Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anton Zvonko Gazvoda

Integracijska arhitektura računalniškega oblaka, komunikacijskih in aplikacijskih storitev na primeru IPTV

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Matjaž Branko Jurič

Ljubljana 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

[1] V dogovorju z mentorjem lahko kandidat diplomsko delo s pripadajočo izvorno kodo izda tudi pod katero izmed alternativnih licenc, ki ponuja določen del pravic vsem: npr. Creative Commons, GNU GPL. V tem primeru na to mesto vstavite opis licence, na primer tekst [**?**]

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00021/2012
Datum: 10.04.2012

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

Kandidat: **ANTON ZVONKO GAZVODA**

Naslov: **INTEGRACIJSKA ARHITEKTURA RAČUNALNIŠKEGA OBLAKA, KOMUNIKACIJSKIH IN APLIKACIJSKIH STORITEV NA PRIMERU IPTV**

INTEGRATION ARCHITECTURE FOR CLOUD COMPUTING, COMMUNICATION AND SOFTWARE AS A SERVICE ON IPTV CASE

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Proučite komunikacijske storitve in aplikacijske storitve v računalniškem oblaku. Identificirajte možnosti za integracijo in analizirajte tehnologije platforme Java EE, primerne za izvedbo integracije. Zasnujte integracijsko arhitekturo računalniškega oblaka, komunikacijskih in aplikacijskih storitev. Implementirajte prototip integracijske arhitekture na primeru IPTV.

Mentor:

prof. dr. Matjaž B. Jurič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Anton Zvonko Gazvoda, z vpisno številko **63090075**, sem avtor diplomskega dela z naslovom:

Integracijska arhitektura računalniškega oblaka, komunikacijskih in aplikacijskih storitev na primeru IPTV

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Matjaž Branko Jurič,
- 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 14. september 2012 Podpis avtorja:

Zahvalil bi se prof. dr. Matjažu Branku Juriču za pomoč in vodenje pri izdelavi diplomske naloge. Zahvaljujem se tudi staršema in bratoma za podporo med študijem.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Komunikacije kot storitev **3**

2.1 Združevanje komunikacijskih storitev 4

2.1.1 Protokol za vzpostavljanje seje 5

2.1.2 Kontekstualno osnovane aplikacije 5

2.1.3 SIP sporočanje 6

2.1.4 SIP infrastruktura 7

2.1.5 SIP URI naslovi 7

2.1.6 SIP interakcije 8

2.1.7 Vzorci SIP interakcij 9

2.1.8 SIP dogodki 9

2.1.9 Prisotnost 9

2.1.10 IP multimedijski podsistemi 12

2.2 CaaS referenčna arhitektura 12

2.2.1 Zahteve komunikacijskih rešitev 13

2.2.2 Konceptualna arhitektura CaaS rešitev 13

2.2.2.1 Aplikacija CaaS 15

2.2.2.1.1 Nivo vmesnika za upravljanje 15

2.2.2.1.2 Nivo orkestracije komunikacijskih storitev 16
2.2.2.1.3 Nivo komunikacijskih soritev 17
2.2.2.2 SIP storitve 17
2.2.2.3 Aplikacija za upravljanje 18
2.2.2.4 Varnostne storitve 18
2.2.2.5 Imeniške storitve 18
2.2.2.6 Poslovni podporni sistemi/Operacijski podporni sistemi 18
2.2.2.7 IMS infrastruktura 19
2.2.2.8 Oddaljena domena 19

3 Komunikacijske storitve z aplikacijam tipa SaaS 21
3.1 SaaS integracija 22
3.1.1 Komunikacijske storitve preko spletnih storitev 22
3.1.2 Prilagajanje storitev 23
3.1.3 Povezljivost 24
3.2 Pregled sistemov 26
3.3 Komunikacijske storitve preko spletnih storitev 27
3.4 Ogrodje SaaS adapter 28
3.4.1 Implementacija - BPEL ali implementacija po meri . . 29
3.4.2 Arhitektura ogradja SaaS adapter 30
3.5 Obhod požarnega zidu in NAT 33
3.5.1 Ključne ideje PASS 34
3.5.2 PASS agent 35
3.5.3 PASS strežnik 35
3.5.4 Potek izvajanja 36
3.5.5 Varnost 36

4 Aplikacija z uporabo CaaS: merjenje gledanosti vsebin v IPTV 37
4.1 Primer aplikacije za merjenje gledanosti vsebin IPTV 38
4.2 Pregled CaaS storitev v IPTV porazdeljenem sistemu 39

4.3 Arhitektura sistema za merjenje IPTV občinstva 41
4.3.1 Merjenje občinstva IPTV na osnovi STB sistema . . . 41
4.3.2 Omrežno osnovano merjenje statistik občinstva IPTV . 42
4.3.3 Primerjava shem za merjenje občinstva 43

5 Tehnologije uporabljene pri razvoju lastne aplikacije 45
5.1 Pregled uporabljenih tehnologij 45
5.1.1 JAX-WS 47
5.1.2 JAXB 48
5.1.3 JPA 50
5.1.4 Java Message Service - JMS 50
5.1.5 Sejna in sporočilna zrna 51

6 Načrtovanje in razvoj aplikacije 53
6.1 Funkcionalnosti 55
6.1.1 Scenarij “nakup” 55
6.1.2 Scenarij “glasuj” 56
6.1.3 Scenarij “daruj” 56
6.1.4 Scenarij “sprejmi nova naročila” 57
6.1.5 Scenarij “rezultati glasovanja” 57
6.1.6 Scenarij “seznam donacij” 57
6.2 Arhitektura sistema 58
6.2.1 Opis delovanja posameznih komponent 59
6.2.1.1 Strežnik za posredovanje zahtev 59
6.2.1.2 SaaS aplikacija 59
6.2.1.2.1 Sprejem zahtev 60
6.2.1.2.2 Dostavljanje zahtev strankam 61
6.3 Arhitektura prototipne aplikacije 62
6.3.1 Konceptualni model podatkovne baze 62
6.3.2 Arhitektura aplikacije 63
6.3.2.1 Sinhrona spletna storitev 63
6.3.2.2 Asinhrona spletna storitev 66

6.3.2.2.1 Primerjava izvajanja klica po arhitekturah 68
6.3.3 Aplikacija pošiljatelja zahtev - IPTV ponudnika 68
6.3.4 Aplikacija stranke za sprejem naročil/rezultatov glasov/obvestil o darovanju 70

7 Sklepne ugotovitve **73**

Povzetek

Računalništvo v oblaku ponuja nov pristop k ponujanju informacijskih storitev. Informacijske storitve se izvajajo v oblaku in so uporabnikom dostopne kot storitve na zahtevo. Model nudenja programskih aplikacij, ki se izvajajo v oblaku, imenujemo Software as a Service, krajše SaaS in v prevodu programska oprema kot storitev. Aplikacije in z njo povezani podatki gostujejo v oblaku, ki je lahko javen ali zaseben. SaaS aplikacije so dostopne preko omrežja. V primeru, da aplikacija gostuje v javnem oblaku do nje dostopamo preko interneta. Za dostop do SaaS aplikacije se tipično uporablja tanek odjemalec prek spletnega brskalnika.

Stranke iz razlogov, kot zaupanje v ponudnika, neznane lokacije podatkov, varnosti in izgube podatkov nekaterih svojih občutljivih aplikacij ne želijo seliti v oblak. Določen del aplikacij pa bo stranka vseeno imela v oblaku. Pri tem naletimo na problem soodvisnosti aplikacij med katerimi je potrebna integracija. V diplomi je opisano, kako poteka integracija oziroma, kako se omogoči komuniciranje med aplikacijam v oblaku in aplikacijam znotraj organizacij. Cilj diplomske naloge je prikazati način integracije SaaS aplikacije s komunikacijskimi storitvami IPTV (Internet Protocol television), ki so dostopne kot storitev preko uporabe CaaS (Communications as a service) modela in aplikacijami strank, ki se izvajajo na infrastrukturi znotraj organizacije. Gre za inovativni sistem, ki uporabnikom omogoča naročanje, glasovanje in opravljanje darovanja v humanitarne namene preko interaktivne televizije. Diplomsko delo predstavi enostaven prototip takega sistema.

