UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Klemen Nagode

Skalabilni računalniški sistem za podporo komunikacije v realnem času

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01842/2012
Datum: 02.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **KLEMEN NAGODE**

Naslov: **SKALABILNI RAČUNALNIŠKI SISTEM ZA PODPORO KOMUNIKACIJE V REALNEM ČASU**

SCALABLE COMPUTER SYSTEM TO SUPPORT REAL TIME COMMUNICATION

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Proučite področje dvosmerne komunikacije med strežnikom in odjemalcem preko protokola RTMP v realnem času. Izdelajte načrt za računalniški sistem, ki omogoča realizacijo spletne klepetalnice in s tem komunikacijo med uporabniki v realnem času. Pri izdelavi načrta dajte poseben poudarek skalabilnosti in zanesljivosti sistema. Pri razvoju sistema uporabite tehnologije JavaScript, HTML, CSS in Flash. Razvijete tudi ustreznega odjemalca, ki je odporen na napake v omrežju.

Mentor:

doc. dr. Rok Rupnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Klemen Nagode,

z vpisno številko 63040111,

sem avtor diplomskega dela z naslovom:

Skalabilni računalniški sistem za podporo komunikacije v realnem času

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Roka Rupnika
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 21.06.2012 Podpis avtorja:

Zahvala

Za pomoč pri diplomskemu delu se zahvaljujem mentorju, doc. dr. Roku Rupniku, za številne koristne napotke ter strokoven pregled diplomske naloge.

Na temu mestu bi se za nasvete in usmeritve zahvalil tudi asistentu Andreju Krevlu ter laboratoriju LRK za vso potrebno infrastrukturo, ki je bila potrebna za razvoj diplomske naloge.

Posebna zahvala gre moji partnerki za moralno podporo, pomoč in vrsto spodbud med leti študija.

Staršem

Kazalo

Seznam uporabljenih kratic in simbolov

ACID - Atomicity, Consistency, Isolation, Durability

AJAX - Asynchronous JavaScript and XML

API - Application Programming Interface

AS3 - Action Script 3

CSS Cascading Style Sheets

ECMA European Computer Manufacturers Association

FTP - File Transfer Protocol

HDD - Hard Disc Drive

HTML5 - Hyper Text Markup Language, version 5

HTTP - HyperText Transfer Protocol

IRC - Internet Relay Chat

JS - Java Script

JSON - JavaScript Object Notation

NoSQL - Not Only Structured Query Language

LAMP - Linux, Apache, MySQL, PHP

LOC - Lines of code

OS - Operation System

PPB - Porazdeljena podatkovna baza

RPM - Rotation per Minute

RTMFP - Real Time Flow Protocol

RTMP - Real Time Messaging Protocol

SDD - Solid State Drive

SQL - Structured Query Language

SUPB - Sistem za upravljanje podatkovnih baz

TCP - Transmission Control Protocol

UML - Unified Modeling Language

URL - Uniform Resource Locator

UTP - User Datagram Protocol

XML - Extensible Markup Language

Povzetek

Diplomska naloga obravnava problem dvosmerne komunikacije med strežnikom in odjemalcem preko protokola RTMP ter skuša rešiti problem skalabilnosti in zanesljivosti na konkretnemu produktu. V uvodnih poglavjih je opisana zasnova odjemalca v tehnologijah JavaScript, HTML, CSS in Flash ter obravnava napak, ki nastajajo na strani odjemalca. Sledi obravnava strežniškega dela storitve v tehnologiji Red5, ki jo kasneje nadgradimo v distribuiran informacijski sistem. S konceptom skupnega pomnilnika naredimo sistem skalabilen, zanesljivost pa zagotovimo z nadzornim monitorjem, ki v primeru napak sitem postavi v konsistentno stanje. Diploma temelji na primeru implementacije napredne spletne klepetalnice, ki jo je mogoče integrirati na poljubno spletno stran.

Ključne besede:

dvosmerna komunikacija, RTMP, skalabilni sistemi, replikacija, web development

Abstract

In the thesis we discuss the problem of two-way communication between client and server over RTMP protocol using web technologies. We describe the process of building a fault-tolerant client side application with JavaScript, HTML, CSS and Flash technologies. The discussion is followed by the definition and description of RED5 server side service, which is then upgraded into a distributed information system. The problem of scalability is solved with the use of shared memory, which is implemented as a database replication cluster. The whole system is observed by a control monitor, which detects and resolve problems and thereby provides a more reliable and fault-tolerant system. The thesis provides an insight into the implementation of a web based chat room system that can be integrated into any website.

Key words:

Two-way communication, RTMP, scalable systems, replication, web development

Poglavje 1

Uvod

Uporaba spletnih tehnologij in standardov že več let hitro raste, s prihodom mobilnih tehnologij in HTML5 standardov pa danes spletne tehnologije začenjajo nadomeščati tudi vse več namiznih aplikacij. S popularnostjo spletnih aplikacij obstaja vse več storitev, ki rešujejo problem skalabilnosti in zanesljivosti, vendar pa so te storitve primerne predvsem za aplikacije z enosmerno komunikacijo "odjemalec-strežnik" in ne tudi za komunikacijo v smeri "strežnik-odjemalec".

V okviru diplomske naloge bomo razvili sistem za dvosmerno komunikacijo v realnemu času, ki bo skalabilen, zanesljiv ter bo temeljil na uporabi spletnih tehnologij. Sistem bo mogoče uporabiti za razvoj multiplayer iger, klepetalnic za spletne strani in *on-line* podpore za spletne trgovine.

Problemsko domeno bomo razbili na štiri podprobleme:

- Razvoj odjemalca v tehnologiji HTML, CSS, JavaScript in Flash
- Razvoj strežnika za enosmerno komunikacijo v tehnologijah PHP in MySQL
- Razvoj strežnika za dvosmerno komunikacijo z tehnologijo Java in Red5, ki uporablja RTMP protokol
- Zasnova skalabilnega in zanesljivega sistema za dvosmerno komunikacijo

Odjemalec bo realiziran v tehnologijah HTML in Javascript, za dvosmerno povezavo s strežnikom pa bo poskrbel Adobe Flash, ki podpira RTMP (Real-time messaging protocol) protokol. Odjemalec bo odporen na napake, saj bo v primeru težav v omrežju sposoben obnoviti lastno sejo.

Strežnik za dvosmerno komunikacijo bo implementiran v tehnologiji Java - uporabili bomo strežnik Red5, ki prav tako podpira RTMP protokol.

V nadaljevanju diplome bo zasnovan distribuiran računalniški sistem in njegova topologija, ki bo zagotovila skalabilno in zanesljivo procesiranje z nadzornim monitorjem, ki bo v primeru napak znal ponovno vzpostaviti konsistentno stanje.

Porazdeljeno medprocesno komunikacijo bomo rešili s konceptom skupnega pomnilnika, kjer je pomnilnik predstavljen kot gruča podatkovnih baz. Del distribuiranega sistema bo realiziran, predstavljeni pa bodo tudi izzivi in problemi pri realizaciji ter rešitve, ki bi naš sistem naredile še bolj skalabilen.

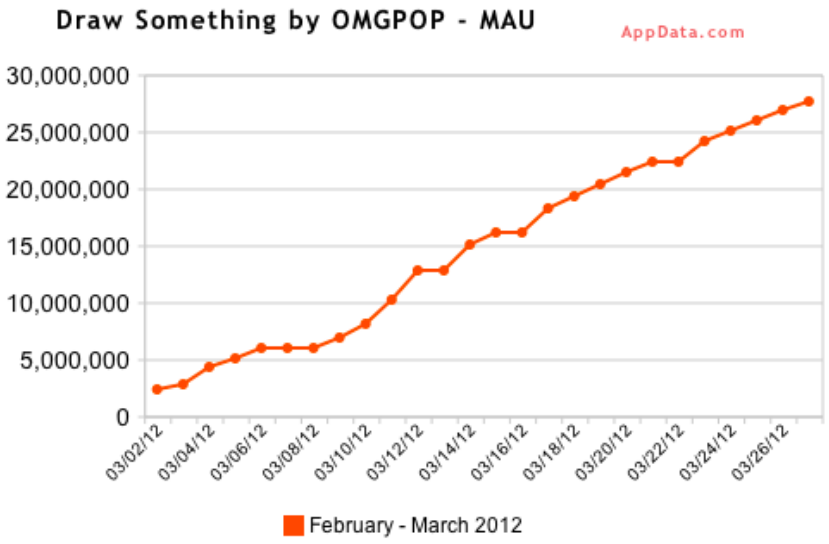
1.1 Potrebe po storitvah, ki delujejo v realnem času

Storitve, ki delujejo v realnem času, danes postajajo del vse več spletnih storitev, kot je na primer Skype, Google Talk, skupno urejanje v Google Docs in številne multiplayer igre. Da je storitev v realnem času lahko tudi ključna diferenciacija na konkurenčnem trgu, lahko vidimo na primeru aplikacije "Draw Something", ki je v tednu dni postala ena izmed najuspešnejših mobilnih aplikacij na svetu. Podjetje OMGPOP je aplikacijo po nekaj tednih delovanja prodalo podjetju Zynga za vrtoglavih 210 miljonov dolarjev, kjub ogromni vsoti pa poznavalci trdijo, da je bila aplikacija vredna milijardo dolarjev.[3]

1.1.1 Uporaba aplikacije Draw Something

- 1,2 milijona prenosov v 10 dneh od izida aplikacije
- 20 milijonov prenosov v prvih 5 tednih
- 15 milijonov dnevno aktivnih uporabnikov
- Vsak teden 6 milijonov novih uporabnikov

Vir: mobyaffiliates.com



Slika 1.1: Število uporabnikov storitve Draw Something (vir: appdata.com)

1.1.2 Finančna analiza aplikacije Draw Something

- Za oglaševanje aplikacije ni bilo porabljeno nič denarja
- $250.000 zaslužka dnevno
- Predvideni letni zaslužek: $100.000.000

Vir: mobyaffiliates.com

1.1.3 Primerjava rasti uporabnikov na sorodnih storitvah

AOL	9 let
Facebook	9 mesecev
Drawsomething	9 dni

Tabela 1.1: Čas, v katerem so storitve dosegle 1.000.000 uporabnikov

Poglavje 2

Specifikacije sistema

2.1 Opredelitev zahtev

Številne spletne strani omogočajo komunikacijo med obiskovalci preko forumov, zasebnih sporočil in komentarjev, le redke strani pa omogočajo komunikacijo med uporabniki v realnem času.

V okviru diplomske naloge bo zasnovan in realiziran sistem, ki bo lastnikom spletnih strani omogočil, da na svojo stran v roku 5 min namestijo klepetalnico, ki bo po delovanju podobna IRC-u, dodane pa bodo tudi nove možnosti, ki jih ponujajo spletne tehnologije.

Klepetalnica bo integrirana z poljubnim uporabniškim sistemom spletne strani, kar bo omogočilo, da bodo obiskovalci spletne strani v klepetalnici avtomatsko vidni pod istim nadimkom kot na strani, v katero so prijavljeni. Prenos seje uporabnika iz tretjega sistema na strežnike klepetalnice bo za končnega uporabnika neviden in povsem avtomatiziran.

Lastniki spletnih strani bodo lahko izbirali jezik spletne klepetalnice, prav tako bodo lahko izbrali tudi barvno shemo, tako da bo podoba identična izgledu spletne strani.

2.1.1 IRC

Internet Relay Chat (angleško IRC) je angleški izraz za spletni klepet, ki je eden od razširjenih načinov trenutnega (instant) skupinskega sporočanja in sporazumevanja na Internetu. Konec avgusta 1988 se je rodil IRC, program za večuporabniški pogovor, ki je bil sprva namenjen pogovorom v okviru javnega BBS-a na finski univerzi v mestu Oulu. [12]

2.2 Načrtovanje informacijskega sistema

Načrtovanje in razvoj sistema bomo razdelili na štiri segmente, kateri bodo natančneje opisani v svojih poglavjih.

- Odjemalec
- Strežnik za enosmerno komunikacijo (Označili ga bomo s kratico LAMP)
- Strežnik za dvosmerno komunikacijo (Označili ga bomo s kratico RED5)
- Zasnova skalabilnega in zanesljivega sistema za dvosmerno komunikacijo

2.2.1 Uporabniki in odgovornosti v klepetalnici

Preden opišemo podrobnosti implementacije, bomo predstavili ključne vloge uporabnikov ter ključne funkcionalnosti klepetalnice. Klepetalnica ima več tipov uporabnikov, ki so definirani v spodnjih tabelah.

