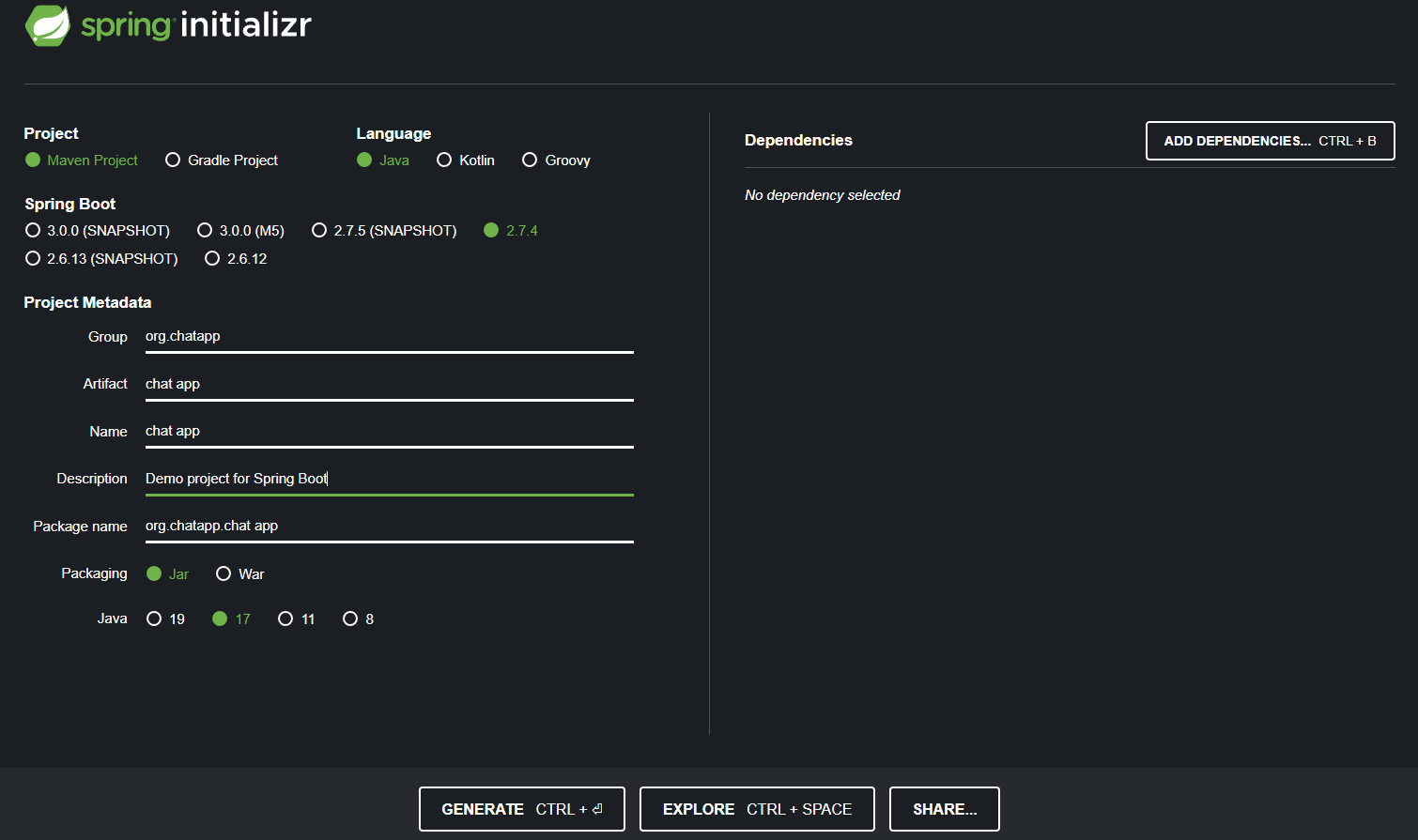
Да би се избегло почињање од нуле и да би се уштедело време, уведен је Спринг Боот. Спринг Боот користи Спринг као свој темељни слој и омогућава програмерима да направе самосталну апликацију без конфигурације или са минималном конфигурацијом. Посебно је усмерен ка изградњи веб апликација и микросервиса.

Спринг боот омогућава лакшу конфигурацију и мање посла око инфраструкруре пројекта. Свакако се базина на основном спринг-у и не може функционисати без њега. За ову апликцаију коришћен је спринг боот верзије 2.7.4.

## Креирање спринг боот пројекта

То ће бити Спринг Боот Мавен пројекат најлакше се генерише уз помоћ Spring Initializr (<https://start.spring.io/>). Ово је најбржи начин за креирање Спринг апликације, а ИнтеллиЈ ИДЕА обезбеђује додатак у оквиру самог софтвера за спринг инитиализр.

У суштини Spring Initializr је веб апликација која аутоматски генерише Spring Boot структуру пројекта за нас. Не генерише код, али се доста смањује посао око конфигурације. На слици 2 приказан је изглед ове веб апликације.