Ključne besede: aplikacija tipa SaaS, integracija aplikacij, spletna storitev, internetna televizija, vmesnik

Abstract

Cloud computing offers new approach to offering information services. Information services are carried out in the cloud and they are available on-demand. Model of providing software applications in cloud is called Software as a Service. Applications and related data are hosted in cloud, which can be private or public. Saas Applications are accessible over the network. Applications in the public cloud is hosted on remote location in datacenter and are accessible ower Internet. Customers usually access to Saas Applications using a thin client over web browser.

Customers don't want to move their applications to the cloud due of reasons such as lack of trust in cloud provider, unknown data location, safety and data loss. Some of the customers will still have part of their applications in the cloud regardless of risk losing data or safety. The applications often depend on each other so integration is required. In the following sections is explained how integration is made and how communication between applications in the cloud and applications within organizations, so called on-premise applications, is enabled. Way to integrate Saas applications with IPTV communications services which are accessable as a services (CaaS) and customers on-premise applications, which are carried out on the infrastructure of the organization, is primary goal of this diploma thesis. We will introduce an innovative system that allows IPTV subscribers to send order of some advertised product, vote during reality show and purse donation to charity via interactive television. The thesis presents a simple prototype of such system.

Keywords: Software as a Service application, application integration, web service, Internet television, interface

Poglavje 1

Uvod

Koncept računalništva v oblaku sega v obdobje petdesetih let prejšnjega stoletja, ko so korporacije, ki so si lastile glavni računalnik (mainframe) ostalim korporacijam omogočile dostop do računalnika preko lahkih klientov. Ker je bil nakup glavnega računalnika zelo velik strošek je bilo pomembno najti način kako povrniti stroške. Tako so lastniki glavnih računalnikov ostalim korporacijam, ki si niso mogle privoščiti nakup svojega glavnega računalnika, omogočile dostop večim uporabnikom, ki so med sabo delili iste vire računalnika.

Danes računalništvo v oblaku predstavlja uporabo računalniških virov, strojno in programsko opremo, ki so dostavljeni kot storitev preko omrežja. Ime izhaja iz simbola oblaka, ki abstraktno prikazuje kompleksno infrastrukturo v sistemskih diagramih. Osnovni trije modeli računalništva so: Infrastructure as a service (IaaS) - infrastruktura kot storitev, Platfom as a Service (PaaS) - platforma kot storitev in Software as a Service (SaaS) - programska oprema kot storitev. Pri diplomskem delu smo se osredotočili na SaaS in Communications as a Service (CaaS) - komunikacijske storitve kot storitev, kratica ima sicer v literaturi tudi drug pomen, vendar jo bomo pri diplomskem delu vseeno uporabili v tem kontekstu. Model nudenja komunikacijskih storitev kot storitve (CaaS) smo v delu opisali podrobneje. SaaS aplikacije so na voljo v uporabo večim strankam, prednost aplikacij, ki gostujejo v oblaku

je elastičnost in razširljivost, poleg tega aplikacijo vzdržuje in nadgrajuje gostitelj, kar pomeni, da stranka za to ne potrebuje svojega osebja. Stranki se zaračuna samo uporaba programskih storitev v oblaku.

V okviru diplomske naloge smo se lotili načrtovanja arhitekture sistema, ki bi naročnikom internetne televizije ob gledanju televizijskega oglasa za produkt le tega naročil s pritiskom na gumb, ali pa ob gledanju resničnostne oddaje glasoval s pritiskom na gumb na daljinskem upravljalniku, namesto da bi moral opraviti telefonski klic. Izziv implementacije takega sistema je zelo obsežen. V mislim moramo imeti kako potem tako naročilo dostaviti prodajalcu produkta, kako dostaviti oddan glas pravi stranki. Tak sistem se bo integriral z že obstoječimi sistemi, na primer z infrastrukturo ponudnika internetne televizije, na drugi strani pa se bo povezoval s prodajalci oglaševanih produktov, katerim bo dostavil nova naročila in rezultate glasovanja. Pomembno je tudi to, da obstoječe sisteme čim manj obremenjujemo in ne vsiljujemo večjih nadgradenj teh sistemov.

Opisali bomo sistem, ki omogoča merjenje statistike gledanosti posameznih kanalov uporabnikov internetne televizije. Primer sistema je bil predstavljen v članku An innovative application over Communications-as-a-Service: network based multicast IPTV audience mesurement. Na osnovi tega primera smo se lotili načrtovanja svojega sistema. Razvili smo prototip SaaS aplikacije, ki povezuje naročnika internetne televizije in stranko, ki na primer prodaja produkt, in tako omogoči dostavo teh naročil. SaaS aplikacija ustreza večkratno-najemni arhitekturi, kar pomeni, da jo uporablja več naročnikov, ki si delijo vire. SaaS aplikacija je bila realizirana kot množica vmesnikov, ki so implementirani kot spletne storitve. Spletne storitve smo implementirali v Java EE platformi z uporabo JAX-WS (Java API for XML Web Services). V diplomskem delu smo predstavili dve podobni arhitekturi SaaS aplikacije in opisali prednosti ene pred drugo. Pri razvoju prototipa smo implementirali odjemalno aplikacijo, ki simulira pošiljanje zahtev za naročanje produkta, glasovanje in darovanje v humanitarne namene. Implementirali in razložili smo tudi delovanje odjemalne aplikacije prodajalca oglaševanega produkta.

Poglavje 2

Komunikacije kot storitev

Komunikacije kot storitev je model nudenja virov za podporo komunikacijskih storitev v podjetju. Takšne komunikacije lahko vključujejo VoIP (Voice over Internet Protocol), imenik strank, konferenčno klicanje in podobno. CaaS ponudnik je odgovoren za vso strojno in programsko opremo, katero upravlja sam in hrati nudi garantirano kvaliteto storitev (Quality of Service - QoS). CaaS podjetjem omogoča selektivno uvajanje storitev in aplikacij, ki jih plačujejo glede na uporabo (pay as you go).

Komuniciranje preko interneta ni nov pojav. Preko interneta že vrsto let pošiljamo video vsebine, zvok in podatke in to brezplačno oziroma za nizko ceno. Paradoksalno je bil potek sprejetja komunikacijskih storitev preko interneta relativno počasen. Večina podjetij in posameznikov komunicira preko omrežij, ki obstajajo že od odkritja telefona, kljub dolgotrajni zaskrbljenosti glede kvalitete storitev in zanesljivostjo teh komunikacijskih sistemov.

Splošno prepričanje je, da ko govorimo o IP komunikacijah vedno pomislimo na internet. IP se že vrsto let uporablja za komuniciranje v lokalnih in posebnih omrežjih. Primer komuniciranja preko IP so IP osnovani telefonski sistemi namenjeni telefoniji v podjetju. Primer IP komunikacij, ki potekajo preko interneta, so programski telefonski sistemi za osebne računalnike (software-based).

IP komunikacije so v poslovnih sistemih omogočene preko privatnih cen-

tral (Private Branch Exchange - PBX) z implementacijo signalizacijskih protokolov preko IP: H.323 ali Session Initiation Protocol (SIP). PBX sistemi, ki bazirajo na IP, podjetjem nudijo prednosti, kot vzpostavitev telefonskega sistema v podjetju na istem omrežju, ki se uporablja za prenos podatkov. Združenje glasovnega in podatkovnega prenosa po omrežju podjetja zmanjšuje stroške postavljanja ločenih omrežij za podatke in zvok. Ravno tako zmanjšajo stroške administracije, saj lahko uporabnik svoj telefon priklopi nekje drugje v omrežju brez pomoči administratorja. Slaba stran teh sistemov je, da so dragi, zahtevni za vzdrževanje in upravljanje; kar jih ustvari predrage za majhna in srednje velika podjetja.