Lastnik spletne strani

Opis	Lastnik spletne strani lahko na posebnem administratorskem naslovu dobi potrebno skripto, ki jo prenese na svoj spletni strežnik in tako omogoči delovanje klepetalnice. Lastnik lahko ureja barvno shemo klepetalnice in izbere jezik za navigacijo klepetalnice. Ima pravice do urejanja vseh funkcionalnosti klepetalnice in določanjem privilegijev uporabnikov.
Tip	Ključni uporabnik
Odgovornosti	Skrbi za pravilno namestitev klepetalnice in v primeru prenosa spletne strani na nov strežnik ponovno namesti klepetalnico.
Kriterij uspešnosti	Čas, v katerem lastnik namesti klepetalnico na svojo spletno stran
Vključenost	Čas, v katerem lastnik namesti klepetalnico na svojo spletno stran

Tabela 2.1: Lastnik spletne strani

Administrator sobe v klepetalnici

Opis	Administrator v klepetalnici je vezan na pogovorno sobo in ima pravico do uporabe posebnih funkcij: odstranitev uporabnika iz sobe, blokiranje vstopa uporabnikom z določenim IP naslovom in možnost ignoriranja uporabnika. Prav tako lahko administrator nastavlja privilegije ostalim uporabnikom in nastavi tip sobe
Tip	Ključni uporabnik
Odgovornosti	Skrbi za kvalitetno raven pogovora v sobi
Kriterij uspešnosti	Zadovoljstvo uporabnikov v sobi
Vključenost	Ves čas

Tabela 2.2: Administrator sobe v klepetalnici

Moderator

Opis	Moderator ima možnost potrjevanja objav uporabnikov, ko je soba v moderiranem načinu. Ima tudi pravico za blokiranje in odstranjevanje uporabnikov.
Tip	Pomožni uporabnik
Odgovornosti	Skrbi za kvalitetno raven pogovora v sobi
Kriterij uspešnosti	Zadovoljstvo uporabnikov v sobi
Vključenost	Ko je v sobi poseben gost

Tabela 2.3: Moderator

Uporabnik s pravico govora

Opis	Uporabnik lahko govori v določeni sobi, tudi če je soba moderirana
Tip	Pomožni uporabnik
Vključenost	Ves čas

Tabela 2.4: Uporabnik s pravico govora

Poseben gost

Opis	Poseben gost ima pravico do govora, tudi ko je soba v moderiranem načinu. Posebni gost je navadno slavna osebnost, ki jo uredniki spletne strani povabijo na klepet z obiskovalci spletne strani. Sporočila posebnega gosta se pobarvajo v posebno barvo. Ostali uporabniki ne morejo začeti privatnega klepeta z njim.
Tip	Pomožni uporabnik
Odgovornosti	Skrbi za kvalitetne odgovore na vprašanja obiskovalcev
Kriterij uspešnosti	Zadovoljstvo uporabnikov v sobi

Tabela 2.5: Poseben gost

Prijavljen uporabnik

Opis	Uporabnik je oseba, ki se je v storitev prijavila, kar pomeni, da je bila predhodno registrirana in verificirana preko elektronskega naslova. Pred vstopom v klepetalnico se mora vedno prijaviti.
Tip	Ključni uporabnik
Vključenost	Ves čas

Tabela 2.6: Prijavljen uporabnik

Neprijavljen uporabnik

Opis	Neprijavljen uporabnik je uporabnik, ki se v storitev vključi tako, da vnese samo vzdevek ali pa mu je ta avtomatsko dodan in je v klepetalnici viden kot gost
Tip	Pomožni uporabnik
Vključenost	Ves čas

Tabela 2.7: Neprijavljen uporabnik

2.2.2 Pregled izdelka in opis funkcionalnosti

- Klepet v sobi, v katero se lahko vključi vsak
- Možnost različnih privilegijev uporabnikov v sobah
- Nastavljanje statusa dosegljivosti uporabnika (dosegljiv, odsoten, zaseden)
- Možnost moderiranja klepeta v sobi
  - Uporabniki, ki ne spoštujejo pravil, so lahko odstranjeni iz klepetalnice
  - Blokiranje IP naslova uporabnika
  - Ignoriranje uporabnikov z nastavljanjem pištkotka v brskalniku
- Nastavljanje tipa sobe
  - Prosti pogovor, v katerem lahko sodelujejo vsi
  - Soba, v kateri lahko govorijo samo registrirani uporabniki
  - Moderirana soba, v kateri lahko govorijo samo uporabniki s pravico govora in posebni gosti. Moderator potrjuje objavo sporočil uporabnikov
  - Lobby room: soba, v kateri je mogoče oddati in sprejeti izziv za igranje multiplayer iger
- Privaten klepet
  - Možnost klepeta
  - Možnost igranja multiplayer iger
- Nastavitve zasebnosti uporabnikov (vsak ali nihče lahko začne privaten pogovor z uporabnikom)

Poglavje 3

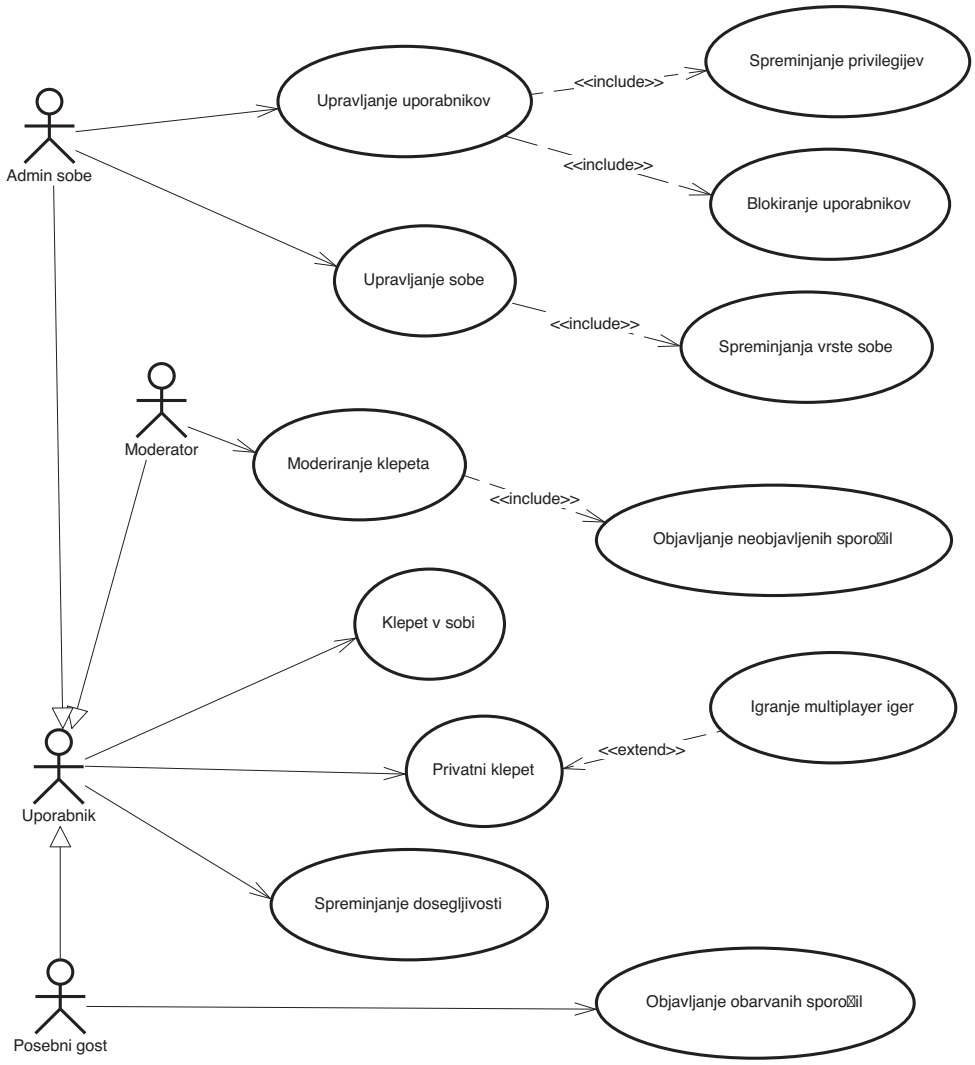
Razvoj odjemalca za klepetalnico

V poglavju bomo opisali potek razvoja odjemalca za klepetalnico, ki bo deloval kot spletna storitev v okviru spletnega brskalnika. Opisani bodo ključni primeri uporabe z zaslonskimi maskami in pa tehnična rešitev problema. Za načrtovanje smo uporabili orodje Power Designer, razvojno okolje za Javascript, HTML in CSS pa je Aptana Studio 3.

V začetku poglavja bomo predstavili ključne zahteve, kompleksnejše operacije pa bomo opisali z UML diagrami. Sledi opis uporabljenih tehnologij in knjižnic, ki so nam olajšale delo, nato pa bomo predstavili težave odjemalca, ki nastajajo zaradi nestabilnega okolja (problem prekinitve povezave ...) in jih rešili. Dotaknili se bomo tudi možnih tveganj pri razvoju sorodne storitve za mobilne naprave iOS in Android, ki sta vodilni platformi za razvoj mobilnih aplikacij.

Odjemalec bo imel številne funkcionalnosti, ki bodo poleg uporabe v klepetalnici koristile tudi drugim storitvah, zato jih bomo na koncu poglavja predstavili in implementirali primer take storitve.

3.1 Primeri uporabe odjemalca



Slika 3.1: Slika prikazuje primere uporabe, ki smo jih implementirali pri odjemalcu storitve

3.1.1 Opisi primerov uporabe

Klepet v sobi

Klepet v sobi je omogočen vsem uporabnikom, razen če je admin to funkcijo onemogočil. Sporočila, ki so poslana v sobo, vidijo vsi uporabniki, ki so trenutno v sobi. Na desni strani klepetalnice je mogoče videti tudi seznam vseh uporabnikov, ki so v določeni sobi. Ob vsakem uporabniku je tudi indikator dosegljivosti (dosegljiv, zaseden, odsoten). Klik na uporabnika ponudi različne možnosti operacij nad uporabnikom:

- Začni privatni klepet
- Odstrani uporabnika iz klepeta
- Blokiraj uporabnikov IP

Slika 3.2: Zaslonska maska klepetalnice z odprto sobo, v kateri se lahko pogovarja več uporabnikov

Upravljanje uporabnikov

Ko uporabnik vstopi v sobo, ki je prazna, dobi pravico Admin-a, kar mu omogoči upravljanje uporabnikov in upravljanje sobe. Nad vsemi uporabniki lahko izvede funkcijo blokiranja, uporabnike lahko tudi samo začasno odstrani iz klepeta. Vsem uporabnikom lahko nastavlja vse tipe privilegijev: admin, moderator, poseben gost, uporabnik s pravico govora in navaden uporabnik.

Upravljanje sobe

Admin lahko ureja tip sobe, kjer ima možnost izbirati med sobo, v kateri lahko govori vsak, sobo, v kateri lahko govorijo samo prijavljeni uporabniki, in moderirano sobo, kjer morajo biti vsa sporočila uporabnikov pred objavo potrjena iz strani moderatorja.

Moderiranje klepeta

Funkcija moderiranja pripada moderatorju, ki v primeru moderirane sobe objavlja ali zavrača sporočila uporabnikov. Tako skrbi, da ima pogovor v sobi smisel. Funkcija je posebej primerna, ko je v sobi posebni gost. Moderator lahko ob koncu klepeta celoten klepet izvozi v RSS format, ki je uporaben za objavo povzetka klepeta na novičarskih spletnih straneh.

Privatni klepet

Privatni klepet je pogovor med natanko dvema uporabnikoma, ki ni javno viden. Uporabnika imata poleg možnosti klepeta tudi možnost, da začneta z igranjem multiplayer iger. Z enim uporabnikom je v danem trenutku možno igrati natanko eno igro.

Slika 3.3: Zaslonska maska privatnega klepeta

Spreminjanje dosegljivosti

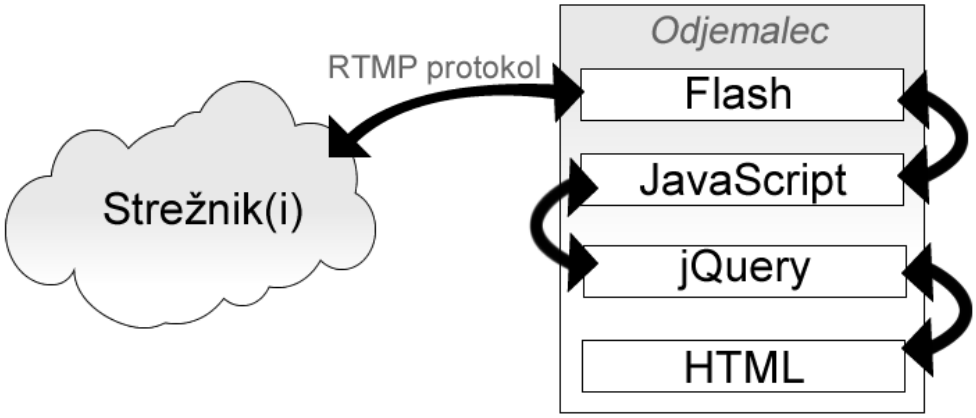
Uporabniki imajo tudi indikator dosegljivosti (dosegljiv, odsoten, zaseden), iz katerega je razvidno, ali je uporabnik trenutno pripravljen na pogovor. Indikator "odsoten" se avtomatsko vklopi, če uporabnik miške po zaslonu ni premaknil dlje časa.

Objavljanje obarvanih sporočil

Ko je v klepetalnici aktiven poseben gost, želimo, da njegova sporočila izstopajo od sporočil ostalih uporabnikov.

3.2 Tehnična zasnova odjemalca

Odjemalec je sprogramiran v tehnologijah HTML, Javascript in CSS, za povezavo s strežnikom pa skrbi Flash, ki podpira RTMP protokol.