У Project делу дефинише се тип пројекта, затим је потребно одабрати програмски језик. Можемо да бирамо између Jave, Kotlin-a и Groovy. Верзију спринг боот фрејмворка селектујемо испод истоимене секције. Снапсхоот верзије су тестне верзије и нису увек стабилне. Последња стабилна верзија је 2.7.4. Потребно је такође попунити податке о апликацији и одабрати врсту пакета и верзију Јаве.

У депендециес делу се селектују зависности, односно које сервисе желимо додати у апликацију. Најчешће коришћене зависности су: Spring Web, Spring Security, Spring JPA…

Када се заврши са основним конфигурисањем приојекта, све што је преостало јесте да се преузме генерисани пројекат. Пројекат који се преузима је аутоматски повезан са свим селектованим сервисима, на пример са базом података, конфигурсан је да користи интернет ако је додата зависност за веб сокет итд.

# План развоја апликације

## Коришћени алати и програмска подршка

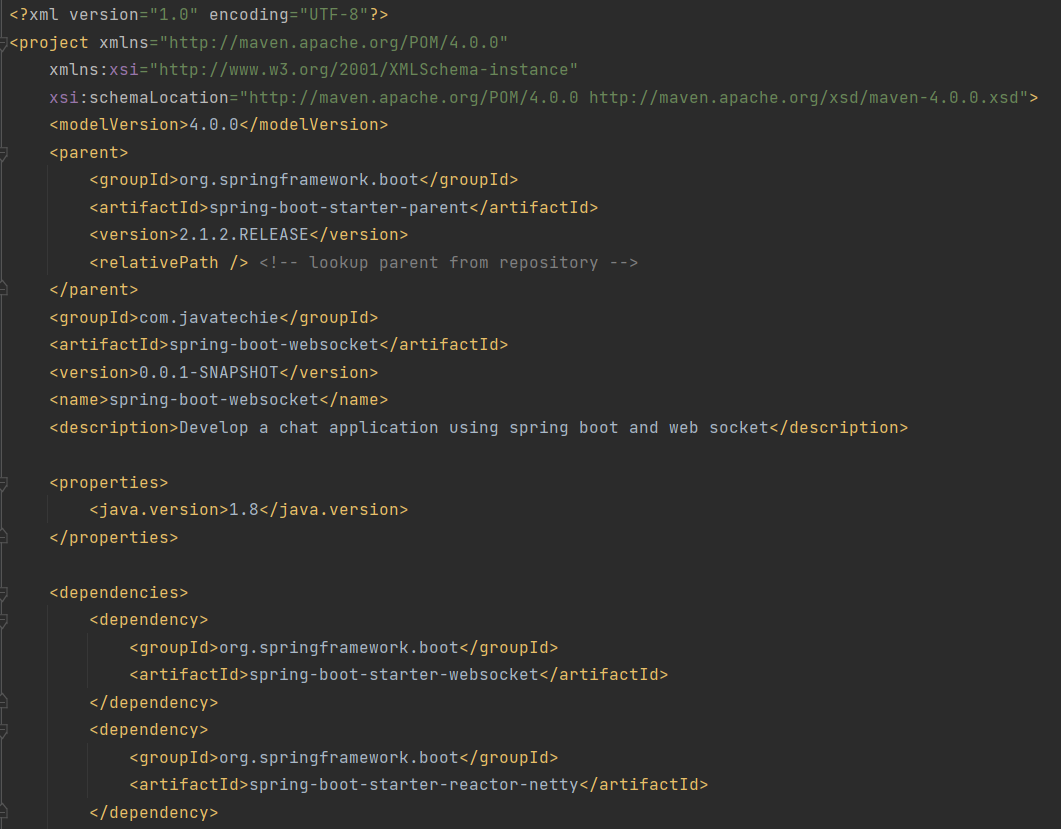
### Spring Boot

### Maven

Мавен је алатка за прављење и управљање пројектима која се обично користи у фрејмворцима изграђеним у Јави. Развила га је Апацхе Софтваре Фоундатион.

Пре него што је Мавен ушао у пројекте, све зависности, које нису ништа друго него ЈАР датотеке, морале су се ручно додавати у фрејмворк. Такође, требали смо се побринути за надоградњу софтвера у нашем пројекту.

Мавеном управља датотека пројектног модела (POM). У овој датотеци се додају све зависности и конфигуришу плагинови који се додају, као и основне конфигурације пројекта попут имена, верзије јаве и пакета.



### Websocket

Веб сокет је двосмерна конекција између веб претраживача и сервера. Једном када се успостави веб сокет конекција, конекција остаје отворена све док клијент или сервер не одлуче да затворе ову везу.

Типичан случај употребе може бити када апликација укључује више корисника који међусобно комуницирају, на пример у апликацији за допицивање.

Веб сокет се у Спринг додаје завишношћу „spring-boot-starter-web-socket“.

### STOMP

STOMP (Simple Text Orientated Messaging Protocol) Једноставни текстуално оријентисан протокол за поруке је HTTP протокол који фунционише са веб сокетом тако што додаје заглавље на поруку о омогућава једноставнију комуиникацију.