Rezultat tega je zapostavljen trg manjših in srednjih podjetij, ki bi imela na voljo stroškovno učinkovito telefonijo. Zadnjih nekaj let ponudniki poslovnih komunikacij vlagajo v razvoj komunikacijskih storitev v gostujočem okolju. Aplikacije, ki nudijo komunikacijske storitve v gostujočih okoljih imenujemo CaaS. CaaS temelji na modelu SaaS z dodatki, ki so specifični za aplikacije za komuniciranje.

Razumevanje modela CaaS je pomembno pri razvoju naše aplikacije, ki bo uporabljala rešitev CaaS za dostop do storitev IPTV.

2.1 Združevanje komunikacijskih storitev

V zadnjem desetletju se v telekomunikacijski industriji pogosto pojavlja izraz združene komunikacijske storitve. Izraz opisuje sobivanje telefonskih, video in podatkovnih komunikacij znotraj istega omrežja.

Vse več storitev je na voljo na večih napravah, meja med programskimi aplikacijami in komunikacijskimi aplikacijami bledi. Tradicionalne lastnosti komunikacijskih aplikacij – kot dogodkovno vzpostavljena povezava med dvema končnima točkama (point-to-point konference, multicast), uporaba komunikacijskih protokolov bo postalo potrebno tudi pri poslovnih in razvedrilnih aplikacijah.

Te zahteve ustvarjajo paradigmo za arhitekte programskih rešitev. Za-

radi pomanjkanja ogrodij za razvoj storitev CaaS, ki se šele uveljavljajo, je za razvijalce takih storitev pomembno poznavanje omrežja in omrežnih protokolov za dostopanje do storitev CaaS in tudi protokolov znotraj same storitve CaaS.

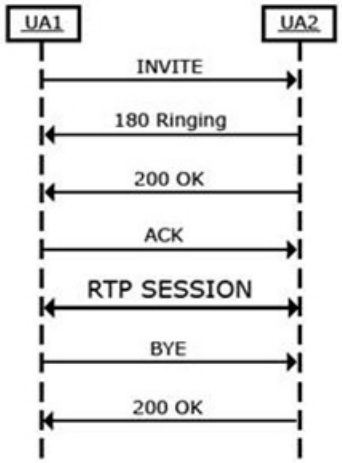
2.1.1 Protokol za vzpostavljanje seje

Protokol za vzpostavitev (Session Initiation Protocol - SIP) seje je bil uporabljen pri razvoju prvih storitev v CaaS prostoru. SIP je protokol aplikacijske plasti, ki omogoča aplikacijam (SIP aplikacijam), da se povežejo v SIP infrastrukturo ne glede na prenosni medij, ali je ta žični ali brezžični. SIP infrastruktura nudi okolje za kreiranje dogodkovno vodene aplikacije, ki se poveže s končnimi uporabniki na napravah ki jo uporabljajo. Aplikacije, ki uporabljajo SIP protokol lahko komunicirajo s končnim uporabnikom kadarkoli, ne glede na to katero napravo uporabljajo in preko različnih oblik medija (video, zvok in podatki.

2.1.2 Kontekstualno osnovane aplikacije

Prvotne implementacije SIP so bile osredotočene na komuniciranje znotraj korporacij, SIP aplikacije, ki se razvijajo so bolj podobne internetnim aplikacijam kot pa tradicionalni telefoniji, kar povzroča zmedo pri razvoju SIP aplikacij. Poznanih je veliko aplikacij, kot na primer pretakanje glavnih dogodkov športa na mobilni telefon ali pa SIP aplikacije, ki nudijo storitve v kriznih situacijah. Tu se skriva potencial SIP: ponuja nam možnost za razvoj dogodkovno proženih programskih rešitev, ki komunicirajo s končnim uporabnikom kjer koli (glede na napravo).

Obstaja neskončno scenarijev, ko ljudje želijo obvestilo o nekem dogodku ali pa želijo nekoga obvestiti o nekem dogodku. Predstavljajmo si, da sprejmemo obvestilo o težavah na cesti, ki vodi na delo hkrati pa prejmemo informacije o alternativni poti glede na našo trenutno lokacijo. Ali pa se registriramo v hotelu in nam televizijski sistem ob vklopu prikaže glavne dnevne



Slika 2.1: Vzpostavitev SIP seje [1]

dogodke v športu, ki nas zanimajo. Vse to lahko dosežemo s kontekstnimi aplikacijami z implementacijo protokola SIP.

2.1.3 SIP sporočanje

Kot že omenjeno, SIP je protokol aplikacijske plasti, ki vzpostavi sejo med dvema ali večimi uporabniškimi agenti (UA – User Agent)-končne točke s podporo SIP. SIP omogoča uporabniškemu agentu, da proži zahtevo za informacije ali pa zagotovi informacije drugemu uporabniškemu agentu. Uporabniški agent je lahko SIP telefon ali pa multimedijska naprava kot prenosnik ali pameten telefon.

SIP zahteva (SIP request) proži vzpostavitev dialoga med dvema uporabniškima agentoma. SIP odgovor (SIP response) vsebuje trimestno numerično kodo, kjer prva številka predstavlja razred odgovora (koda 2xx predstavlja uspešen odgovor – success response code). SIP transakcija (SIP transaction) zavzema vse od SIP zahteve do zadnjega SIP odgovora. Slika 2.1 predstavlja vzpostavitev (Real-Time Transport Protocol - RTP) seje znotraj SIP seje. Primer RTP seje je telefonski klic – VoIP.

SIP sporočila (SIP messages) so podobna sporočilom elektronske pošte -

z zahtevo na URI (Unified Resource Indicator) naslov in telesom sporočila (body) v MIME (Multipurpose Internet Mail Extensions). Glava SIP sporočila vsebuje informacije namenjene SIP aplikaciji. Telo SIP sporočila podpira več tipov vsebine, od golega teksta in HTML (HyperText Markup Language) do drugih kompleksnejših oblik informacijskih vebin. SIP za vzpostavitev dialoga uporablja SDP (Session Description Protocol), kjer ena stran ustvari SDP zahtevo, druga pa se odzove z odgovorom na to SDP zahtevo (SDP answer).

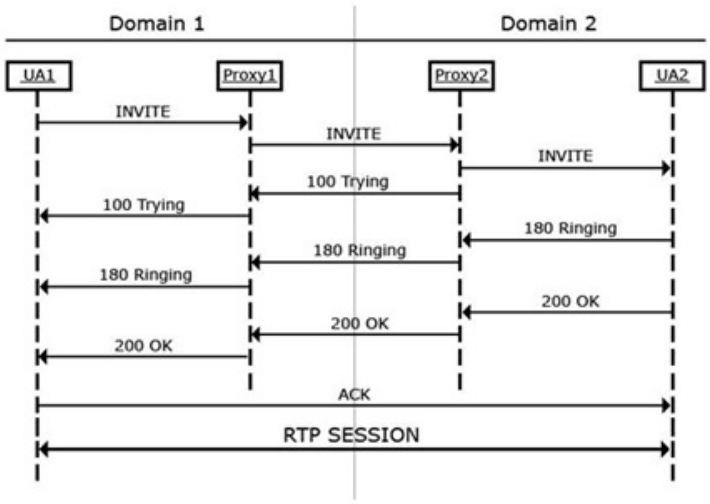
2.1.4 SIP infrastruktura

SIP sestavlja veliko obširnejša infrastruktura kot samo UA, ki vzpostavljajo sejo. Infrastruktura lahko vključuje SIP proxy strežnike, SIP preusmeritvene strežnike in SIP register strežnike. Vse te naprave so logične, kar pomeni, da niso ločeni fizični elementi.

SIP proxy strežnik sprejema SIP zahteve in jih posreduje naprej. SIP proxy strežnik se pogosto uporablja za NAT (Network Address Translation) ali pa kot požarni zid za nadzor dostopa. SIP preusmeritveni strežnik prejme SIP zahtevo in izvede poizvedbo, rezultat pa vrne UA, ki je poslal zahtevo. SIP register strežnik uporabniku omogoči povezavo z UA preko njegove identitete.