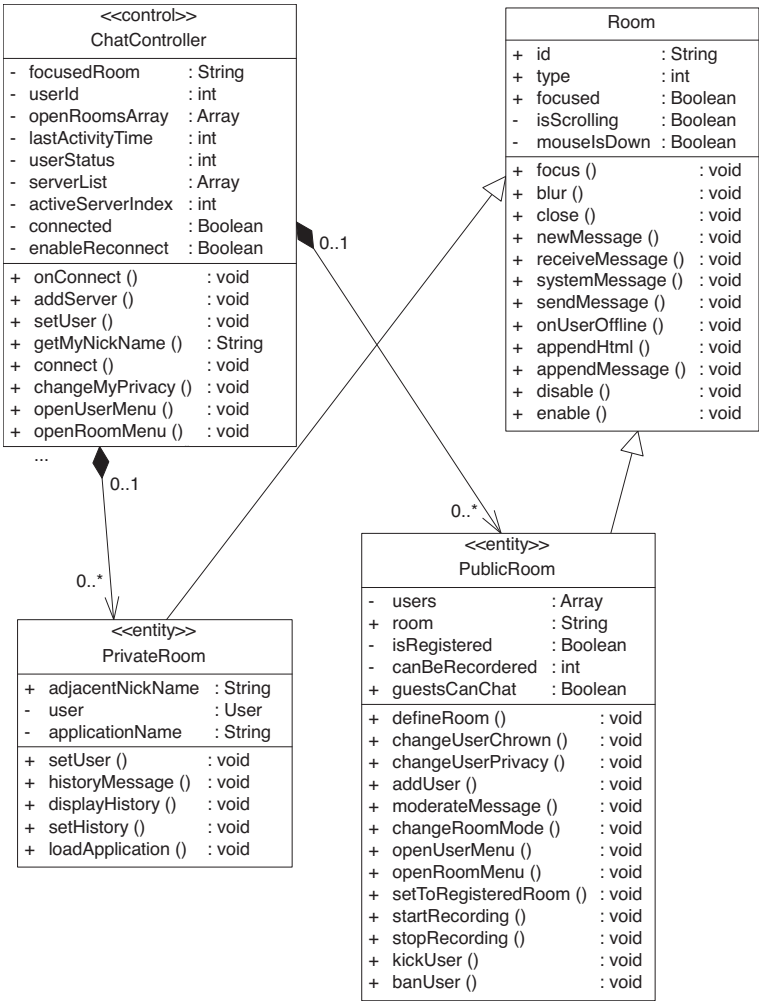


Slika 3.4: Slika prikazuje komunikacijo odjemalca s strežnikom. Brskalnik preko "socket"povezave pošilja sporočila strežniku v formatu JSON, le-ta pa mu na isti način pošilja informacijo nazaj. Flash in JavaScript obojestransko komunicirata in si ves čas izmenjujeta informacije. Javascript preko jQuery-ija spreminja HTML kodo dokumenta in tako skrbi za prikaz uporabniškega vmesnika.

Ključne funkcionalnosti odjemalca so sprogramirane v skriptnem jeziku JavaScript, ki omogoča tudi objektno programiranje in enkapsulacijo. Funkcije klepetalnice so razdeljene na več smiselnih razredov, ki so realizirani glede na primere uporabe. Načrtovanje in objektni pristop je bilo nujno potrebno, saj se je izkazalo, da je samo ključni del JavaScript storitve zahteval več kot 3000 LOC (Line Of Code).

3.2.1 Razredni diagram odjemalca z opisi

Pripravili smo razredni diagram za JavaScript razrede, da celoten problem lažje razložimo. Vsi razredi se nahajajo v mapi: /js/chat/.



Slika 3.5: Poenostavljen razredni diagram prikazuje ključne atribute in ključne operacije nad razredom

ChatController

Razred skrbi za delovanje celotne klepetalnice in je vstopna in izhodna točka za pošiljanje vseh sporočil. Implementirane ima mehanizme za povezavo s strežnikom, ob prekinitvah povezave pa je sposoben sam ponovno vzpostaviti povezavo. Razred ima celoten nadzor nad vsemi entitetami in zna odpirati in zapirati okna za klepet z uporabniki. V razredu so tudi implementacije funkcionalnosti, ki niso vezani izključno na sobo ali privaten klepet (npr: spreminjanje dosegljivosti in zasebnosti uporabnika)

PrivateRoom

Razred skrbi za vizualizacijo privatnih sporočil med uporabniki. Njegova posebna lastnost je, da so v njem implementirane metode, ki znajo v primeru osvežitve strani obnoviti zgodovino klepeta. V privatnem klepetu je mogoče igrati tudi multiplayer igre, kar zagotavlja metoda "loadApplication(), ki naloži Flash igro in poskrbi, da se igra nemoteno naloži tudi na nasprotnikovi strani. Ko je igra pripravljena na obeh straneh, se igranje iger lahko prične.

PublicRoom

Implementacija javne sobe vključuje vse funkcije za upravljanje sobe in uporabnikov v njej. Definira tudi funkcije za moderiranje in potrjevanje neobjavljenih sporočil, ki so zanimiva v primeru, ko je soba moderirana. Vključuje tudi funkcije za začetek in konec snemanja klepeta, kar nam vrne RSS s povzetkom klepeta.

Ostali razredi

Poleg zgoraj opisanih razredov je pomemben razred za razvoj programske opreme opravljal še SystemRoom, ki sprejema vsa sistemska sporočila, ki so potrebna za nemoteno delovanje klepetalnice. Soba je privzeto skrita, vendar jo lahko prikažemo in preko nje vidimo celoten vhodni in izhodni promet klepetalnice, kar precej olajša razhroščevanje aplikacije.

Implementiran je tudi razred GameRoom, ki omogoča, da uporabniki v njem oddajo zahtevo za igranje iger. Zahteve za igre so vidne vsem uporabnikom, ko pa na zahtevo klikne nek drug uporabnik, se odpre privatni klepet z odprto prednastavljeno igro.

3.3 Uporabljene tehnologije

3.3.1 HTML

Hyper Text Markup Language (slovensko jezik za označevanje nadbesedila, kratica HTML) je označevalni jezik za izdelavo spletnih strani. Predstavlja osnovo spletnega dokumenta. S pomočjo HTML razen prikaza dokumenta v brskalniku hkrati določimo tudi strukturo in semantični pomen delov dokumenta. [11]

3.3.2 JavaScript

JavaScript je objektni skriptni programski jezik, ki ga je razvil Netscape, da bi spletnim programerjem pomagal pri ustvarjanju interaktivnih spletnih strani. Jezik je bil razvit neodvisno od Jave, vendar si z njo deli številne lastnosti in strukture. JavaScript lahko sodeluje s HTML-kodo in s tem poživi stran z dinamičnim izvajanjem. JavaScript podpirajo velika programska podjetja in kot odprt jezik ga lahko uporablja vsakdo, ne da bi pri tem potreboval licenco. [13]

Objektni pristopi v JavaScript-u

JavaScript že v jedru deluje kot objektni jezik. Seznami so objekti, funkcije so objekti in tudi objekti so objekti. Objekt je definiran kot par ime-vrednost. Imena so predstavljena kot nizi, vrednosti pa so lahko nizi, logične vrednosti, števila ali objekti. Objekti so definirani kot *hash* tabele, tako da so vrednosti hitro dostopne. [4]
Objekti se kreirajo s konstruktorji, ki so funkcije, ki inicializirajo objekt.
JavaScript prav tako omogoča privatne in javne instančne spremenljivke, enako velja tudi za metode.

Javne spremenljivke in metode

```
1 function MojRazred(vrednost) {
2     this.vrednost = vrednost;
3 }
4
5 var obj = new MojRazred("abc");
6 alert(obj.vrednost); // vrne "abc"
```

Listing 3.1: Tehnika za definiranje javnih instančnih spremenljivk

```
1 MojRazred.prototype.javnaFunkcija = function (string) {
2     return string+"def";
3 }
4
5 alert(obj.javnaFunkcija("abc"); // vrne "abcdef"
```

Listing 3.2: Tehnika za definiranje javnih metod preko prototype načina

Privatne spremenljivke in metode

Privatne spremenljivke lahko dodajamo v konstruktorju, z uporabo ukaza *var*.

```
function MojRazred(vrednost) {
    var self = this;
    var privatnaVrednost = vrednost;

    function privatnaMetoda() {
        return "abcpvt";
    }
}
```

Listing 3.3: Tehnika za definiranje privatnih metod in vrednosti

Konvencija priporoča, da v konstruktorju definiramo tudi spremenljivko *self*, ki kaže na kreirani objekt. Posebna spremenljivka omogoča, da je objekt dostopen privatnim metodam - s tem se izognemo napaki v ECMA Script Language specifikaciji, ki povzroči, da uporaba *this* ne deluje pravilno v notranjih funkcijah v razredu.

Privatne metode ne morejo biti klicane iz javnih metod, za kar obstajajo še *privileged* metode.

Privilegirane metode

Privilegirane metode imajo dostop do privatnih spremenljivk in metod ter so dosegljive tudi drugim javnim metodam.

```
function MojRazred(vrednost) {
     this.membername = function (...) {...};
}
```

Listing 3.4: Tehnika za definiranje priviligiranih metod

jQuery

Razvoj v JavaScriptu je bistveno olajšala knjižnica jQuery, ki razvijalcu omogoči, da se ne ukvarja z nestandardnimi oblikami implementacij JavaScript jezika in poskrbi za to, da program deluje na vseh popularnih brskalnikih, poleg tega pa zahteva bistveno manj napisane kode. jQuery poenostavi manipulacijo s HTML, bistveno olajša upravljanje JavaScript dogodkov, animacij in razvoj AJAX aplikacij.

Primer: Skrivanje HTML elementa z id atributom:

```
<div id="moj_id">Vsebina</div>
```

Listing 3.5: HTML koda

```
1 $("#moj_id").hide();
```

Listing 3.6: JavaScript koda za skrivanje elementa

SWFObject

Vključevanje SWF (flash) datotek zaradi različnih interpretacij W3C standardov zahteva precej različnih pristopov za vdelavo. Knjižnica SWFObject nam omogoča, da lahko Flash objekte vključimo na spletno stran ne glede na brskalnik, kar smo s pridom uporabili na vseh odsekih, kjer uporabljamo Flash.

3.3.3 CSS

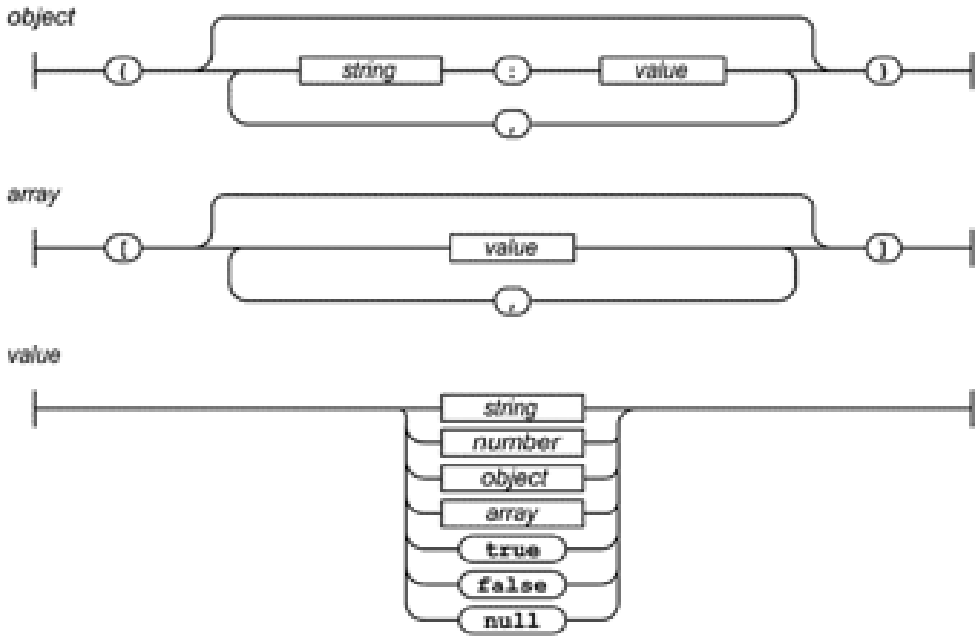
Cascading Style Sheets (sl. kaskadne stilske podloge) poznane pod kratico CSS so podloge, predstavljene v obliki preprostega slogovnega jezika, ki skrbi za prezentacijo spletnih strani. Z njimi definiramo stil HTML oz. XHTML elementov v smislu pravil, kako se naj ti prikažejo na strani. Določamo lahko barve, velikosti, odmike, poravnave, obrobe, pozicije in vrsto drugih atributov, prav tako pa lahko nadziramo aktivnosti, ki jih uporabnik nad elementi strani izvaja (npr. prekritje povezave z miško). Podloge so bile razvite z namenom konsistentnega načina podajanja informacij o stilu spletnim dokumentom. [10]

3.3.4 Adobe Flash

Adobe Flash je multimedijska platforma, ki je namenjena za ustvarjanje animacij, videoposnetkov in interaktivnih spletnih strani. Vključuje tudi svoj skriptni jezik, ki med drugim že več let omogoča tudi ustvarjanje "socket"povezav ter podporo RTMP protokolu, ki je namenjen izdelavi storitev v realnem času.

3.3.5 JSON

JSON (JavaScript Object Notation) je podatkovna struktura, ki za opis strukture potrebuje zelo malo redundantne informacije. Opisan je na način, ki je zelo lahko berljiv in hitro razumljiv za človeka. Iz skriptnega jezika JavaScript je bila ideja o formatu hitro prenešena tudi na druge platforme, tako da danes lahko najdemo knjižnice za procesiranje JSON formata praktično za vsak popularen programski jezik. S tem formatom lahko enostavno opišemo širše uporabljene podatkovne strukture kot so objekti, polja (arrays), nize, števila ...