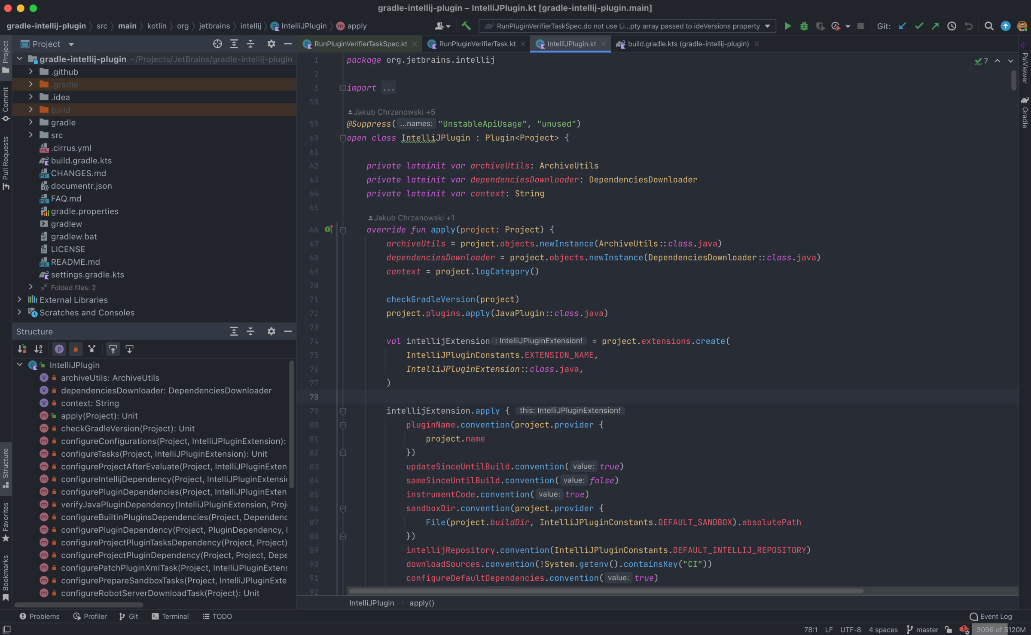
STOMP је лако имплементирати уз помоћ стомп брокера, који представљају главни контролни објекат овог протокола. Брокер је потребно имплементирати и на серверској и на клијентској страни како би комуникација била могућа.

### IntelliJ IDEA

IntelliJ IDEA је интегрирано развојно окружење (eng. IDE – integrated development environment) за развој софтвера. Развијен је 2001. године од стране Jetbrains-а као једно од првих интегрираних развојних окружења за Јаву. Пружа подршку и за друге језике, неки од њих су SQL, HTML, Javascript.

Две доступне верзије овог

окружења, Community edition и Ultimate edition, употребљавају се за развој програма у комерцијалне сврхе. За разлику од бесплатне верзије, Community edition, Ultimate edition нуди бројне погодности везене уз лакши развој апликација у Спринг Боот-у.



### JavaScript

JavaScript је објектно оријентирани скриптни језик који је развијен од стране Netscape компаније како би омогућио веб дизајнерима да дизајнирају интерактивније веб странице. Уобичајене HTML странице су погодне за приказивање статичког садржаја, попут једноставних слика или текста. Међутим, данас се странице са статичким садржајем прилично ретко виђају.

У овом пројекту ЈаваСцрипт је коришћен да повеже клијентски део апликације са сервером. За ове потребе користи се концепт сокета у ЈаваСцрипт-у кроз SockJS.

SockJS отвара конекциони порт и преко ИП адресе се повезује сервером. На овај начин сервер путем СТОМП протокола може да шаље поруке.

### JPA

Спецификација JPA (Java Persistence API) омогућава објектно-релационо мапирање у Java апликацијама које доста упрошћава управљање подацима.

Поред тога дефинисан је потпуно нови програмски модел за писање перзистентних класа, класа које своје стање могу трајно сачувати у бази података. И ова спецификација, названа JPA омогућава писање POJO (Plain Old Java Object) објеката који у великој мери преузимају функционалност ентитета уз поједностављен развој. Овај нови тип компоненте се уобичајено зове ентитет и пише се као POJO објекат доупуњен анотацијама које описују начин складиштења објеката у бази података.

У овој апликацији база података је јако једноставна, састоји се од само једног ентитета. Сложенија база података није потребна јер служи искључиво да би новоконектовани корисници могли да виде претходне порт

## 3.2Коришћене зависности

## 3.3Глобални опис

### Перспектива система

.

### Карактеристике система

## Дефинисање захтева апликације

### Функционални захтеви

У табели 2. приказани су функционални захтеви за администратора.