2.1.5 SIP URI naslovi

Vsak UA ima svoj URI (Uniform Resource Identifier) naslov preko katerega je lociran med SIP transakcijo. SIP interakcija lahko vsebuje več kot le en URI. Uporaba večih URI naslovov nudi podporo kontekstno osnovanim rešitvam tako, da loči uporabnika in UA h kateremu se pošiljajo SIP sporočila. Seznam virov je uporabljen za podporo skupinskim operacijam, na primer konference. Poznamo tri tipe SIP URI-jev: uporabniški URI, URI naprave in URI storitve. Uporabniški URI je identiteta uporabnika v komunikacijskih sistemih. Uporabniški URI je naslov zapisa (Address Of Record -



Slika 2.2: Vzpostavitev RTP seje [1]

AOR) končnega uporabnika. URI naprave predstavlja eno SIP UA instanco. Naprava je lahko začasno povezana z uporabnikom. V primeru zunanjega naslavljanja, SIP veže URI naprave z AOR. URI storitve predstavlja SIP interakcijo tipa mnogo v mnogo, na primer konferenčna seja.

2.1.6 SIP interakcije

Slika 2.2 prikazuje vzpostavitev RTP seje med dvema UA, ki se nahajata v različnih domenah. Prikazan je postopek vzpostavljanja povezave UA-ja s pošiljanjem INVITE zahteve drugemu UA-ju. INVITE zahteva vsebuje URI uporabnika, ki je kontaktiran z zahtevo. Proxy strežnika usmerjata zahteve med domenama in UA-ji. Ko se vzpostavi medijska seja, proxy strežnika nista več potrebna v interakciji. Katere naprave, poleg UA-jev, morajo sodelovati med medijsko sejo, se odloči arhitekt SIP rešitve.

2.1.7 Vzorci SIP interakcij

Na sliki 2.2 je prikazana standardna SIP interakcija v kateri sodelujeta dva UA in dva namestniška strežnika. Vzorec take interakcije je uporabljen pri VoIP komunikacijah (UA to UA). SIP poleg tega vzorca podpira še nekaj ostalih:vsak z vsakim, oddajanje sporočil večim uporabnikom (multicast) in konference.

SIP interakcije tipa vsak z vsakim ne vključujejo nobenih posrednikov. Z uporabo dinamičnega DNS lahko UA uporabi URI brez registracije s SIP strežnikom. Pošiljanje zahtev na URI naslove vseh razpoložljivih SIP registrov, da odgovorijo na zahtevo. Vzorec konferenc deluje tako, da vzpostavi interakcijo z vsemi sodelujočimi.

2.1.8 SIP dogodki

SIP dogodki (SIP events) omogočajo SIP komponentam, da se prijavijo na dogodke druge SIP naprave. Z uporabo naroči/obvesti (subscribe/notify) vzorca lahko SIP UA obvestimo, ko se spremeni status oddaljene aplikacije. Koncept dogodkovno vodenih aplikacij ni nov, vendar možnost, da usmerimo dogodke UA v primeru prisotnosti, je močan komunikacijski mehanizem.

2.1.9 Prisotnost

Prisotnost (presence) je kritični del kontekstnih aplikacij. Prisotnost je pomemben koncept, gre za “dial tone of the 21st century.” Prisotnost komunikacijam da kontekst. Storitve instantnega sporočanja (Instant messaging - IM) za usmerjanje sporočil uporabljajo prisotnost. Če ste na računalniku prijavljeni v storitev IM in se prijavite še na drugem računalnike z istim uporabniškim računom, bodo vaša sporočila usmerjena na računalnik, ki ga trenutno uporabljate glede na informacije o prisotnosti. Glavna funkcionalnost, ki je implementirana za podporo prisotnosti je znana kot »rendezvous« - točka srečanja, kar omogoča vzpostavitev zveze z nekom z uporabo AOR, brez poznavanja načina kako je povezan v omrežje.

Obstaja število vzorcev arhitektur sistemov, ki uporabljajo prisotnost. Arhitekturni vzorec vsak z vsakim konceptu prisotnosti omogoča UA napravam neposredno naročanje na UA iz katerega bodo prejemali dogodke. Alternativni arhitekturni vzorec vsak z vsakim arhitekturnega vzorca lahko implementiramo tako, da uporabimo prisotnostni strežnik kot posrednik, ki skrbi za vsa naročila, ki prispejo iz UA naprav.

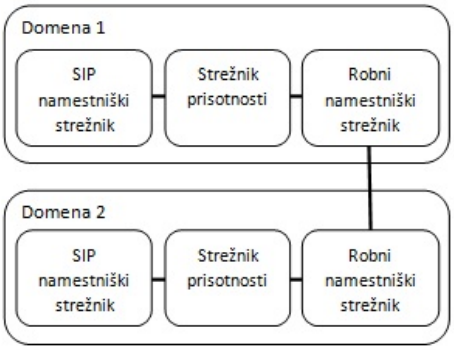
Pogost vzorec za načrtovanje sistemov osnovanih na protokolu SIP je Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) arhitekturni vzorec. Ta vzorec temelji na odprtih standardih in vsebuje število vlog med različnimi elementi procesiranja. Z arhitekturo posrednika SIP vsebuje centraliziran strežnik prisotnosti (Presence Server -PS), ki nadzoruje vse naročnike. Druga komponenta SIMPLE arhitekture je "presentity" oz. prisotna entiteta. "Presentity" je oseba, ki ima opisljivo prisotnost npr. "zaseden" znotraj IM programa. Arhitektura vsebuje tudi entiteto opazovalec (watcher), ki predstavlja končno točko, ki se prijavi na spremembe podatka o prisotnosti. Še eno vlogo predstavlja prisotnostni agent (Presence Agent -PA), ki obvešča gledalca (watcher) o spremembah prisotnosti vira. Zadnjo vlogo pa predstavlja uporabniški agent prisotnosti (Presence User Agent - PUA), program ki ima informacije o trenutnem stanju osebe.

Diagram 2.3 prikazuje vse programske elemente znotraj SIMPLE arhitekture in opisuje njihove medsebojne interakcije.

Da je objavljanje informacij o prisotnosti med ločenimi domenami mogoče je arhitekturi dodan robni namestniški strežnik (edge-proxy). V tem scenariju se prisotnostni strežnik znotraj domene obnaša kot prisotnostni agent uporabnikom druge domene in je obveščeni o vseh spremembah stanja uporabnikov.

Ko ima uporabnik možnost, da se poveže z večimi uporabniškimi agenti, vedno obstaja tudi možnost konflikta. Ob objavljanju informacij o stanju velja pravilo, da je zadnji UA, ki objavi stanje AOR. S katerimi komponentami omogočimo med domensko objavljanje informacij o prisotnosti je razvidno

Slika 2.3: Vloge SIMPLE athitekturnega vzorca [1]



Slika 2.4: Med domenskego objavljanje informacij o prisotnosti [1]

na sliki 2.4.

2.1.10 IP multimedijski podsistemi

Skupaj z razumevanjem SIP, načrtovanje rešitev CaaS zahteva tudi razumevanje konteksta omrežja v katerem bo rešitev delovala. V prejšnjem delu sta bila predstavljena dva scenarija: domenski (intra-domain) in med domenski (cross-domain) scenarij. V prejšnjem poglavju nismo zvedeli ničesar o tem, kako CaaS razširiti v mobilna in PSTN omrežja. IP multimedijski podsistem (IP Multimedia Subsystem - IMS) je del omrežja, ki lahko vpliva na to, kako so načrtovane komunikacijske rešitve. IMS je element omrežij tretje generacije (3G), ki olajšuje združevanje interneta in mobilnih omrežij.

Ker brezžična omrežja že nudijo oblike internetnih komunikacij – kot pošiljanje tekstovnih sporočil in internetno brskanje se je smiselno vprašati o potrebi IMS. IMS dodaja vrednost mobilnim omrežjem na dveh ključnih področjih:

- Možno je zagotoviti kvaliteto storitev v 3G paketnih omrežjih, ki izboljša kvaliteto medijskih storitev.
- Storitev lahko integriramo v omrežno okolje, kar je omogočeno s standardnimi vmesniki IMS, ki so na voljo razvijalcem.