Slika 3.6: Struktura opisnega jezika JSON (vir: Wikipedia)

```
{
  "function":"privateUserMessage",
  "params":{
    "user":{
          "userId":1,
          "userName":"klemen"
    },
          "message":"Hi there"
     }
}
```

Listing 3.7: Primer uporabe JSON sintakse, ki prikazuje strukturo privatnega sporočila

3.4 Delovanje odjemalca v nestabilnih pogojih

Končni cilj aplikacije je tudi stabilno delovanje v nestabilnih pogojih, do katerih lahko pride zaradi napak pri odjemalcu ali strežniku za komunikacijo v realnem času.

3.4.1 Težave pri vzpostavitvi povezave

Odjemalec ima lokalno vedno spisek delujočih strežnikov, na katere se takoj po inicializaciji poskuša povezati. Če se povezava ne vzpostavi po več sekundah, lahko z veliko mero gotovosti predvidevamo, da je na strani strežnika prišlo do težave, zato iz spiska delujočih strežnikov vzamemo naslednjega in se skušamo povezati z njim . Ko se neuspešno skušamo povezati na zadnjega izmed strežnikov, je naš spisek očitno zastarel, zato iz posebnega URL naslova zahtevamo nov seznam strežnikov in nadaljujemo z poskušanjem vzpostavljanja povezave.

Do težav pri vzpostavitvi povezave lahko pride, tudi če nam omrežje, preko katerega se povezujemo na strežnik, ne dovoljuje protokola RTMP ali vrat 8080, ki so privzeta za RTMP protokol. Protokol RTMP omogoča tudi tuneliranje protokola, kar pomeni, da se celoten promet izvede preko vrat 80 in HTTP protokola. Tuneliran protokol je sicer počasnejši in ima večjo latenco, vendar deluje. Ker je tudi ta težava precej pogosta, se ob poskusu vzpostavitve povezave odjemalec najprej skuša povezati preko vrat 8080, nato pa še preko vrat 80, kar zagotavlja delovanje tudi na omrežjih z omejenim dostopom.

3.4.2 Težave zaradi prekinitve povezave

Ključna napaka pri odjemalcu je prekinitev povezave s strežnikom, kar protokol RTMP zazna in nas o tem obvesti. Omenjen klic uporabimo za poskus ponovne vzpostavitve s strežnikom. Ker je bila seja nekaj časa prekinjena, obstaja možnost, da je strežnik do uporabnika skušal poslati sporočilo, ki pa seveda ne prispe na cilj. V kolikor gre za prekinitev in ponovno vzpostavitev povezave v smotrnem času (do 15s), strežnik uporabniku pošlje tudi zgodovino vseh odprtih sob, kar odjemalec tudi pravilno prikaže.

3.4.3 Težave zaradi človeške napake

V kolikor uporabnik nenamenoma osveži spletno stran, upoštevamo podoben postopek kot pri prekinitvi povezave, tako da se seja uporabnika v celoti povrne na zadnje stanje. Sejo se ponovno vzpostavi preko PHP sejnega piškotka, ki ga uporabimo tudi za verifikacijo seje na strežniku za komunikacijo v realnem času.

3.5 Tveganja in ublažitve tveganja pri razvoju odjemalca

Mobilne naprave so vse do aprila 2011 skušale v svoje platforme vgrajevati podporo za Flash. Stvari so se začele drastično spreminjati, ko je Steve Jobs napovedal, da v bodočih verzijah iOS-a Apple ne bo več podpiral Flash-a, ker ni odprtokoden, ima varnostne luknje, nima dobre podpore za vmesnike na dotik in ima preveliko porabo energije. [2]

Ker je podpora HTML5 standarda v brskalnikih vse večja, smo skušali uporabo Flasha čim bolj zminimizirati, tako da bo celotno funkcionalnost Flasha mogoče hitro in enostavno zamenjati s HTML5 standardi ali z drugim vmesnikom, ki podpira ta protokol.

V kolikor bi izbrane funkcionalnosti želeli implementirati tudi na mobilnih napravah, bi povezavo s strežnikom implemenitrali preko "native"knjižnic, ki že obstajajo za vodilne mobilne platforme.

3.5.1 Apple iOS

RTMP protokol je že mogoče uporabiti na najpopularnejših mobilnih napravah. Knjižnice za RTMP protokol razvija Adobe, možno pa je uporabiti tudi druge rešitve, npr:

The Midnight Coders - Web ORB for mobile
(http://www.themidnightcoders.com)

3.5.2 Google Android

Kljub temu, da je Google še nekaj časa nazaj bil prepričan, da je odprtost in podpora Flashu prava smer, vse kaže, da Flash v naslednjih verzijah Android-a ne bo več podprt. Zato že obstajajo knjižnice, ki omogočajo uporabo RTMP protokola brez uporabe Flash-a.

AFTEK RTMP library (http://www.aftek.com/)

3.5.3 Razvoj RED5

Pričakuje se, da se bo na strežnik RED5 v kratkem možno povezati tudi preko HTML5 WebSocket povezav, katere moderni spletni brskalniki že podpirajo, kar bo omogočilo še lažjo implementacijo povezave na strežnik.

3.5.4 Možne implementacije odjemalca

V poglavju smo opisali implementacijo klepetalnice, vendar lahko s prilagoditvami odjemalca rešimo še marsikateri drugi problem, ki zahteva dvosmerno komunikacijo. V okviru diplomske naloge je razvita knjižnica za okolje AS3, ki omogoča enostavno in hitro implementacijo multiplayer iger. Našo storitev bi lahko brez težav uporabili tudi za storitev on-line podpore na spletnih trgovinah - kodo odjemalca bi bilo treba le malenkost prilagoditi.

Izdelava Flash iger

Razvijalec se v okviru razvoja Flash igre ne zaveda kompleksnosti sistema v ozadju, saj mu knjižnica ponuja enostavne možnosti za povezavo z naključnim drugim uporabnikom, ki želi igrati v istem trenutku. Knjižnica omogoča pošiljanje in sprejemanje sporočil, v igro pa je povsem enostavno vključiti tudi klepet z uporabnikom, s katerim trenutno igramo. Čas razvoja igre je zaradi ključne poenostavitve minimalen, tako da je lahko povsem funkcionalna igra zaključena v nekaj dneh.

Knjižnica implementira naslednje funkcije:

- init() // Poveže s strežnikom in avtomatsko najde soigralca za igro. Funkcija avtomatsko določi tudi, čigava je prva poteza
- send() // Omogoča pošiljanje ukazov nasprotniku
- addReceiveListener() // Omogoča, da dodamo funkcijo, ki bo klicana ob prejemu ukaza/sporočila
- reset() // ponovna inicializacija igre

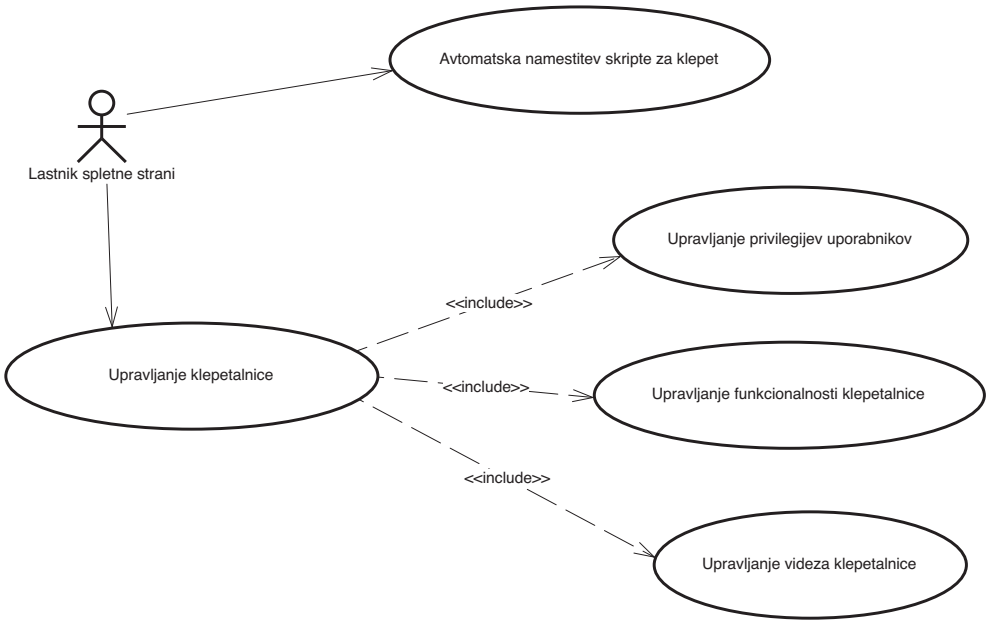
Slika 3.7: Primer uporabe naše storitve za reševanje drugih problemov, ki zahtevajo dvosmerno komunikacijo

Poglavje 4

Strežnik za enosmerno komunikacijo

Pomemben del naše storitve je Apache strežnik, ki procesira PHP in MySQL, ter je odgovoren za to, da se odjemalcu vrne program za klepetalnico in ustvari sejo uporabnika. Za hitro označevanje strežnika za enosmerno komunikacijo bomo uporabljali kratico LAMP (Linux, Apache, MySQL, PHP). Za ustvarjanje unikatne seje uporabimo kar PHP vgrajeno funkcijo session_start(), ki za uporabnika zgenerira unikaten niz znakov, ki unikatno definira sejo, odjemalec pa si ta niz shrani v piškotek v brskalniku. Piškotek se posreduje tudi strežniku Red5, ki od LAMP strežnika zahteva ključne podatke o uporabniku s posredovano sejo, s čimer dosežemo, da se seja uporabnika prenaša tudi med sicer zaprtimi računalniškimi sistemi, kar je podrobneje opisano v sledečem poglavju.

4.1 Ključni primeri uporabe



Slika 4.1: Ključni primeri uporabe, ki se vršijo na strežniku za enosmerno komunikacijo v tehnologiji LAMP

4.1.1 Opisi primerov uporabe

Avtomatska namestitev skripte za klepet

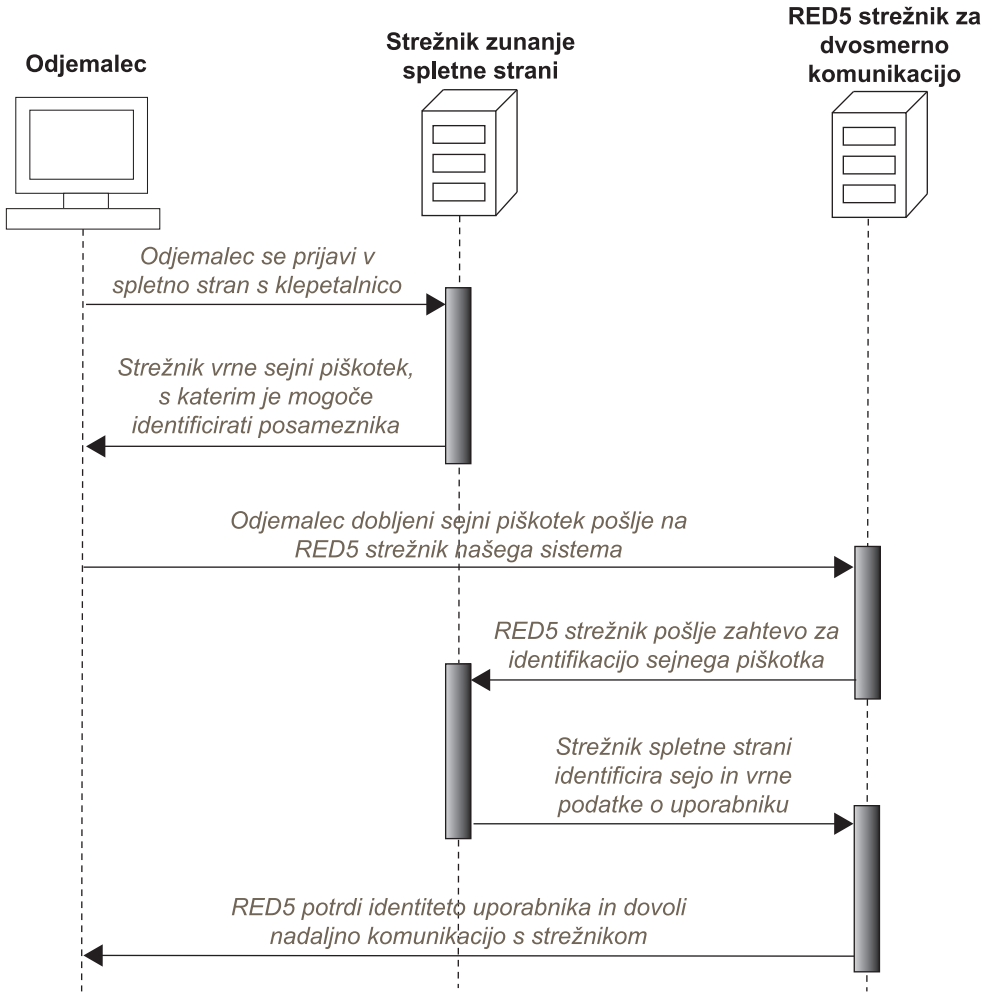
Strežnik LAMP skrbi za to, da lahko lasniki spletne strani enostavno namestijo klepetalnico na spletno stran v roku nekaj minut. Skripta shrani vse parametri, ki so potrebni za to, da je komunikacija med več zaprtimi sistemi sploh možna.

Upravljanje klepetalnice

Strežnik skrbi tudi za upravljanje parametrov, kar lahko počnejo lastniki spletnih strani po predhodni avtentikaciji. Lastniki spletne strani lahko upravljajo administratorje po sobah, nastavljajo lahko videz klepetalnice in druge funkcionalnosti.