*Табела. Функционални захтеви за корисника*

|  |  |  |  |
| --- | --- | --- | --- |
| Редни број | Захтев | Опис | Приоритет |
| 1. | Логовање | Систем треба да обезбеди приступ аутентификованом кориснику | 1 |
| 2. | Приказ претходних порука | Систем треба да обезбеди да корисник види поруке које су послате пре него што се конектовао | 1 |
| 3. | Слање порука | Систем треба да проследи свим осталим корисницима поруку коју је корисник послао | 1 |

### Нефункционални захтеви

У табели 2. приказани су нефункционални захтеви.

*Табела 3. Нефункционални захтеви система*

|  |  |  |  |
| --- | --- | --- | --- |
| Редни број | Захтев | Опис | Приоритет |
| 1. | Перформансе | Систем треба да одговара на корисникове захтеве у што краћем року | 1 |
| 2. | Безбедност | Систем треба да забрани приступ неаутентификованим корисницима | 1 |
| 3. | Поузданост | Систем треба да буде доступан 24/7 | 1 |
| 4. | Ажурност | Уколико се на систему открију грешке потребно их је уклонити у што краћем року | 1 |
| 5. | Интуитивност | Навигација кроз систем треба да буде што једноставнија | 1 |
| 6. | Windows подршка | Систем ће подржавати Windows оперативне системе | 1 |
| 7. | Скалабилност | Систем ће подржавати приступ више корисника истовремено | 1 |
| 8. | Дизајн | Систем ће имати савремени дизајн | 2 |
| 9. | Подршка за различите прегледаче | Систему ће моћи да се приступи са различитх веб прегледача | 2 |
| 10. | Техничка подршка корисницима | Корисници ће на располагању имати техничку подршку на даљину | 3 |

### Процесни захтеви

У табели 4. Приказани су процесни захтеви.

*Табела 4. Процесни захтеви система*

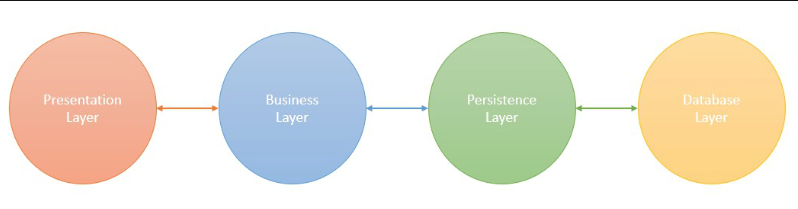
|  |  |  |  |
| --- | --- | --- | --- |
| Редни број | Захтев | Опис | Приоритет |
| 1. | Програмски језик | Апликација ће бити рађена у Java програмском језику | 1 |
| 2. | Фрејмворк | Апликација ће користити Spring Boot фрејмворк | 1 |
| 3. | База података | Апликација ће користити ЈПА базу података, и у њој ће бити чувани сви подаци | 1 |
| 5. | Фронтенд | Фронтенд део система биће кодиран у HTML-у уз подршку Bootstrap-a и JavaScript-a | 1 |
| 6. | Развојно окружење | За развој апликације користиће се InteliJ развојно окружење | 2 |

# Архитектура апликације

## Spring Boot архитектура

Спринг боот користи вишеслојну архитектуру, у којој сваки слој комуницира са слојем изнад и испод у хијерархији. Четири слоја које користи спринг боот су:

* Слој презентације
* Бизнис слој
* Истрајни слој
* Слој базе података



Слој презентације се налази на врху хијерархије и он је углавном задужен за манипулацију ХТТП захтева које шаље клијент. Овај слој има могућност интерпретације JSON (Java Script Object Notion) објеката. Овде се такође налази кориснички интерфејс апликације. Када слој презентације прими захтев, прослеђује га бизнис слоју.

Бизнис слој је задужен за функционалности као што су валидација, ауторизација и бизнис логика. У овом слоју се налази код који представља логику шта се дешава у којим ситуацијама. Овде се такође налази безбедносни део апликације, тако да се спречава неауторизовани приступ подацима.

Истрајни слој је слој који комуницира између бизнис слоја и слоја базе података и према томе је задужен за превођење рачунарских објеката у колоне у бази података и обрнуто. У овом слују се дефинишу упити који ће се вршити над базом податка и еквивалентно је са репозиторијским класама у другим архитектурама. Сва складишна логика чува се у овом слоју.

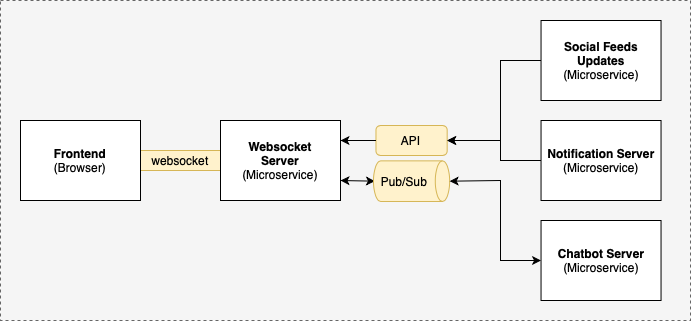
У последњем слоју као што му име каже налази се база података. Базе података могу бити MySql, MongoDB, Postgre или било које друге. У овом слоју се извршавају CRUD (Create Read Update Delete) операције. Састоји се од технологија које омогућавају нашој мобилној или веб апликацији могућност да чува и преузима податке из базе.