Ti standardni vmesniki ustvarjajo “plug and play” okolje za komunikacijske storitve kot na primer (voicemail) storitev glasovna pošta.

Povejmo le, da je razširitev CaaS v mobilna in PSTN omrežja možno preko IMS, v podrobnosti, dostopne na [1], se v tem diplomskem delu ne bomo spuščali.

2.2 CaaS referenčna arhitektura

Načrtovanje aplikacij večkratno-najemne (multi-tenant) arhitekture, ki so namenjene večim odjemalcem in so v interakciji z komunikacijskimi omrežji, kar predstavlja CaaS, je zelo kompleksna naloga.

2.2.1 Zahteve komunikacijskih rešitev

Ključna razlika med komunikacijskimi aplikacijami in standardnim poslovnim aplikacijami je visoka prioriteta nefunkcionalnih zahtev kot 99.999% dostopnost (availability), visoke performance (kratek odzivni čas), zanesljiva varnost, vmesniki za nadzor uporabe storitev in možnost konfiguriranja in administriranja storitev.

Še ena lastnost komunikacijskih aplikacij je, da so misijsko kritične in njihovo vseprisotno uvajanje znotraj organizacije. Te lastnosti povzročijo skorajšnji nemogoč nadzor v okolju, kjer bo rešitev uvedena. To ustvari odvisnosti, ki so edinstvene pri komunikacijskih aplikacijah, na primer: zahteva po interakciji z obstoječim PBX sistemom zaradi interoperabilnosti z namiznim telefonom ali pa odvisnost od omrežja v katerem bo rešitev uvedena.

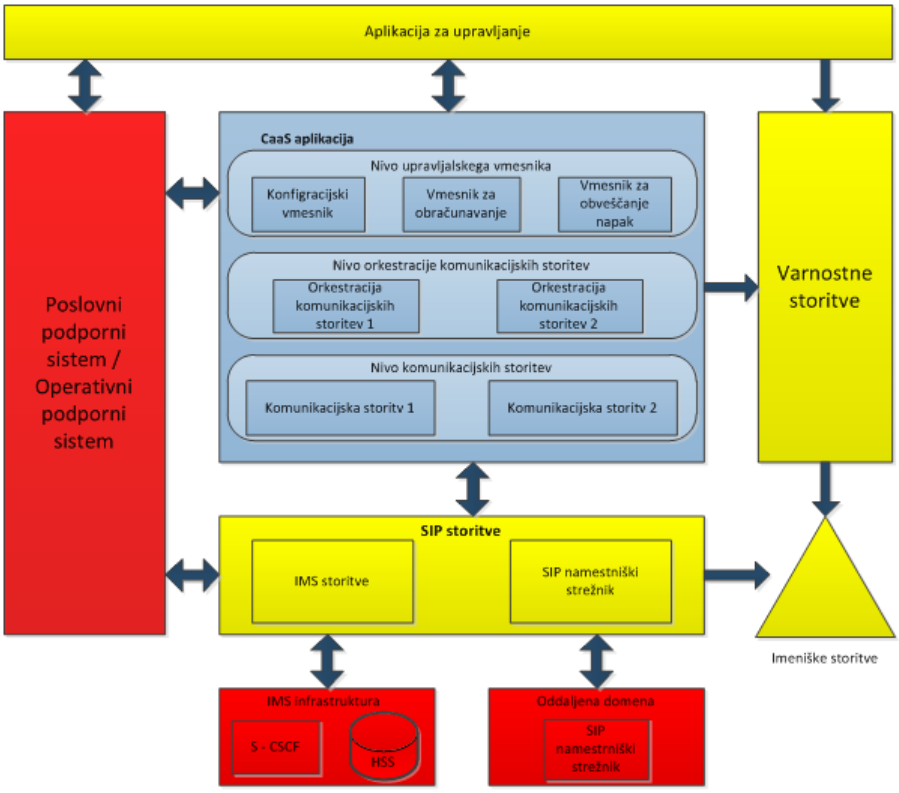
Naslednja spremenljivka je neločljivo povezana z gostitelji rešitve, vključno s CaaS aplikacijami, je dostopnost podpornih sistemov za operacije (Operations Support System - OSS) znotraj okolja kamor bo rešitev uvedena. Vsaka uvedba rešitve bo skoraj gotovo namenjena različnemu OSS produktu. Načrtovanje rešitev v tako okolje je dinamično, odločitve ki jih moramo sprejeti o protokolih in vzorcih interakcij, ki bodo uporabljeni za integracijo rešitve v gostujoče okolje.

2.2.2 Konceptualna arhitektura CaaS rešitev

Diagram na sliki 2.5 vsebuje konceptualno arhitekturo CaaS rešitve. Posamezne komponente diagrama so predstavljene z različnimi barvami:

- Aplikacija CaaS (modra).
- Dodatne komponente, ki jih proizvedejo načrtovalci rešitev CaaS, če je to potrebno (rumena).
- Komponente, ki se nahajajo v okolju, kjer bo storitev nudena (rdeča).

Kot je bilo navedeno prej bo okolje, v katerem se bo rešitev nahajala, vplivalo na zahteve po komponentah, ki niso del aplikacije CaaS.



Slika 2.5: Referenčna arhitektura CaaS [1]

Pomembno je omeniti, da je konceptualna arhitektura visokonivojska, vsaka komponenta se razdeli v podkomponente ki podpirajo specifične storitve, ki izpolnjujejo zahteve rešitve. Konceptualni diagram ne predstavlja fizične implementacije.

2.2.2.1 Aplikacija CaaS

CaaS aplikacija predstavlja množico storitev, ki izpolnjujejo komunikacijske funkcije kot na primer IP telefonija. Ker množica teh storitev vsebuje poslovno vrednost rešitve CaaS, se največ truda za razvoj vlaga v ta del arhitekture. Aplikacija CaaS je implementirana na več nivojski arhitekturi. Več nivojska arhitektura povzema različne dele aplikacije na večih nivojih, kar omogoča uporabo nivojev v različnih kontekstih, npr. nivo komunikacijskih storitev je lahko prilagodljiv posameznim zahtevam različnih strank. Na konceptualnem nivoju CaaS aplikacijo sestavljajo trije nivoji:

- Nivo vmesnika za upravljanje (Management interface layer)
- Nivo orkestracije komunikacijskih storitev (Communications service orchestration layer) – avtomatizirano ureditev, usklajevanje in upravljanje)
- Nivo komunikacijskih storitev (Communications service layer)

Delitev na nivoje je povsem na logičnem nivoju, različne komponente se lahko izvajajo v istem procesu kot druge komponente.

2.2.2.1.1 Nivo vmesnika za upravljanje

Vsaka komunikacijska aplikacija zahteva možnost, da obstaja izvajalec (ponudnik storitve, podjetje ali ponudnik okolja), ki lahko izvaja naloge nad naslednjim funkcijam: napakam, konfiguracija, vodenje uporabniških računov, performance in varnost (fault, configuration, accounting, performance, security - FCAPS). Izvajanje nalog omogoča vmesnik za upravljanje.

Komponenta	**Opis**
Konfiguracijski vmesnik	Vmesniki je namenjen konfiguraciji strojne in programske opreme, ki podpirajo CaaS aplikacijo. Vmesnik omogoča konfiguracijo fizičnega okolja, kot konfiguracija strojne in omrežne opreme ter konfiguracija logičnega okolja – storitve, ki tečejo v okolju komunikacijskih rešitev.
Vmesnik za obračunavanje	Vmesnik ponudniku storitev zagotavlja informacije o zaračunavanju. Vmesnik je lahko dogodkovno voden, nudi podatkovni tok (data stream) ali pa je klican periodično.
Vmesnik za obveščanje o napakah	Ta vmesnik zagotavlja informacije o napakah in performancah. Glede na vsebino se informacija lahko posreduje kot dogodek (trap SNMP) ali pa je klican (pulled) od uporabnika podatkov.