4.2 Sinhronizacija seje s strežnikom za dvosmerno komunikacijo

Klepetalnico je mogoče namestiti na katerokoli spletno stran, kar pomeni, da je potrebno sejo uporabnika prenesti tudi v naš sistem, kar prikazuje spodnji diagram.



Slika 4.2: Enostaven način za prenos seje omogoča hitro inštalacijo klepetalnice na katerokoli eksterno spletno stran, ne glede na to, kakšno tehnologijo uporablja. Vsa komunikacija poteka v notaciji JSON.

4.3 Namestitev klepetalnice na spletno stran

Opisana arhitektura omogoča, da se klepetalnico vključi na sleherno spletno stran. Če lastnik spletne strani želi namestiti klepetalnico, mora upoštevati naslednje korake:

1.) Za prikaz klepetalnice na spletni strani je potrebno uporabiti »JavaScript« kodo, ki bo klepetalnico prikazal v obliki »iframe« okna.

```
<!--  chat: start -->
<div id="chat_placeholder"></div> <script type="text/javascript
    ">
<!--
_chat_width="100%";
_chat_height="400px";
_chat_integrated=true;
_chat_session_cookie_name="PHPSESSID";
document.write("<script src=\"http://chat.ltd/js/chat/
    integrated/Chat.js\"

type=\"text/javascript\"></sc"+"ript>"); -->
</script>
<!--  chat: ends here -->
```

Listing 4.1: HTML koda za prikaz klepetalnice

Skripti je potrebno podati ime piškotka, v katerem se nahaja seja uporabnika (v PHP-ju se piškotek privzeto imenuje PHPSESSID). Preko tega piškotka bo lahko naš strežnik identificiral uporabnika in pridobil njegove podatke (vzdevek in avatar).

2.) Ko uporabnik vstopi v klepetalnico, naš strežnik zahteva skripto za prenos seje, katero postavi lastnik spletne strani. Red5 strežnik v tej zahtevi vedno pošlje tudi piškotek s sejo uporabnika, tako da je zahteva ista kot zahteva uporabnika, realizacija pa je zato zelo enostavna, saj se lahko uporabljajo vgrajeni ukazi programskega jezika, na katerem je spletna stran postavljena.

Psevdo koda skripte za prenos seje:

```
if(GET.action == "check_if_logged"){
  session_start();
  if(SESSION.user_id){ # check if user is logged in
    print '{"error":0}';
  }else{
```

```
6       print '{"error":1, "msg":"user is not logged in"}';
7     }
8   }else if(GET.action == ''identification''){
9     session_start();
10    user = getUserByIdFromDb(SESSION.user_id);  # read user from
          database
11    if(user){
12      print '{
13        "error":0,
14        "nickname":"'+user.userName+ '",
15        "displayName":"'+user.userName+'",
16        "picture":"'+user.picture+'"
17        }';
18    }else {
19      print '{"error":1, "msg":"no user in database"}' ;
20    }
21  }
```

Listing 4.2: HTML koda za prikaz klepetalnice

Delovanje skripte je enostavno preveriti, tako da se v brskalniku prijavimo na portal in obiščemo sledeče URL-je:

http://www.namisljeniportal.si/pot-do-skripte.php?action=check_if_logged

Ker smo prijavljeni, mora vrniti {"error":0}

http://www.namisljeniportal.si/pot-do-skripte.php?action=identification

Zahteva mora vrniti podatke o trenutno prijavljenim uporabniku.

Če se na spletni strani odjavimo, morata zgornja url-ja vrniti napako: {"error":1}

4.4 Avtomatsko generiranje skripte za prenos seje

V tehnologiji LAMP je izdelan vmesnik, ki bo lastnikom spletne strani omogočil povsem enostavno namestitev klepetalnice na katerokoli spletno stran. Klepetalnica je tako na katerikoli spletni strani postavljena v roku nekaj minut. Strežnik LAMP mora biti sposoben komunicirati s podatkovno bazo strežnika

RED5, saj ji bodo administratorji prek administracije spreminjali parametre (npr. kdo je moderator v sobi itd.). Ker je direkten dostop do baze klepetalnice onemogočen, je bilo potrebno napisati protokol, preko katerega lahko vršimo spremembe na strežniku klepetalnice. V ta namen je tudi na strežniku RED5 postavljen strežnik Apache, ki bo znal izvrševati SQL poizvedbe, ki jih pošilja LAMP strežnik.

4.4.1 Postopek avtomatske namestitve

Postopek namestitve je voden preko spletnega vmesnika, kjer uporabnik vnaša parametre svoje spletne strani, naš sistem pa mu generira skripto, ki je prilagojena samo njemu. Prvi del generiranja skripte se vrši na našemu strežniku, ko uporabnik izbira, kakšno tehnologijo uporablja na svojemu strežniku. Kasneje mora generirano skripto prenesti na svoj strežnik, vnesti podatke za dostop do podatkovne baze in izbrati, v kateri sejni spremenljivki je shranjena seja uporabnika. Namestitvena skripta sama poskuša izbrati prave vrednosti, tako da uporabnik v večini primerov le potrjuje predlagane nastavitve.

Slika 4.3: Uporabnik vnese domeno spletne strani in izbere tehnologijo, na kateri je spletna stran razvita (trenutno je razvit samo generator za PHP in MySQL, ki sodita med najbolj razširjeni orodji za razvoj spletnih strani).

Slika 4.4: Uporabniku se zgenerira skripta, katero mora prenesti na strežnik spletne strani (navadno preko FTP protokola).

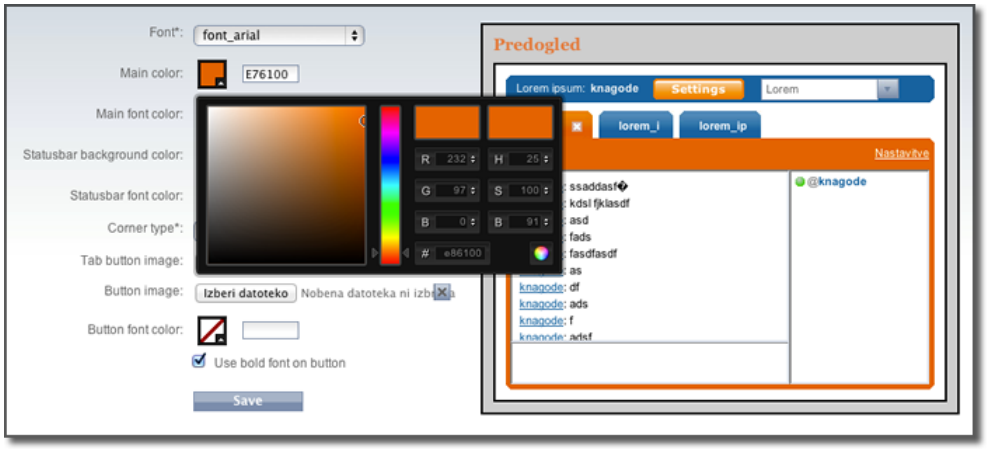
Slika 4.5: Skripta zahteva uporabniško ime in geslo za podatkovno bazo.

Slika 4.6: Skripta analizira podatkovno bazo in skuša ugotoviti, v kateri tabeli se nahajajo podatki o uporabnikih, ter polja v tabeli, kjer se nahaja vzdevek in avatar uporabnika.

Slika 4.7: Skripta analizira tabelo uporabnika ter skuša ugotoviti, kje se nahaja vzdevek in avatar uporabnika.

Slika 4.8: Skripta analizira sejo uporabnika in skuša ugotoviti, v kateri sejni spremenljivki o uporabnikovem id-ju. !!!!

Slika 4.9: Po vnesenih podatkih se na strežniku lastnika spletne strani avtomatsko generira skripta z nastavitvami, ki bo znala prenesti sejo tretje strani na Red5 strežnik.



Slika 4.10: Opcijsko si lahko lastnik spletne strani izbere tudi barvno shemo klepetalnice, izbere jezik in slikice za ključne gumbe - tako je klepetalnica lahko na las podobna vsaki spletni strani.

4.5 Uporabljene tehnologije

4.5.1 PHP

PHP (trenutno tričrkovni rekurzivni akronim za PHP Hypertext Preprocessor, izvirno pa Personal Home Page Tools, slovensko orodja za osebno spletno stran) je razširjen odprtokodni programski jezik, ki se uporablja za strežniške uporabe oziroma za razvoj dinamičnih spletnih vsebin. Lahko ga primerjamo z Microsoftovim sistemom ASP, VBScript in JScript, Sun Microsystemovim sistemom JSP in Java ter sistemom CGI in Perl. Danes je PHP eno izmed najpopularnejših orodij za izdelavo spletnih strani, PHP pa med drugim uporabljajo tudi najbolj obiskane spletne strani (Facebook.com). [**?**]

4.5.2 MySQL

MySQL (je svetovno najbolj pogosto uporabljen sistem za upravljanje podatkovne baze, ki zagotavlja dostop do podatkovne baze več uporabnikov. Sistem je poimenovan po hčeri razvijalca Michael-a Widenius-a. SQL ali strukturirani povpraševalni jezik za delo s podatkovnimi bazami (angl. Structured Query Language) je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkovnimi zbirkami, s programskimi stavki, ki posnemajo ukaze v naravnem jeziku. Določen je z ANSI/ISO SQL standardom.[**?**]

4.5.3 Apache

Apache je programska oprema za strežnik, ki igra ključno vlogo pri razvoju svetovnega spleta. Leta 2009 je Apache programska oprema oskrbovala več kot 100 milijonov spletnih strani, kar Apache postavlja med svetovno najbolj uporabljen spletni strežnik. Apache je bil prva resna alternativa Netscape Communications Corporation web server (danes Oracle iPlanet Web Server), sedaj pa je že razvit v številne veje. Apache privzeto deluje na UNIX operacijskih sistemih, vendar so različice strežnika razvite za vse popularne operacijske sisteme (vključno Microsoft Windows in Apple OSX). [9]

Poglavje 5

Strežnik za dvosmerno komunikacijo

Do prihoda standarda HTML5, ki ga podpira vse več brskalnikov, HTML standard ni vseboval specifikacije, ki bi omogočala odprto povezavo s strežnikom za komunikacijo v realnem času. Slabost protokola je torej, da odjemalec brez težav pošlje sporočilo na strežnik, ko pa želi strežnik poslati sporočilo odjemalcu, pridemo do problema, ki ga je možno rešiti na več načinov.

- HTTP pooling
- Comet
- HTML5 websockets
- RTMP protokol
- RTMFP protokol

5.1 Možne implementacije dvosmerne komunikacije

Strežnik za dvosmerno komunikacijo je implementiran v okolju RED5, ki poleg enostavnih tekstovnih sporočil omogoča tudi avdio in video komunikacijo. Predstavili bomo še ključne alternativne koncepte, od katerih ima vsaka tako prednosti kot slabosti.

5.1.1 HTTP pooling

HTTP pooling je koncept, s katerim bomo navidezno ustvarili obojestransko komunikacijo s strežnikom. Ustvari se skripta, ki na vsakih nekaj sekund vpraša strežnik, če zanj obstaja novo sporočilo. V primeru, da obstaja sporočilo, ga strežnik vrne, če ne, vrne prazen odgovor. Navidezno zgleda, kot da strežnik v vsakem trenutku lahko pošlje sporočilo odjemalcu, vendar pa ima opisani način številne pomanjkljivosti. Zakasnitve so v povprečju zelo visoke in so sorazmerne z dolžino intervala preverjanja novega sporočila. V kolikor bi želeli manjše zakasnitve, moramo skrajšati interval, kar pa obenem pomeni tudi bistveno večjo obremenitev za strežnik.

Če je "pooling" interval nastavljen na 1s, procesiranje zahtevka na strežnikovi strani pa traja 10ms, vidimo, da je strežnik polno obremenjen že pri 100 uporabnikih naše aplikacije, ki pa bi kljub zelo kratkemu intervalu še vedno imela ogromne zakasnitve, strežnik pa bi bil polno zaseden ne glede na intenzivnost komunikacije med računalniki. Za razliko od drugih pristopov je HTTP pooling enostaven za realizacijo in ne potrebuje posebne logike na strežnikovi strani - uporabimo lahko PHP ali podoben skriptni jezik.

5.1.2 Comet

Comet je princip, ki je sicer zelo podoben HTTP pooling-u, le da ima komunikacija manjše zakasnitve in zaseda manj strežnikovih resursov.

Odjemalec pošlje na strežnik vprašanje, če je zanj kakšno sporočilo, strežnik pa z odgovorom čaka, vse dokler za odjemalca nima sporočila. Ko dobi sporočilo zanj, mu ga vrne, odjemalec pa nato takoj spet vzpostavi povezavo in čaka na nova sporočila. Odjemalec ne potrebuje dodatnih vtičnikov, ki bi omogočali tak način komunikacije, saj je vsa potrebna funkcionalnost že vgrajena v JavaScript tudi pri starejših brskalnikih. Seveda pa strežniki in sorodne tehnologije niso bili razviti za tak način komunikacije, zato je za optimalno delovanje potrebno uporabiti strežnik, ki je narejen izključno za opisan problem (npr. Tomcat server: http://tomcat.apache.org/).

Comet je pod imenom Channel API podprt tudi v Google App Engine-u, kateri nam omogoča izdelavo visoko skalabilnih internetnih aplikacij v oblaku.

Ker Comet uporablja izključno HTML standarde, pa zaradi tega ni mogoče, da bi preko povezave pošiljali tudi zvok ali video.