Доста програмера се одлучује за ову архитектуру апликација зато што се лако проналазе грешке и одржавање и ажурирање су доста једноставнији. Када се Спринг боот не би користио, програмери би морали сами имплементирати овакву архитектуру и ручно повезали слојеве. Ово је још један од разлога који чине спринг боот најпопуларнијим фрејмворком.

## Архитектура веб сокета

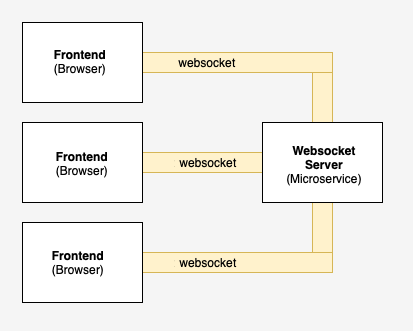
У мрежном програмирањи сокет је API који представља крајњу тачку двосмерног комуницирања између две апликације путем интернета. Веб сокет са друге стране обезбеђује комуникацију између клијента и сервера.

На слици испод приказана је архитектура веб апликације реализоване уз помоћ веб сокета. Веб сокет у овој архитектури представља канал за комуникацију између сервера и клијента. Да не постоји веб сокет, клијент би био преоптерећен функционалностима за комуникацију са свим сервисима.



Сервер у овом случају путем веб сокета клијенту одговара на захтеве које је он послао, проваерава права и аутентификацију, затим прослеђује одговарајуће податке.

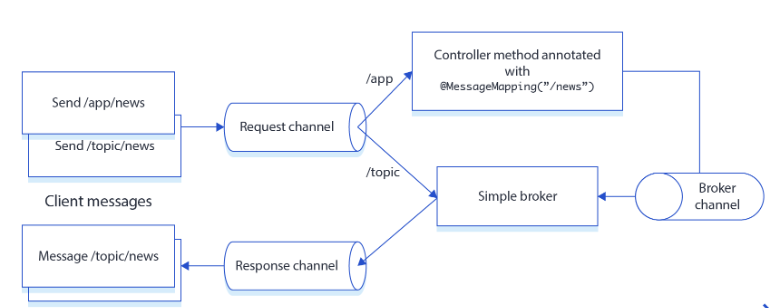
Сервер ће врло вероватно бити повезан са више клијената истовремено. Како је веб сокет предвиђен за један на један комуникацију, сервер мора отворити више сокета за комуникацију.



На слици изнад је приказан део архитектуре апликације са више повезаних корисника истовермено.

## STOMP архитектура

СТОМП је текстуално оријентисан протокол намењен да буде једноставан и широко распрорсрањен. Долази са основном шемом за поруке (садржи само заглавље на почетку поруке). СТОМП захтева да систем за поруке има одговарајућу конфигурацију и подршку како би протокол био могућ.



СТОМП у заглавље поруке убацује префиксе како би се лакше конфигурисао. Оснофни префикси су „/app“ i „/topic“. Префикс /апп служи да се одреди клијент којем се шаљу поруке. Када се клијент повезује потребно је да конфигурише одговарајући /app префикс. Префикс „/топиц“ служи да стомп брокер одреди којем клијенту се шаљу поруке. На пример може се подесити да порука са /topic/pera стигне искључиво клијенту Пера, а /topic/all стиже свима на групни чет.

Из горе приказане архитектуре моће се приметити такође да СТОМП користи канале за захтев и за одговор. Кроз канал за захтев стижу поруке, одређивањем шта брокер треба да уради њом помоћу префикса, шаље се одређеним клијентима.

На примеру апликације за дописивање СТОМП је конфикгурисан да има само један /topic префикс (за групни чет) и да се повеже са клијентима уз помоћ /app/devchat префикса.

Спринг боот и јавасцрипт у потпуности подржавају СТОМП, тако да је јако једноставно извршити конфигурације у овим софтверима.

## База података

База података је основни облик чувања података. Базе могу достићи висок ниво комплексности код великих пројеката, где се користи велики број табела и релација.

За потребе чет апликације у бази података потребно је чувати само један ентитет који описује поруку, заједо са садржајем, пошиљаоцем и примаоцем. Сврха чувања порука је искључиво да новоконектовани корисници могу да виде претходне поруке.

База података која се импементира је ЈПА, то је апликациони пшрограмски интерфејс који спринг-у омогућава перзистентност података кроз анотације ПОЈО класа.