Tabela 2.1: Komponente nivoja vmesnika za upravljanje

Različne funkcije komponent z zunanjimi aplikacijami komunicirajo na drugačen način. Nekatere vmesnike lahko implementiramo po (request/response) vzorcu zahteva/odgovor izmenjave sporočil (Message Exchange Pattern - MEP), ki so na voljo kot spletne storitve, drugi vmesniki pa lahko objavljajo dogodke preko protokola SNMP (Simple Network Management Protocol) z uporabo SNMP pasti.

Tebela 2.1 opisuje komponente vsebovane na nivoju vmesnika za upravljanje CaaS aplikacije.

2.2.2.1.2 Nivo orkestracije komunikacijskih storitev

Na tem nivoju se izvaja orkestriranje nad komunikacijskimi storitvami. Nivo zagotavlja stopnjo agilnosti CaaS rešitve tako, da dovoli prilagoditev (customization) storitev glede na zahteve uporabnika. Primer orkestracije je podan v tabeli 2.2.

Komponenta	Opis
Orkestracija komunikacijskih storitev	Orkestracija komunikacij združuje komunikacijske storitve s čimer omogoči komunikacijske operacije. Na primer Automatic Call Distribution (ACD) aplikacija ima lahko specifične zahteve za usmerjanje glede na vneseno vrednost iz tipkovnice – »Pritisni 0 za operaterja.« Te zahteve usmerjanja so lahko implementirane kot orkestracija: (1) sprejmi klic, (2) poziv za vnos, (3) procesiraj vhod in (4) usmeri klic pravi destinaciji.

Tabela 2.2: Komponenta orkestracijskega nivoja

Komponenta	Opis
Komunikacijska storitev	Storitve, ki izvajajo množico komunikacijskih opravil – usmerjanje klica ali pa glasovna pošta (voicemail).

Tabela 2.3: Komponenta nivoja komunikacijskih storitev

2.2.2.1.3 Nivo komunikacijskih soritev

Nivo predstavlja storitve, ki izvajajo naloge komuniciranja, na primer – logika funkcije usmerjanja »forward call« posreduj klic. Storitve so lahko implementirane kot razredne knjižnice, lahko pa so kar celoten komunikacijski sistem kot PBX ali pa sistem za glasovno pošto.

2.2.2.2 SIP storitve

SIP storitve CaaS rešitvam ponujajo možnost komuniciranja preko SIP protokola. Te storitve razumejo zmožnosti SIP, kot je kako usmeriti klic glede na informacije o prisotnosti, ali pa kako spremeniti stanje uporabnika na podlagi SIP dogodkov. Te storitve vsebujejo tudi IMS storitve, ki izmenjujejo informacije z HSS (Home Subscriber Server) in S-CSCF (Serving - Call Session Control Function).

2.2.2.3 Aplikacija za upravljanje

Aplikacija za upravljanje s CaaS ni nujno potrebna. Ali je upravljanje potrebno je odvisno od potreb ponudnika storitev. Aplikacija za upravljanje lahko dopolnjuje CaaS aplikacijo, tako da ponudi dodatne storitve, ki so tipično vsebovane v poslovnih podpornih sistemih (BSS – business support systems) in podpornih sistemih operacij (OSS – Operations support systems). Te storitve lahko vključujejo upravljanje fizičnega okolja (strežnike, procese in omrežja) kot tudi odzivanje in izvajanje akcij ob napakah, preslikovanje logičnih modelov (stranke in storitev) v fizično okolje – storitev X teče na strežniku 1234 ter nudenje informacij o zaračunavanju.

2.2.2.4 Varnostne storitve

Varnostne storitve CaaS so tudi lahko del obstoječega okolja ponudnika storitev. V CaaS okolju obstaja potreba po skupni varnosti na lokaciji uporabnika in ponudnika storitev. Če združen varnostni sistem ni del okolja ponudnika storitev je naloga proizvajalca CaaS aplikacije, da za varnostne storitve poskrbi on.

2.2.2.5 Imeniške storitve

Podobno kot pri varnostnih storitvah so ravno tako potrebne skupne imeniške storitve. Če teh storitev ne nudi ponudnik storitve je naloga proizvajalca CaaS aplikacije, da podpre tudi te storitve.

2.2.2.6 Poslovni podporni sistemi/Operacijski podporni sistemi

BSS/OSS so visoko zmogljivi (carrier-grade) sistemi, ki avtomatizirajo poslovne procese ponudnika storitev. Naloge teh sistemov, med drugimi, so: zaračunavanje, oskrbovanje (provisioning) in upravljanje napak. BSS/OSS sistemi so skoraj vedno del okolja ponudnika storitev, zato imajo načrtovalci CaaS rešitev zelo malo vpliva na te sisteme. CaaS načrtovalci morajo zagotoviti vmesnike za interakcijo s sistemi BSS in OSS.

2.2.2.7 IMS infrastruktura

IMS infrastrukturo priskrbi ponudnik komunikacijskih storitev. Tako kot pri omrežni infrastrukturi pri spletnih straneh, CaaS načrtovalci nimajo nikakršnega vpliva na opremo, ki se prodaja, tako se morajo držati standardov, ki so del IMS specifikacije.

2.2.2.8 Oddaljena domena

Oddaljena domena v konceptualni arhitekturi prikazuje E2E (Enterprise to enterprise) SIP komunikacijo. V začetku se bodo CaaS aplikacije omejile na komunikacije znotraj omrežja ali pa E2E komunikacije osnovane na SIP storitvah enega ponudnika. Ko pa se bodo internetne komunikacije bolj razvile pa bo odvisnost od enega ponudnika začela pojenjati.

Povzeto po [1].

Poglavje 3

Komunikacijske storitve z aplikacijam tipa SaaS

Programska oprema kot storitev (Software as a Service - SaaS) je model dostave aplikacij. SaaS se od običajnega dostavljanja aplikacij razlikuje po tem, da aplikacije gostujejo v omrežju ponudnika storitev in se dostavljajo kot spletne aplikacije. Aplikacije so dostopne kot storitve, ki so izpostavljene preko abstraktnih vmesnikov kot npr. spletne storitve. Spletne storitve omogočajo dostavo programskih aplikacij na zahtevo preko internetnega omrežja.

SaaS model dostave aplikacij na zahtevo na zahteva uvajanja velikih infrastrukturnih sprememb na strani stranke, kar močno zmanjša vnaprejšnjo zavezanost virom. Poleg tega so SaaS rešitve dostavljive v zelo kratkem času in z minimalnim naporom, kar jih naredi zelo zanimive za podjetja. SaaS je enojna, večkratno-najemna arhitektura, ki večim uporabnikom omogoča maksimalno deljenje virov brez medsebojnega motenja. Tak pristop omogoča uporabnikom transparentno dostavljanje popravkov in nadgradenj.

S hitrim sprejemanjem SaaS hkrati narašča tudi potreba po integriranju lastnih aplikacij podjetij s SaaS aplikacijami. Če hočemo, da so SaaS rešitve uporabne morajo integrirati druge storitve in aplikacije. Te storitve največkrat nudijo lastni sistemi podjetij. Tu imamo zakonske in varnostne omejitve, občutljivi podatki in aplikacije so lahko hranjene samo znotraj

podjetja, morajo pa biti dostopni SaaS aplikacijam, ko jih te potrebujejo. Poslovni procesi so lahko zelo kompleksni, zato so podprti z večimi aplikacijami in storitvami, kar zahteva integracijo lastnih aplikacij podjetij s SaaS aplikacijam.

Komunikacijske storitve (telefonija, video konference, prisotnost, obvestilne in sporočilne storitve) so ključni element poslovnih procesov. Komunikacijske storitve najpogosteje bivajo znotraj poslovnega omrežja. SaaS aplikacije lahko avtomatizirajo komunikacijske procese, zagotovijo učinkovito ravnanje s priložnostmi preko sodelovanja ter nadzor poslovnih interakcij, če so integrirane s komunikacijskimi storitvami. To lahko poveča produktivnost in učinkovitost podjetij.