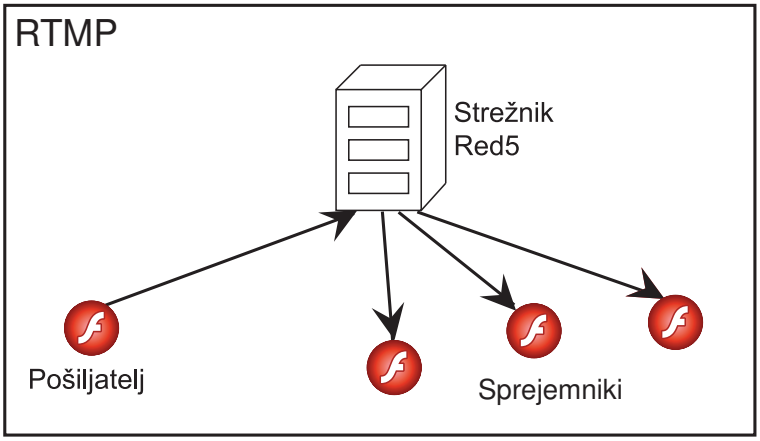
Protokol uporabljajo številne popularne storitve, med drugim tudi slovenski Vox.io, ki preko RTMP protokola omogoča brezplačno telefoniranje direktno iz brskalnika.

5.1.3 WebSockets

S HTML5 je W3C standariziral tudi WebSockets, ki je prihodnost za obojestransko komunikacijo odjemalec-strežnik. Povezava s strežnikom je persistentna, kar zagotavlja minimalne zakasnitve in hitro odzivnost strežnika. Moderni brskalniki že omogočajo uporabo WebSockets-ov, starejši brskalniki pa se lahko na WebSocket strežnik povežejo tudi preko Flash-a, kar je dobro izkoristila JavaScript platforma node.js.

5.1.4 RTMP

RTMP (Real Time Messaging Protocol) bazira na TCP protokolu in omogoča stalno povezavo med odjemalcem in strežnikom, kar pomeni dvosmerno komunikacijo z nizko latenco in hitro odzivnostjo. Prvotno je bil namenjen za procesiranje in prenos video in avdio vsebin, kasneje pa je bil protokol razširjen in je začel omogočati tudi druge tipe podatkov. Protokol je razvilo podjetje Macromedia (danes v lasti Adobe) za svoj zelo razširjen vtičnik Flash, v navezavi z Macromedia Media Server (http://www.adobe.com/products/flash-media-server-family.html). Hitro so bile razvite tudi odprtokodne rešitve, ki so uporabljene tudi v realizaciji diplomske naloge (Red 5 strežnik). Protokol omogoča tudi enkripcijo ter tuneliranje preko HTTP protokola.



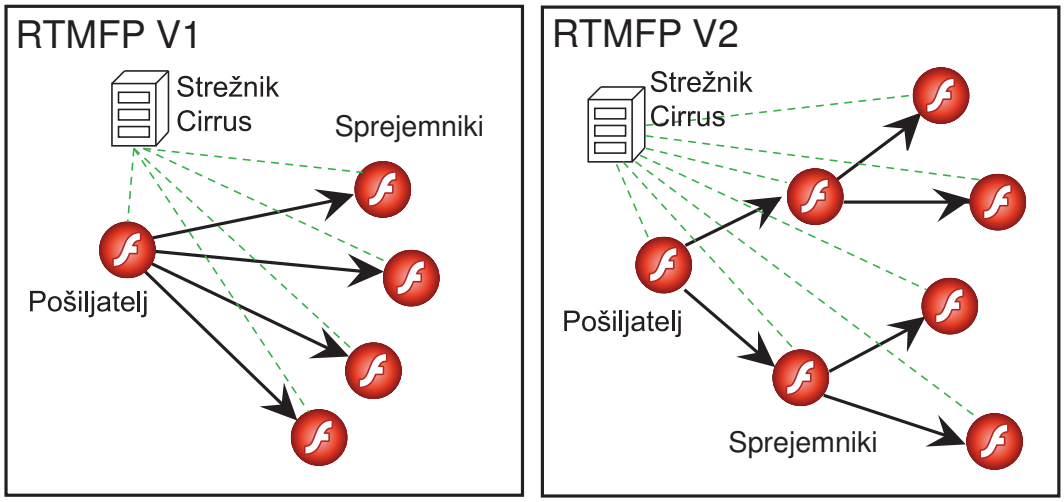
Slika 5.1: Pošiljanje sporočil preko protokola RTMP se vedno izvede preko strežnika. Odjemalec vedno pošlje le eno sporočilo, katero strežnik pošlje vsem drugim odjemalcem, ki morajo sporočilo prejeti.

5.1.5 RTMFP

Adobe poleg protokola RTMP razvija tudi protokol RTMFP (Real Time Media Flow Protocol), ki nastaja pod okriljem projekta Cirrus od leta 2008. Protokol omogoča komunikacijo odjemalcev brez posredovanja strežnika, kar razbremeni strežnik, ker pa se znebimo tudi vmesnega člena v omrežju, je komunikacija hitrejša in odzivnejša. Največjo razbremenitev strežnika lahko pričakujemo, če protokol uporabimo za uporabo video ali avdio storitev.

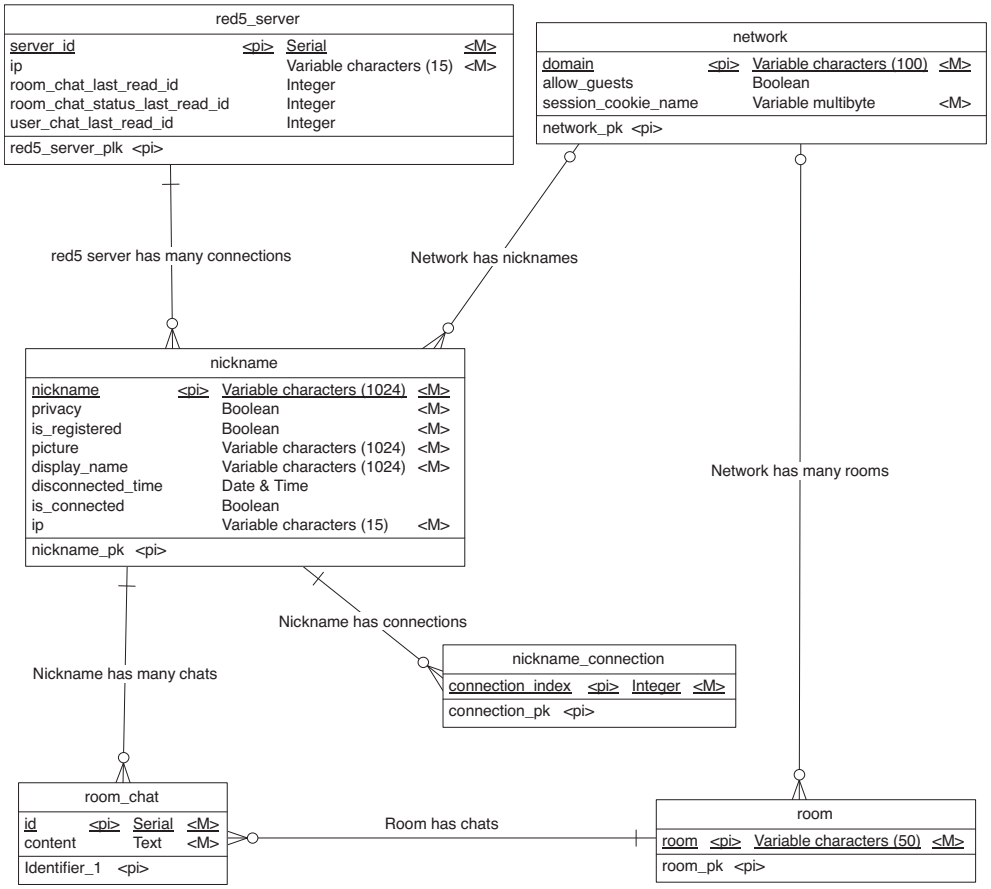
Protokol je podprt v Adobe Flash Player-ju 10, ki je danes nameščen na 90% računalnikov, ki uporabljajo internet. Kljub temu, da protokol omogoča neposredno komunikacijo med dvema odjemalcema, še vedno potrebujemo strežnik, preko katerega se inicializira povezava, po tem pa komunikacija deluje brez posrednika. Strežnik, ki omogoča protokol RTMFP je npr. Cirrus strežnik. [1]

Za razliko od protokola RTMP, ki temelji na TCP protokolu (Transmission Control Protocol), RTMFP protokol temelji na protokolu UDP (User Datagram Protocol), kar pomeni manjšo latenco in manjši *overhead*, vendar je zaradi opisanih prednosti protokol manj zanesljiv.



Slika 5.2: Protokol RTMFP omogoča komunikacijo odjemalcev brez posrednika, kar bistveno razbremeni strežnik. Prva verzija protokola je delovala tako, da pošiljatelj preko omrežja pošlje sporočilo vsem prejemnikom. Protokol je v naslednjih verzijah nadgrajen tako, da pošiljatelj pošlje sporočilo samo parim sprejemnikom, sprejemniki pa nato reproducirajo sporočilo drugim prejemnikom.

5.2 Konceptualni model klepetalnice



Slika 5.3: Poenostavljen konceptualni model prikazuje osnovne entitete in relacije med njimi. Zaradi preglednosti so prikazane le ključne entitete.

5.3 Red5

Red5 Media server je strežnik, ki omogoča obojestransko komunikacijo odjemalca s strežnikom preko RTMP. Strežnik temelji na tehnologiji Java in tudi strežniške programe se tako razvija kar v Javi. S pomočjo strežnika lahko razvijamo aplikacije za video konference, multiplayer igre, klepetalnice in druge aplikacije, ki tečejo v realnem času. Red5 je postavljen na strežniku Windows Server 2003, možno pa ga je postaviti tudi na platformo Linux.

5.3.1 Uporaba Red5

Strežnik sam po sebi že skrbi za kreiranje novih niti ob vzpostavitvi nove povezave, sami se moramo osredotočiti le na funkcionalnost naših skript.

Strežnik Red5 poganja več aplikacij, med katerimi je vsaka v svojem direktoriju znotraj direktorija za aplikacije. Ob zagonu strežnik Red5 pregleda cel direktorij z aplikacijami ter aplikacije tudi zažene. Vse aplikacije se morajo držati pravilne datotečne strukture, v mapi WEB-INF pa so navedene tudi vse nastavitve aplikacije. Najpomembnejša datoteka je red5-web.xml, ki opiše našo aplikacijo in pove, kateri razred se mora klicati ob zagonu strežnika.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.
       springframework.org/dtd/spring-beans.dtd">
3  <beans>
4
5    <bean id="placeholderConfig" class="org.springframework.beans
       .factory.config.PropertyPlaceholderConfigurer">
6       <property name="location" value="/WEB-INF/red5-web.
          properties" />
7    </bean>
8
9    <bean id="web.context" class="org.red5.server.Context"
       autowire="byType" />
10
11   <bean id="web.scope" class="org.red5.server.WebScope"
12      init-method="register">
13     <property name="server" ref="red5.server" />
14     <property name="parent" ref="global.scope" />
15     <property name="context" ref="web.context" />
16     <property name="handler" ref="web.handler" />
17     <property name="contextPath" value="${webapp.contextPath}"
          />
18     <property name="virtualHosts" value="${webapp.virtualHosts}
          " />
19   </bean>
20
21   <bean id="web.handler"
22       class="chat.Application"
23     singleton="true" />
24
25
26
27 </beans>
```

Listing 5.1: Nastavitve za Red5 aplikacijo

Programiranje aplikacije v okolju Red5 se izvaja v Javi, za razvojno okolje pa smo uporabili Eclipse. Ob začetku programiranja moramo razširiti razred ApplicationAdapter in znotraj njega prepisati osnovne metode, ki so potrebne za delovanje strežniške aplikacije:

5.3.2 Zagon strežnika

```
1 public boolean appStart (IScope app )
```

V metodi je definirana vzpostavitev povezave z bazo, zažene se nit za preverjanje novih sporočil v bazi. Ob zagonu je potrebno preveriti, če je predhodna zaustavitev sistema bila nenamerna, ter v tem primeru sistem postaviti v konsistentno stanje: v podatkovni bazi pobrišemo vse povezane uporabnike, pobrišemo neposlana sporočila in čakamo, da se uporabniki ponovno povežejo na strežnik.

5.3.3 Odjemalec želi vzpostaviti povezavo s strežnikom

```
1 public boolean appConnect( IConnection conn , Object[] params )
```

Vsakič, ko se odjemalec poveže na naš strežnik, se kliče metoda appConnect, v kateri moramo odobriti ali zavrniti njegovo zahtevo. Odjemalec nam posreduje tudi sejni piškotek tretjega strežnika. Red5 na URL tretjega strežnika pošlje zahtevo za identifikacijo sejnega piškotka. Vrnjene podatke si Red5 kešira, tako da ob krajši prekinitvi povezave z odjemalcem ni potrebno pošiljati zahtevkov na tretji server.

V primeru, da je bila seja že predhodno začeta (prišlo je do prekinitve), mora strežnik odjemalcu poslati tudi vse potrebne podatke, da se seja klepeta prenese do odjemalca. Prvi argument v funkciji predstavlja odprto povezavo do odjemalca, ki jo bomo uporabili za pošiljanje in sprejemanje sporočil. Aktivno povezavo dodamo v seznam, indeks povezave uporabnika pa shranimo v podatkovno bazo. Implementiran je tudi seznam indeksov neaktivnih povezav, tako da vsaka povezava ne dobi vedno novega indeksa, ampak se lahko uporabi tudi indeks že prekinjene povezave, s čimer zagotovimo minimalno porabo pomnilnika. Ker v podatkovni bazi hranimo indeks, je časovna zahtevnost za dostop do povezave uporabnika O(1).

V primeru, da se uporabnik z isto sejo na strežnik prijavi dvakrat, je za istega uporabnika treba hraniti več povezav. Naša aplikacija bo tako delovala tudi v primeru, da ima uporabnik odprt klepet v več zavihkih v brskalniku.

5.3.4 Odjemalčeva povezava je prekinjena