@Entity – Предтавља да је означена класа ентитет у бази

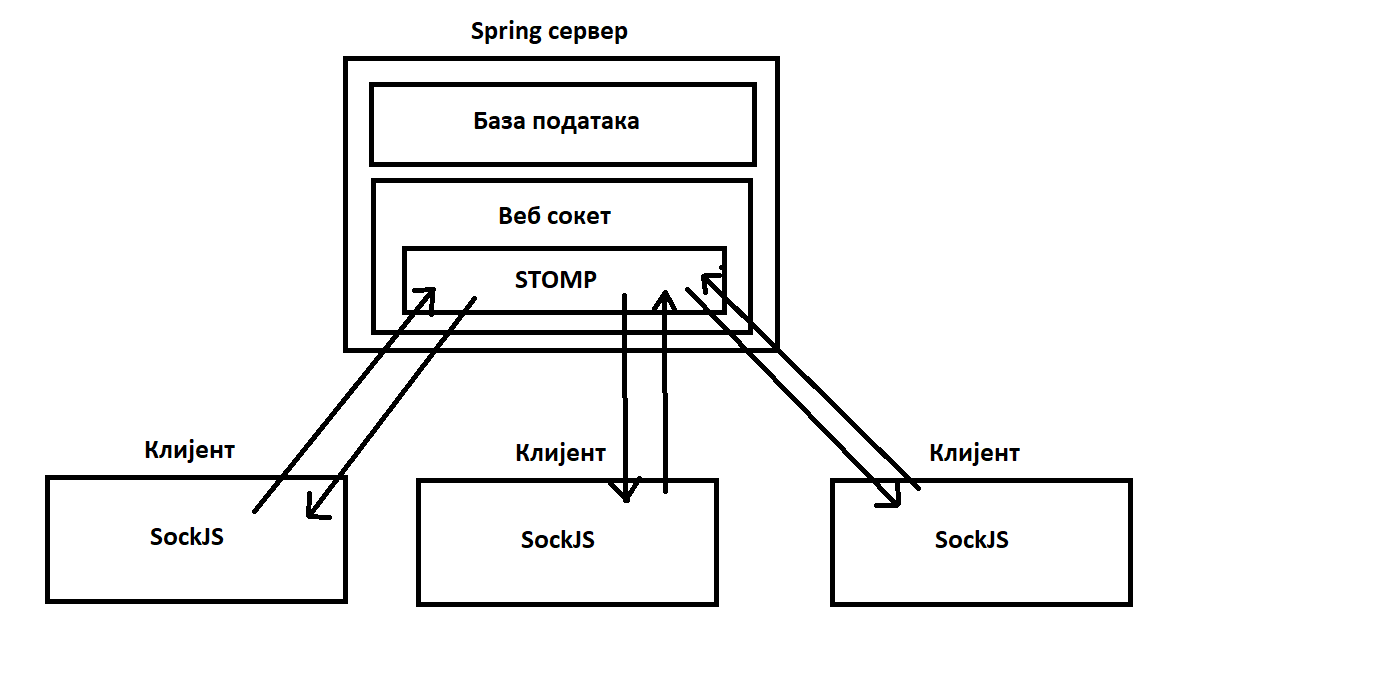
@Id – Представља поље у класи које служи као идентификатор

@Table – Означава табелу којом је мапирана класа (Уколико јесте)

@Colmun – Означава колону у табели којом је мапирано поље (Уколико јесе)