V praktičnem delu bomo razvili aplikacijo tipa SaaS, ki bo namenjena povezovanju IPTV sistema, ki posreduje zahteve uporabnikov, z aplikacijam strank, ki sprejemajo zahteve uporabnikov IPTV. V tem delu bomo povzeli članek [2].

3.1 SaaS integracija

SaaS platforma zajema storitveno usmerjeno arhitekturo (Service Oriented Architecture - SOA) in spletne storitve (Web Services). Spletne storitve se uporabljajo kot standarden mehanizem za integracijo storitev. Spletne storitve za komuniciranje z drugimi SaaS platformami in svojimi odjemalci uporabljajo SOAP ovojnico. Funkcionalnosti in podatki, ki jih ponuja SaaS platforma, so izpostavljene preko vmesnikov spletnih storitev, kar omogoča odjemalcem programsko poizvedovanje oziroma posodabljanje aplikacijskih podatkov. Podobno kot SaaS aplikacije nudijo svoje storitve preko spletnih vmesnikov jih nudijo tudi aplikacije znotraj podjetij.

3.1.1 Komunikacijske storitve preko spletnih storitev

Če hočemo, da SaaS aplikacije uporabljajo komunikacijske storitve, morajo biti tudi te dostopne preko vmesnikov spletnih storitev. Komunikacijske sto-

ritve, ki so dostopne preko vmesnikov spletnih storitev smo že spoznali - CaaS. Komunikacijske storitve so tipično kompleksne storitve, ki hranijo stanje - storitve s stanjem (stateful). Pri izpostavljanju komunikacijskih storitev preko vmesnikov spletnih storitev, moramo upoštevati nekaj dejstev.

Komunikacijske storitve so dvostranske. Končne točke storitev (endpoint) lahko poleg strežnika nudi tudi odjemalec, tako lahko strežnik sproži zahtevo ali pa pošlje dogodek storitveni končni točki, ki je prvotno prožila zahtevo za storitev (odjemalec). Poleg tega so komunikacijske storitve asinhrone. To pomeni, da lahko končna točka, ki je poslala zahtevo za izvedbo storitve dobi samo potrditev ponudnika storitev o prejetju zahteve, rezultat pa se pošlje kasneje, ko se procesiranje zahteve konča.

Obveščanje z dogodki je še ena pogosta in kritična funkcija komunikacijskih storitev. Primer so telefonske storitve, ko telefon od strežnika (PBX) prejme dogodek "zvonjenje", brez da bi odjemalec (telefon) poslal zahtevo. V tem primeru morajo odjemalčeve aplikacije poslušati dogodke iz strežnika.

Komunikacijske storitve hranijo stanje (stateful) in so hkrati sejne (session-aware). Interakcija med odjemalcem in strežnikom poteka v zaporedju, interakcija ima pomen samo znotraj konteksta seje. Ta dejstva moramo upoštevati pri nudenju vmesnika spletne storitve komunikacijskim storitvam.

3.1.2 Prilagajanje storitev

Integracija aplikacij tipa SaaS z aplikacijam, ki so uvedene na infrastrukturi znotraj podjetij je redka zaradi sledečih razlogov.

Kljub temu, da poznamo veliko standardov spletnih storitev, ponudniki podpirajo različne množice standardov. Tudi v primeru uporabe istih standardov ponudniki nudijo različne verzije in svoje razširitve. WS-I Basic Profile nudi smernice za doseganje interoperabilnosti spletnih storitev, vendar to ne reši težav same interoperabilnosti. Ta problem integracije se največkrat rešuje z uporabo adapterja. Na primer CSTA Phase III [5] uporablja WS-Eventing in WS-BaseNotification za obveščanje o dogodkih, medtem ko SaaS platforme uporabljajo dosti bolj poenostavljene dogodkovne storitve.

Komunikacijske storitve pogosto vključujejo izvajanje večih operacij v določenem zaporedju. V takem primeru je bolje uporabiti adapter storitev, ki omogoči kompozicijo in orkestracijo storitev. Ta način zmanjša število interakcij med SaaS in aplikacijo podjetja, kar omogoči, da se SaaS rešitev osredotoči na poslovno logiko in ne na podrobnosti komuniciranja. Na primer konferenčna storitev, za začetek konference se mora odjemalčeva aplikacija najprej prijaviti na konferenčni strežnik, pridobiti ID seje, nato pa lahko pokliče operacijo, ki bo začela konferenco s specifičnim sejnim ID-jem. Če hoče odjemalčeva aplikacija nadzirati status konference in njenih udeležencev se mora prijaviti tudi na prejemanje dogodkov, ki aplikacijo zanimajo. V tem primeru bo adapter storitev izvedel vse manjše operacije in tako omogočil dostavo dogodkov med vmesnikoma. To storitvi omogoči, da ovije preslikovanje sporočil in pretvarjanje ostalih podrobnosti v obliko primerno za prenos, na drugi strani pa ponudi visokonivojsko storitev namenjeno SaaS aplikacijam.

3.1.3 Povezljivost

Komunikacijske storitve običajno prebivajo znotraj poslovnega omrežja, kar pomeni, da so zaščitene z požarnim zidom, poleg tega uporabljajo še NAT (Network Address Translation) . SaaS aplikacije pa se tipično nahajajo v oblaku nekega ponudnika. Za omogočitev integracije mora podjetje oz. organizacija SaaS aplikacijam dovoliti dostop do njihovih storitev preko NAT in požarnih zidov. NAT in požarni zid organizacije vpliva na integracijo v dveh vidikih. Lokacija (Uniform resource locator - URL) aplikacije podjetja je veljaven le znotraj omrežja podjetja, naslova ni mogoče usmerjati v javni domeni. Drugi vidik se nanaša na tipične nastavitve požarnega zidu, ta v navadi dovoljuje le promet, ki je namenjen v zunanje omrežje, blokira pa ves promet, ki je namenjen v notranje omrežje. Posledično SaaS aplikacije ne more poslati zahtevo spletni storitvi, ker jo prestreže požarni zid.

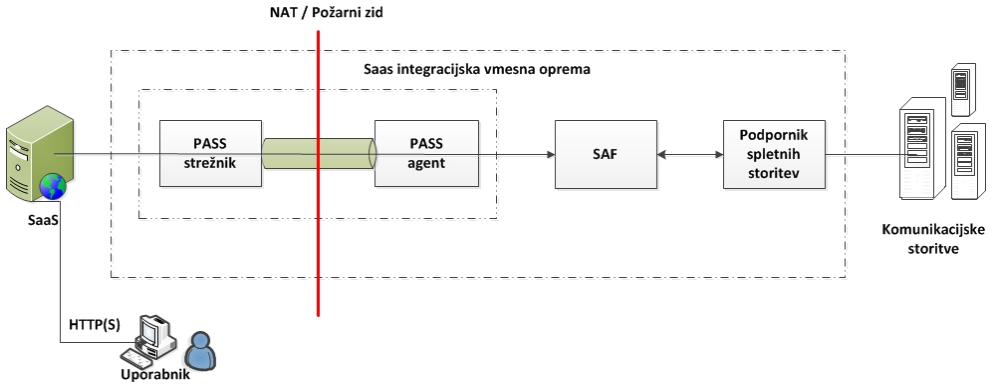
Da aplikacije podjetij postanejo dostopne je tako potrebno izvesti spremembe na požarnem zidu, ki bo potem dovolil promet SaaS aplikacijam, ki pride od določenega gostitelja, ali pa uvesti povratni namestniški strežnik

znotraj DMZ, ki bo izvajal usmerjanje prometa k notranjim aplikacijam. Ker so lahko arhitekture varnosti znotraj podjetja zelo napredne, so odobritveni procesi za take spremembe dolgotrajni in obsežni, kar ovira dostavo SaaS. Poleg tega, da se s spreminjanjem požarnega zida lahko pojavijo nova varnostna tveganja, hkrati postane težja podpora SaaS storitvenemu modelu (service delivery model). Zagotavljanje dinamičnosti SaaS, kjer lahko dodajamo in odstranjujemo storitve na zahtevo, je postalo ozko grlo.