```
1 public void appDisconnect( IConnection conn)
```

Ko uporabnik prekine povezavo s strežnikom, se kliče metoda appDisconnect. V tej metodi je potrebno vse uporabnike, ki so v interakciji z dotičnim uporabnikom, obvestiti, da je nek uporabnik prekinil svojo sejo. Uporabnik mora biti odstranjen iz vseh sob, v katerih je bil aktiven, odstranjen mora biti iz vseh pogovorov, kjer je bil aktiven. Ker želimo, da se seja uporabnika obnovi po krajši prekinitvi, vseh podatkov ne odstranimo takoj po prekinitvi, ampak te hranimo še 30s. V kolikor se v temu času uporabnik ponovno poveže na naš strežnik, se seja obnovi, v nasprotnem primeru pa sejo dokončno pobrišemo. Če ima uporabnik aktivnih več povezav, se sejo pobriše šele takrat, ko so bile prekinjene vse povezave z našim strežnikom (primer, ko ima uporabnik odprtih več oken s klepetalnico in zapre eno izmed oken).

Ob prekinitvi moramo povezavo odstraniti iz seznama aktivnih povezav, indeks povezave pa vstavimo na sklad neaktivnih povezav, kar zagotavlja, da naslednja povezava v seznamu aktivnih povezav zasede mesto, ki je bilo v temu trenutku prekinjeno. Tako dosežemo minimalno porabo pomnilnika za hranjenje aktivnih povezav.

5.3.5 Pošiljanje sporočil odjemalcu

```
1 public void serverToClient(IConnection conn, String JSON)
```

Vsako sporočilo se odjemalcu pošlje preko te funkcije, format poslanega sporočila pa je JSON. Ker naša aplikacija podpira več povezav istega uporabnika, je potrebno zagotoviti, da se vsa sporočila razpošljejo na vse odjemalčeve povezave.

5.3.6 Prejemanje sporočil odjemalca

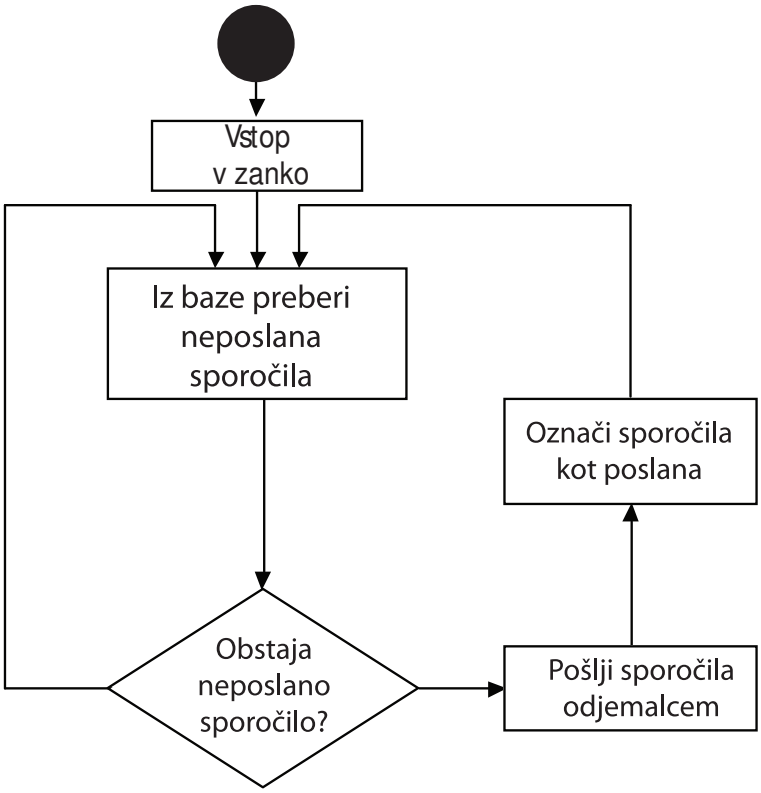
```
1 public void clientToServer(IConnection conn, String JSON)
```

Ko odjemalec pošlje sporočilo strežniku, je potrebno sporočilo shraniti v tabelo neposlanih sporočil. Ob prejemu sporočila je potrebno preveriti, ali ima uporabnik odprtih več povezav do strežnika, in v tem primeru prejeto sporočilo replicirati na vse ostale povezave odjemalca. Tako dosežemo, da odjemalec v vseh svojih zavihkih vidi identični prikaz klepeta. Sporočilo bo odjemalcem posredovano v naslednji iteraciji pošiljanja.

5.3.7 Realizacija prejemanja in pošiljanja sporočil

V strežniku teče zanka, v kateri se za vsakega povezanega uporabnika konstantno preverja, če je zanj prispelo kakšno novo sporočilo, in mu ga v istem

trenutku posreduje. Strežnik si shrani ID zadnjega poslanega sporočila, v naslednji iteraciji zanke pa tako poskusi razposlati samo sporočila, ki imajo večji ID od zadnjega poslanega. Strežnik mora imeti zagotovljen tudi sistem za brisanje starejših sporočil, ker ta po več kot nekaj urah niso več uporabljena. Naš sistem bi lahko sicer posredoval sporočila takoj - brez uporabe hranjenja sporočil v podatkovno bazo, vendar bi s tem izgubili možnost obnavljanje seje uporabnika, poleg tega pa z uporabo podatkovne baze omogočimo, da bo sistem v prihodnosti tudi skalabilen.



Slika 5.4: Ob prihodu sporočila od uporabnika se sporočilo shrani v tabelo sporočil in nato čaka na cikel, da bo sporočilo poslano drugim uporabnikom

5.3.8 Realizacija pregleda statusa uporabnikov

Vsakič, ko uporabnik vstopi v določeno sobo, se vsem uporabnikom sobe pošlje tudi sporočilo, da je nov uporabnik prišel v kanal, in informacijo o njegovemu trenutnemu statusu (dosegljiv, zaseden, odsoten). Ob vsaki spremembi statusa

uporabnika se vsem uporabnikom, ki so v interakciji, pošlje informacija o novem statusu - status uporabnikov je tako v veliki meri zelo podoben pošiljanju sporočil.

Ob vsaki spremembi statusa mora strežnik preveriti, v katerih sobah je uporabnik navzoč, in vsem uporabnikom v teh sobah poslati obvestilo o spremembi statusa uporabnika.

5.3.9 Realizacija vstopa uporabnika v določeno sobo

Ob vstopu uporabnika v sobo je potrebno shraniti, da je uporabnik vstopil v sobo. V istem trenutku je potrebno vsem uporabnikom, ki so že v sobi, poslati tudi sporočilo, da je nek uporabnik prišel v sobo.

Pred vključitvijo v sobo je potrebno preveriti tudi, če je uporabnik ali IP naslov na listi nezaželenih, in mu v tem primeru onemogočiti vstop.

5.4 Uporabljena tehnologija

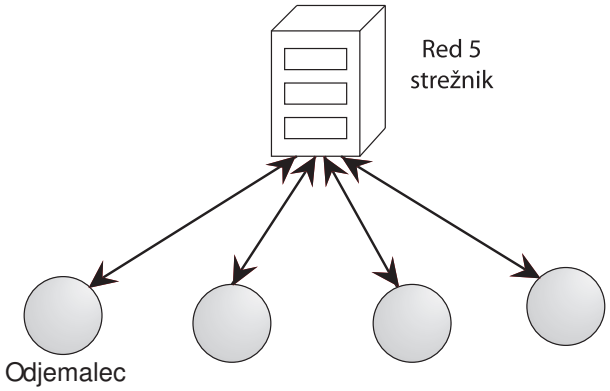
5.4.1 PostgreSQL

Strežnik za dvosmerno komunikacijo uporablja PostgreSQL, ki je sistem za upravljanje podatkovnih baz in se je v okviru testiranja hitrosti (vstavljanje v podatkovno bazo, ki bo ozko grlo sistema) izkazal za boljšega od MySQL. Prav tako omogoča tudi replikacijo in master-master povezavo med strežniki, ki jo bomo potrebovali v primeru, da bi bilo skaliranje in redundanca potrebna.

Poglavje 6

Razširitev klepetalnice v distribuiran računalniški sistem

Trenutno je naš sistem klepetalnice realiziran kot strežnik, ki obdeluje zahteve. Ker pričakujemo, da bo število uporabnikov storitve naraščalo, bo strežnik v nekem trenutku prišel do točke nasičenja - točke, ko strežnik ne zmore obdelati vseh zahtev. Zmoglijvost strežnika se lahko nekaj časa izboljšuje tako, da izboljšamo strojno opremo (povečamo pomnilnik, dokupimo procesorsko moč in število procesorjev), vendar pa cena boljše strojne opreme eksponentno narašča, poleg tega pa bomo tudi z izjemno drago opremo hitro prišli do točke nasičenja. Namesto drage opreme bomo zato implementirali distribuiran računalniški sistem, ki bo zmogel obdelati bistveno večje število zahtev.

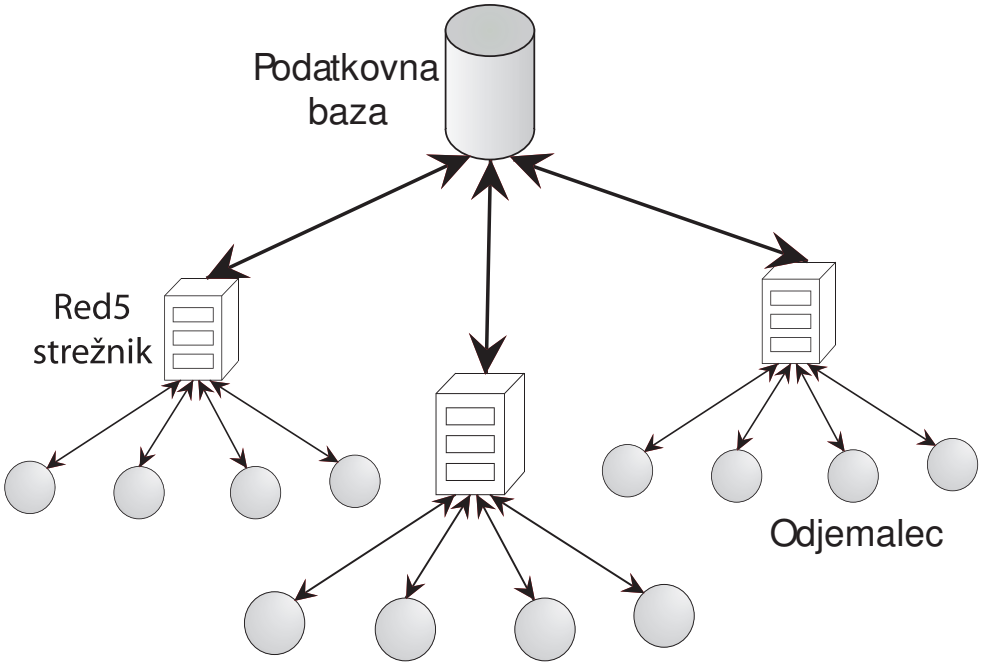


Slika 6.1: Neskalabilen sistem je predstavljen kot strežnik, na katerega se povezujejo odjemalci.

6.1 Koncept skupnega pomnilnika

Naš sistem bi radi porazdelili na več strežnikov, kjer naletimo na težave, saj moramo zagotoviti porazdeljeno medprocesno komunikacijo. Problem bomo rešili s konceptom skupnega pomnilnika, ki bo realiziran kot podatkovna baza, ki jo bodo videla vsa vozlišča v gruči.

Sistem bo ob majhnem številu uporabnikov počasnejši, vendar pa je prednost sistema ta, da bo zmožen obdelati bistveno več zahtev uporabnikov, kot en sam strežnik, ki podatke hrani v pomnilniku. V tem koraku naš sistem zgleda kot centralizirana podatkovna baza, iz katere berejo Red5 strežniki.



Slika 6.2: Shema prikazuje koncept skupnega pomnilnika, ki je realiziran kot podatkovna baza, na katero je povezanih več RED5 strežnikov, kar naš sistem že naredi distribuiran.

6.2 Porazdeljena podatkovna baza

Procesiranje smo v prejšnjem poglavju že porazdelili na več računalnikov, vendar pa je podatkovna baza ostala neporazdeljena, zato bo naš sistem še vedno prišel do točke nasičenja. Da bi zagotovili porazdeljeno procesiranje tudi na nivoju podatkovne baze, bomo uporabili porazdeljeno podatkovno bazo.

Porazdeljena podatkovna baza (PPB) predstavlja nabor več logično povezanih podatkovnih baz, fizično razpršenih (porazdeljenih) po računalnikih, povezanih z računalniškim omrežjem.

6.2.1 Prednosti PPB

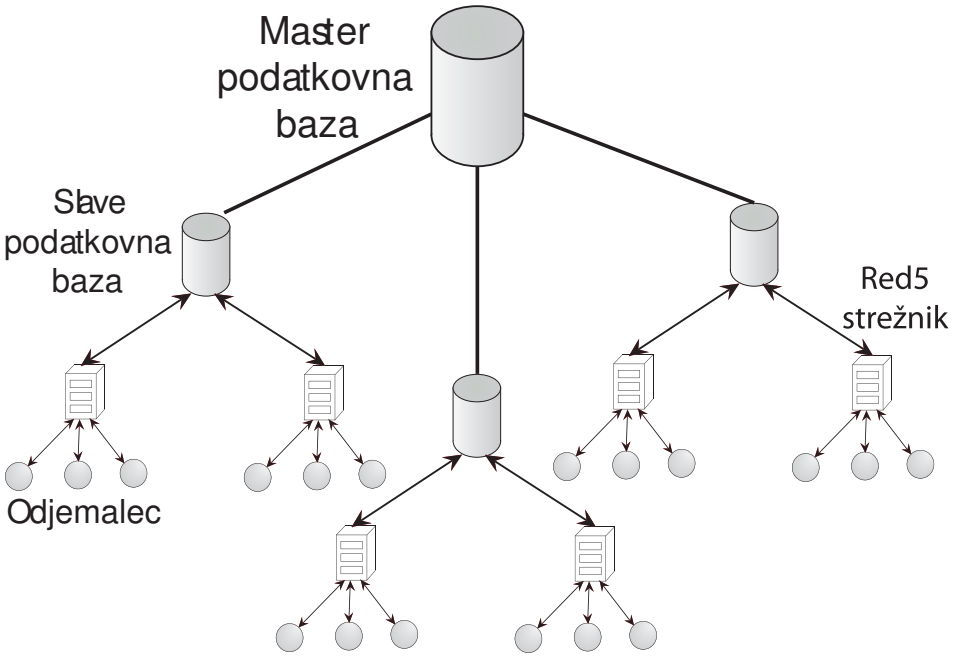
- Lahko izražajo porazdeljeno organizacijsko strukturo
- Izboljšajo porazdeljenost (dostop do vseh podatkov) in avtonomnost (podatki tam, kjer se uporabljajo)
- Izboljšajo razpoložljivost (sistem je porazdeljen. Če odpove eno vozlišče, so druga še vedno dostopna)
- Izboljšajo zanesljivost (replikacija)
- Izboljšajo učinkovitost (podatki porazdeljeni glede na poizvedovanje, paralelnost,...)

6.2.2 Replikacija

O replikaciji govorimo, kadar ima isti podatek v sistemu lahko več fizičnih kopij, ki so na različnih lokacijah. [7]

Pisanje se izvaja na primarno lokacijo, ki spremembe shrani še na vse kopije. Branje se tako izvaja iz poljubne lokacije, medtem ko moramo v programu zagotoviti, da se pisanje vedno izvaja na primarno lokacijo. Ker je tipična lastnost informacijskih sistemov ta, da je pisanj v računalniški sistem bistveno manj kot branj, lahko z replikacijo bistveno izboljšamo zmožnost skaliranja informacijskega sistema. Glede na to, da uporabljamo replikacijo, je šibka točka naše aplikacije primarna podatkovna baza, na kateri izvajamo spremembe. Da bo naš sistem hiter, je potrebno paziti, da imamo takih poizvedb čim manj.

Redundanca ima tudi slabosti, saj je treba vsak podatek shraniti na več fizičnih lokacij, kar pomeni večjo porabo prostora na diskih. Kljub slabostim pa ima kontrolirana redundanca v praksi precejšen pomen, saj pohitri poizvedbe. [5]



Slika 6.3: Prikaz skalabilnega sistema, kjer je skupni pomnilnik realiziran kot gruča podatkovnih baz. Pisanje se izvaja izključno na master podatkovno bazo, branje pa poteka iz katerekoli podatkovne baze. Ker je pisanj precej manj kot branj, z replikacijo dosežemo zelo očitno pohitritev sistema

6.3 Okvirna meritev maksimalne zmogljivosti našega sistema

Da bi izračunali okvirno zmogljivost našega sistema, ob predpostavki, da je ozko grlo primarna podatkovna baza, bomo najprej izmerili frekvenco procesiranja primarne lokacije in nato ocenili, koliko uporabnikom bi lahko stregli.

Meritve so pokazale, da lahko na računalniku z zmogljivostjo 1.3 GHz , 716 MB RAM-a in HDD diskom s 5400 RPM v podatkovno bazo v eni sekundi vstavimo približno 1000 različnih vnosov.

Frekvenca procesiranja je:

$$\nu_{procesiranje} = 1000/s$$

Privzemimo, da za aktivnega uporabnika velja, da sporočilo pošilja na vsa-

kih 10s. Frekvenca pošiljanja sporočil (vstavljanja v PB) enega uporabnika:

$$\nu_{uporabnik} = 0,1/s$$

Iz slednjih podatkov lahko direktno izračunamo, koliko uporabnikom bi lahko sistem stregel:

$$N = \frac{\nu_{procesiranje}}{\nu_{uporabnik}}$$

$$N = 10.000$$

Že prva ocena števila uporabnikov je zadovoljiva in če privzamemo, da je aktivni uporabnik v povprečju aktiven le desetino dneva, lahko pričakujemo, da bo sistem sposoben dnevno procesirati še desetkrat več uporabnikov.

$$N_{realno} = 100.000$$

Meritve so bile izvedene na osebnem računalniku, na katerem podatkovna baza ni imela optimizirane konfiguracije in optimalne strojne opreme (npr SSD disk).

6.4 Dodatna skalabilnost

Glede na analogijo s popularno igro Draw Something, ki ima dnevno 15 milijonov uporabnikov, vidimo, da naš zastavljen sistem še vedno ne bi mogel obdelati toliko zahtev, kot jih lahko nastane v primeru popularnih aplikacij.

Dodatna strojna oprema nam v primeru replikacije ne pomaga več, razen če se osredotočimo na nivo podatkovnega modela in ga skušamo razdeliti na med seboj neodvisne dele, le-te pa postavimo na ločene fizične lokacije. Vsak Red5 strežnik se tako poveže na več primarnih lokacij. Maksimalno razpršitev sheme dobimo, če vsako relacijo premaknemo na svojo fizično lokacijo. Podatke v bazi lahko razpršimo tudi horizontalno - podatke shranimo na različne fizične lokacije glede na vrednost primarnega ključa.

Naša arhitektura je tako vse bolj podobna konceptu NoSQL, poznanem kot sistem za podatkovne baze, ki omogoča obdelavo ogromnih količin podatkov. Uporaba NoSQL zahteva korenite spremembe v programu, saj NoSQL ne uporablja standardnega SQL za povpraševanje po podatkih. Največja sprememba je, da v SQL ne moremo več delati stikov in drugih kompleksnejših operacij nad relacijami.

Deloma smo že poskrbeli za to, da je posamezne tabele možno premakniti na različne fizične lokacije: pošiljanje zasebnih klepetov, klepetov v sobi in statusnih sporočil nismo dali v skupno tabelo, ampak v tri ločene. Po potrebi bi lahko glede na vrsto sporočil še dodatno razbili shemo.

6.4.1 NoSQL

NoSQL ne uporablja jezika SQL za povpraševanje po podatkih. NoSQL podatkovni sistemi so nastali za potrebe glavnih internetnih podjetij, kot so Google, Amazon in Facebook, ki se ukvarjajo z obdelavo ogromnih količin podatkov, katerim sistem za upravljanje podatkovnih baz ni več kos. Podatki so razpršeni na več fizičnih lokacij, poizvedovanje pa ne omogoča kompleksnih operacij (npr stik). Večina NoSQL podatkovnih baz je razvitih z namenom, da tečejo na gručah računalnikov, zato morajo biti porazdeljene in odporne proti napakam. Za ta namen je potrebno sprejeti kompromise glede ACID lastnosti, upravljanja transakcij in zmogljivosti poizvedb. Običajno so namenjene spletnim aplikacijam, so skalabilne, ne zahtevajo prilagodljivejše podatkovne sheme in so večinoma odprtokodne. [6]

6.5 Nadzor delovanja informacijskega sistema

Vsak informacijski sistem je podvržen možnostim napak, zato moramo poskrbeti, da bo sistem v primeru napak ustrezno ukrepal in postavil sistem v konsistentno stanje.

Do napak v sistemu lahko pride zaradi:

- Napake v omrežju
- Napake na disku
- Napake v podatkovni bazi
- Napake na strojni opremi
- Napake v programski opremi
- Težave z električno energije
- Preobremenjenost sistema
- Človeške napake

6.5.1 Detektor napak

Nadzornik je programski modul, ki delno ali v celoti nadzoruje delovanje sistema in procesiranje podatkov. [8]

Več tednov preizkušanja delovanja klepetalnice je pokazalo, da se napake v trenutni različici pojavljajo zaradi uporabe odprtokodnih knjižnic in programske opreme, ki ne upravljajo pravilno s pomnilnikom. Poraba pomnilnika se zato konstantno veča, po obdobju nekaj tednov pa sistem pade.

Za detekcijo napak je izdelan poseben odjemalec, ki teče na vsakem strežniku Red5 in po konceptu srčnega utripa skuša s strežnikom ustvariti povezavo vsakih 10s. Če je servis neodziven, lahko predvidevamo, da je prišlo do napake v sistemu, in ponovno zaženemo Red5 strežnik. Vse povezave med odjemalci in strežnikom se prekinejo, ker pa je tudi odjemalec sposoben ponovno vzpostaviti povezavo s strežnikom, je za končnega uporabnika motnja v sistemu nevidna.

Poseben odjemalec za detekcijo je spisan v tehnologiji Flash in teče na strežniku. Ker je OS (operacijski sistem) Red5 strežnikov Windows Server 2003, je potrebno Flash datoteko pretvoriti v .exe format, ki se ga zažene vsakih 10s preko upravitelja opravil. Ker Flash nima pravic za izvajanje terminalnih ukazov, je potrebno ustvariti tudi .bat skripto, katero lahko požene tudi Flash. V .bat datoteki je program za ponoven zagon servisa Red5, ki se zažene v primeru, da se na RED5 strežnik po več poskusih ne more uspešno povezati.