@Generated value – Означава да се вредост поља аутоматски генерише

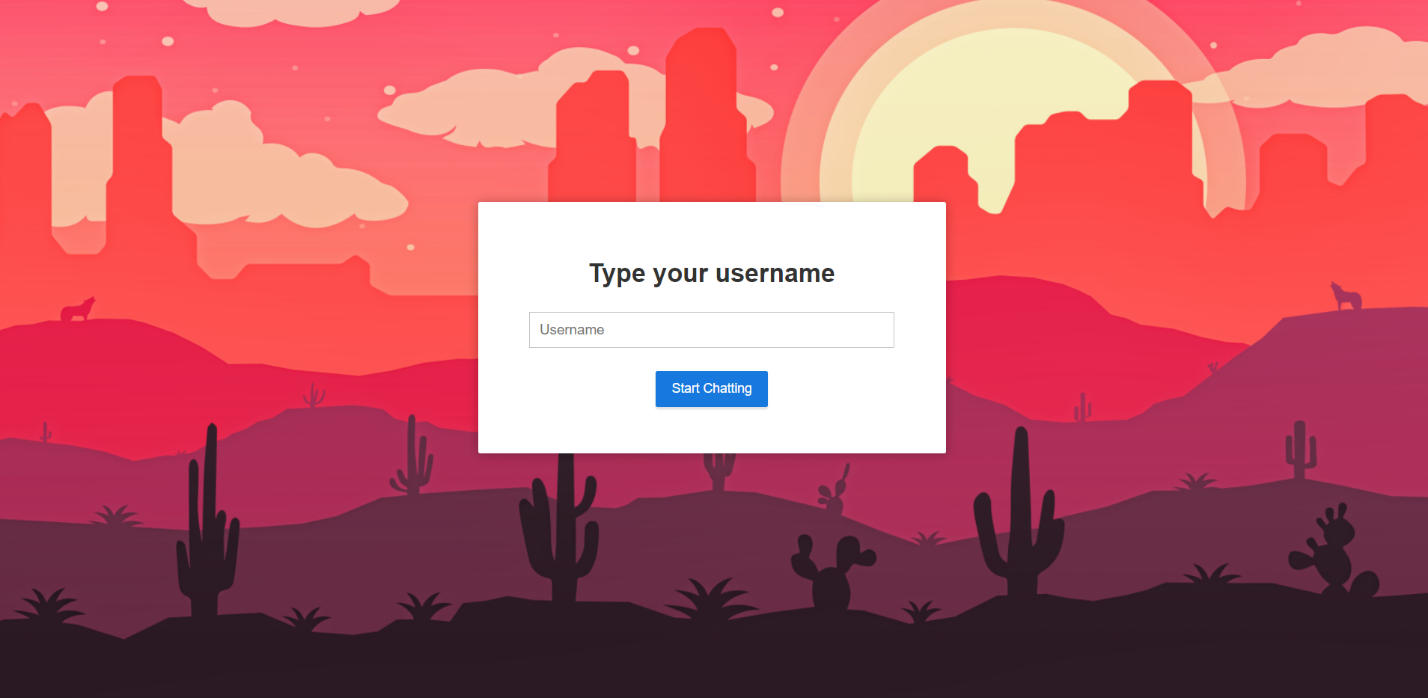
## Генерална архитектура



# Изглед апликације

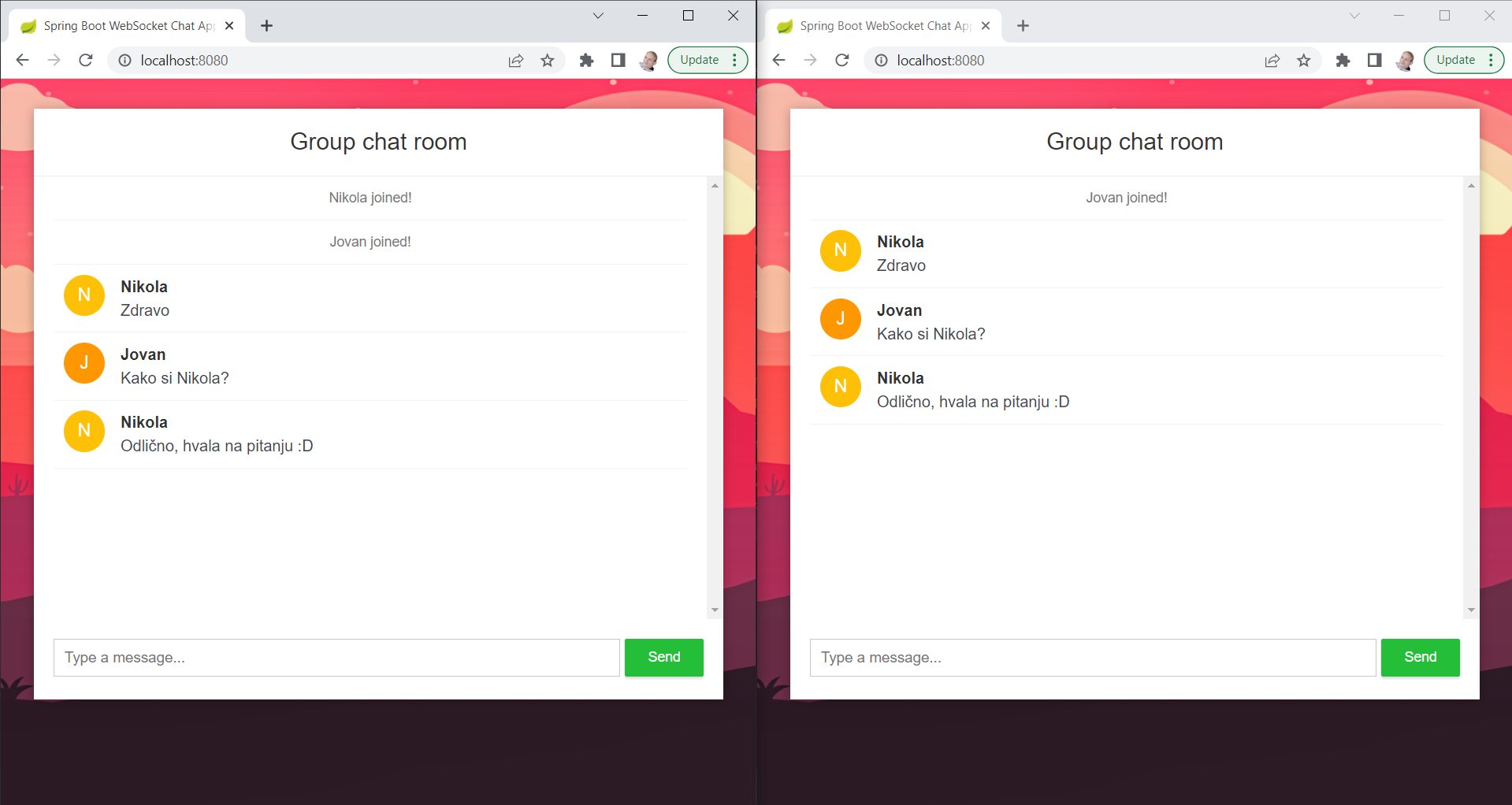
Поред естетичког фактора, изглед апликације треба да задовољи сваког корисника интуитивношћу и једноставношћу коришћења. Ова чет апликација је поприлично једноставна што се корисничког интерфејса тиче, јер се састоји из само два екрана. Један када се корисник конектује на сервер и уноси своје корисничко име и екран за сам групни чет.