```
1 net stop Red5 | net start Red5
```

Poglavje 7

Sklepne ugotovitve

V diplomski nalogi je bil razvit sistem za storitev v realnemu času, ki lastnikom spletnih strani omogoča, da na svojo spletno stran vključijo klepetalnico v nekaj minutah. Strežnik za dvosmerno komunikacijo vsebuje funkcionalnost pošiljanja sporočil skupini ali posameznemu uporabniku, kar lahko s pridom uporabimo za reševanje najrazličnejših problemov. Razvita rešitev je še posebej uporabna za razvoj dvopoteznih multiplayer iger, ki so z uporabo naših knjižnic lahko razvite v nekaj dneh.

Poleg razvite storitve, ki omogoča dvosmerno komunikacijo, je razvit tudi mehanizem, ki distribuira procesiranje na več strežnikov in tako zagotavlja večjo skalabilnost in zanesljivost. Da smo zagotovili maksimalno dosegljivost sistema, smo celotno storitev opremili tudi z distribuiranim nadzornim sistemom, ki skrbi, da se sistem v primeru napak sam postavi v konsistentno stanje.

Ključne segmente storitve smo opremili z UML diagrami (konceptualni modeli, razredni diagrami, primeri uporabe ...), ki omogočajo, da se v projekt brez težav vključijo tudi drugi razvijalci.

V prihodnosti se bomo osredotočili na razvoj odjemalca za mobilne naprave, katerih uporaba danes strmo narašča. Zmožnosti naše storitve bi bilo potrebno preslikati v komercialno zanimiv produkt in ga lansirati na trg. V kolikor bi bila storitev zanimiva na globalnem trgu, bi bilo potrebno podrobno preučiti sposobnost distribuiranega sistema in mu glede na dejansko uporabo izboljšati komponente, ki predstavljajo ozko grlo sistema.

Slike

Tabele

Literatura

[1] Adobe.com, *"Cirrus"*, 2012. Dostopno na: http://labs.adobe.com/technologies/cirrus/

[2] Apple.com, *"Thoughts on flash"*, 2011. Dostopno na: http://www.apple.com/hotnews/thoughts-on-flash/

[3] Blodget H., *"OMGPop Sold Way Too Early "*, 2012. Dostopno na: http://articles.businessinsider.com/2012-03-21/tech/31218762_1_zynga-app-mobile-game

[4] Crockford D., *"Private Members in JavaScript "*, 2012. Dostopno na: http://javascript.crockford.com/private.html

[5] Elmasri R., Navathe B., *Fundamentals of Database Systems*, Boston: Pearson, 2003, pogl. 1.6.

[6] Knez N., *"Analiza podatkovnih baz NoSQL in izdelava odločitvenega modela za izbiro med relacijskimi in NoSQL podatkovnimi bazami"*, Diplomsko delo, 2012.

[7] Vidmar T., *Informacijsko-komunikacijski sistem*, Ljubljana: Pasadena, 2002, pogl. 11.2.3.

[8] Vidmar T., *Računalništvo v oblaku*, Ljubljana: Pasadena, 2011, pogl. 3.7.

[9] Wikipedia, *"Apache strežnik"*, 2012. Dostopno na: http://en.wikipedia.org/wiki/Apache

[10] Wikipedia, *"CSS"*, 2012. Dostopno na: http://sl.wikipedia.org/wiki/CSS

[11] Wikipedia, *"HTML"*, 2012. Dostopno na: http://sl.wikipedia.org/wiki/HTML

[12] Wikipedia, *"Internet Relay Chat"*, 2012. Dostopno na: http://sl.wikipedia.org/wiki/Internet_Relay_Chat

[13] Wikipedia, *"JavaScript"*, 2012. Dostopno na: http://sl.wikipedia.org/wiki/JavaScript