Изглед за ову апликацију написан је у ХТМЛ и ЦСС језицима, док динамичност креирања компоненти покрива јаваскрипт као и функционални део клијнтске апликације.



Екран за логовање састоји се из поља за унос корисничог имена и тастера који повезује корисника на сервер и пребацује апликацију на следећи екран.

Након притиска на тастер „Start chatting“ врше се провере да ли косрисник са унетим корисничким именом већ постоји и да ли је конектован. Уколико јесте приказује се екран који описује грешку, у другом случају корисник је у могућности да се дописује.

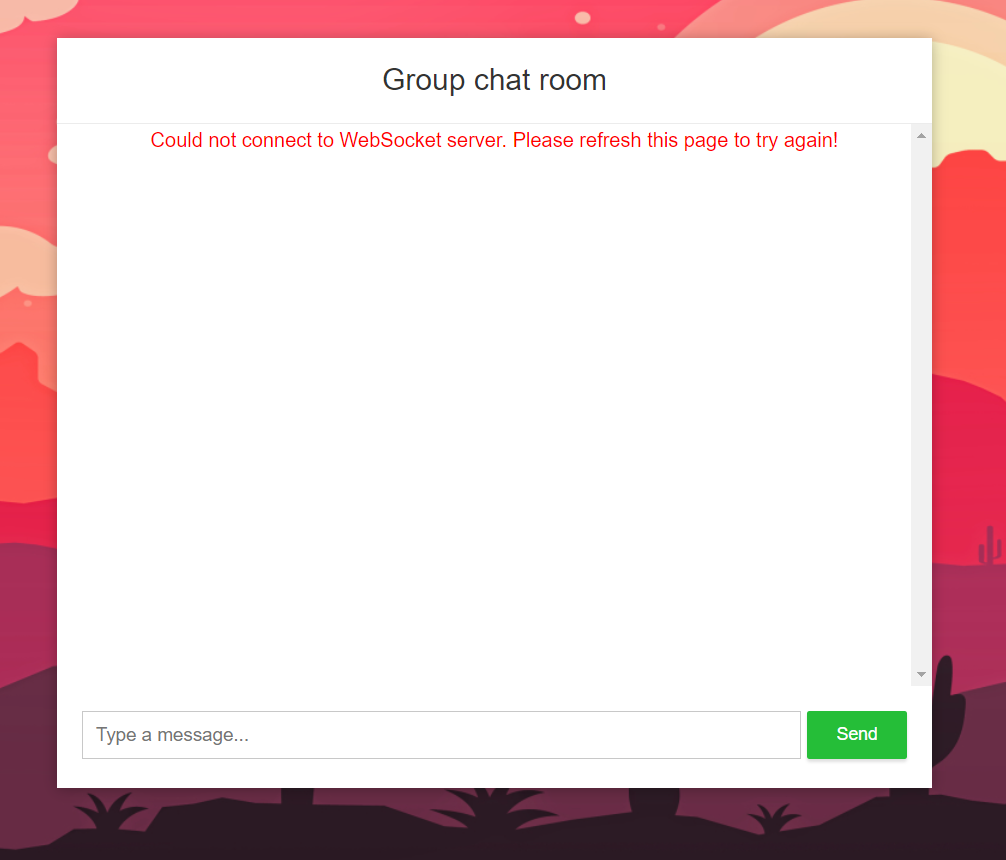


Уколико се корисник успешно улоговао пребачен је на екран за дописивање. Екран за дописивање састоји се од приказа порука, заједно са информацијама о пошиљаоцу, поља за унос поруке и тастера за слање.

Притиском на тастер за слање проверава се да ли је унета порука у одговарајуће поље, уколико јесте шаље се захтев серверу који касније одговара свим клијентима. У другом случају се не дешава ништа.

Приказ порука такође приказује информације о корисницима који су се тек прикључили групном чету. Кад год се неко успешно конектује остали корисници су о томе обавештени.

У случају да се коринсик није успешко конектовао због губитка интернет конекције, интерне серверске грешке или уколико је одабрао исто корисничко име као други корисник, приказује се екран грешке.



Екран за грешку је исти као екран за дописивање, једина је разлика што се не приказују поруке, већ информације о грешци.

# Закључак

# Закључак

# Списак слика

**No table of figures entries found.**

# Литература