VPN (Virtual Private Network) predstavlja še eno možnost povezovanja organizacijskih domen, ki pa ni ustrezna za SaaS. SaaS je večkatno-najemna platforma namenjena večim odjemalcem, ki si delijo isto omrežno infrastrukturo, isto strojno opremo in celo isto izvajalno okolje aplikacije. Z uporabo VPN postanejo aplikacije organizacij dostopne vsem aplikacijam, ki tečejo na SaaS platformi, dostopne se vse aplikacije, tudi tiste za katere to ne želimo.

Alternativni pristop, ki se uporablja pri večih SaaS, je zamenjava integracijskega vzorca s katerim obidemo direktno dostopanje do podatkov in storitev organizacije. Namesto tega aplikacije organizacij pošiljajo (push) podatke aplikacijam tipa SaaS v intervalih oz. ko se podatki spremenijo. V tem primeru vsa sporočila izvirajo znotraj požarnega zidu, kar pomeni, da lahko potujejo navzven. Ta pristop ni primeren za komunikacijske storitve, ker SaaS te storitve pričakuje v realnem času. Poleg tega ta način, objavljanje podatkov (push), ni tako razširljivo (scalable), ob pogostem spreminjanju podatkov, sploh če se prenašajo obsežni podatki, obremenjuje omrežje in prejemnika.

Pristop pošiljanja (push) podatkov aplikaciji SaaS smo uporabili pri razvoju lastne aplikacije, ki sprejema zahteve uporabnikov. Kot je opisano v članku [2] ta način obremenjuje sprejemnika, vendar je to potrebno zaradi narave naše aplikacije - zahteve uproabnikov moramo čim hitreje dostaviti strankam.



Slika 3.1: SaaS integracijska vmesna oprema [2]

3.2 Pregled sistemov

Pri integraciji moramo rešiti tri glavne probleme. Potreben je mehanizem za izpostavitev komunikacijskih storitev kot spletne storitve (CaaS). Potrebujemo tudi ogrodje za prevedbo in sestavljanje storitev. Tretja težava pa se nanaša na požarni zid, ko se aplikacija neke organizacije nahaja znotraj požarnega zidu, potrebujemo način, ki bo omogočil SaaS aplikaciji, da komunicira z aplikacijo organizacije brez spreminjanja konfiguracije požarnega zidu.

Odgovor na zgornje težave nam poda SaaS integracijska vmesna oprema (SaaS Integration Middleware). Vmesna oprema, prikazana na spliki 3.1, omogoči SaaS aplikacijam, da dostopajo do komunikacijskih storitev preko dvosmernih spletnih storitev in obratno. Sestavljajo ga tri glavne komponente:

1. Podpornik spletnih storitev (Web Service Enabler): Komponenta, ki ponudi vmesnike spletne storitve za komunikacijske storitve. Web Service Enabler pretvori storitvene klice v primerne API klice in obratno, tistim komunikacijskim storitvam, ki nimajo vmesnika spletne storitve.

2. Ogrodje SaaS adapter (SaaS Adaptor Framework - SAF): Je posrednik za premostitev SaaS storitev in storitev znotraj organizacije. Glavna

funkcionalnost te komponente je preslikovanje in pretvarjanje podatkov v obliko za prenos (marshalling) ter pretvarjanje in sestavljanje storitev. Nudi tudi mehanizem za razvoj in dostavo novih storitvenih adapterjev.

3. Dvosmerni prehod za spletne storitev (Two-way Web Service Gateway): Poseben prehod za SaaS aplikacije na katerega NAT in požarni zid ne vplivata. Sestavljen je iz dveh pod komponent – PASS strežnika in PASS agenta (Proxy-based firewall/NAT traversal middleware).

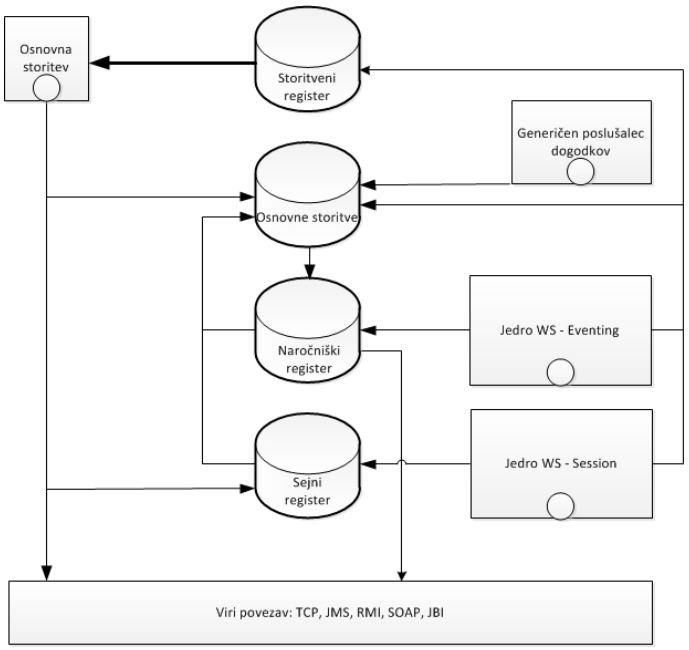
Uporabniki lahko dostopajo do komunikacijskih storitev preko SaaS aplikacij s proženjem spletne storitve na SAF. PASS agent in PASS strežnik omogočata SOAP zahtevam, da obidejo požarni zid preko posebnega varnega kanala. SAF potem proži eno ali več komunikacijskih storitev v imenu SaaS aplikacije, kateri nato pošlje odgovor.

3.3 Komunikacijske storitve preko spletnih storitev

Podpornik spletnih storitev (Web Service Enabler) razširja enosmerne zahteva/odgovor interakcije spletnih storitev v dvosmerne, tako imenovane full duplex interakcije. To omogoča izpostavitev komunikacijskih storitev kot spletne storitve.

Ker večinoma vse telekomunikacijske storitve hranijo stanje (so stateful) in zahtevajo kontekst seje. WS Enabler implementira množico centralnih spletnih storitev. Centralne spletne storitve so generične storitve, ki jih lahko uporablja več aplikacij in storitev, to so – WS-Session, WS-Security, WS-Eventing, itd. Centralne spletne storitve so sentitizirane v sestavi z aplikacijsko specifičnimi osnovnimi storitvami za izvedbo storitvene transakcije. Komponente in njihove medsebojne povezave so prikazane na sliki 3.2.

Osnovna storitev (Base service) komponenta predstavlja katerokoli komponento, ki implementira komunikacijsko spletno storitev in uporablja funkcionalnosti jedrnih spletnih storitev. Te storitve so vpeljane v podpornika



Slika 3.2: Arhitektura podpornika spletnih storitev

spletnih storitev (WS Enabler) in so integrabilne z ostalimi vpeljanimi storitvami. Krogi znotraj storitvenih komponent predstavljajo razširitvene točke, ki jih nudijo spletne storitve. Stanja storitev se hranijo v ustreznih podatkovnih bazah (registrih), tako storitve postanejo zavedne o stanju (stateless).

3.4 Ogrodje SaaS adapter

Ogrodje SaaS adapter (SaaS Adaptor Framework -SAF) je vmesni nivo za povezovanje (middleware) SaaS aplikacij in komunikacijskih storitev. Iz zornega kota SaaS aplikacije SAF predstavlja ponudnika komunikacijskih storitev , ki zaključi klice spletne storitve SaaS aplikacij. Iz perspektive komunikacijskih storitev je SAF odjemalec storitev, ki pretvarja zahteve SaaS aplikacije v množico novih zahtev namenjenih storitvam v ozadju (back-end). SAF nudi mehanizem vtičnika za obdelavo večih storitev. To razvijalcem omogoča enostaven razvoj in vpeljavo novih storitvenih adapterjev za nove storitve.

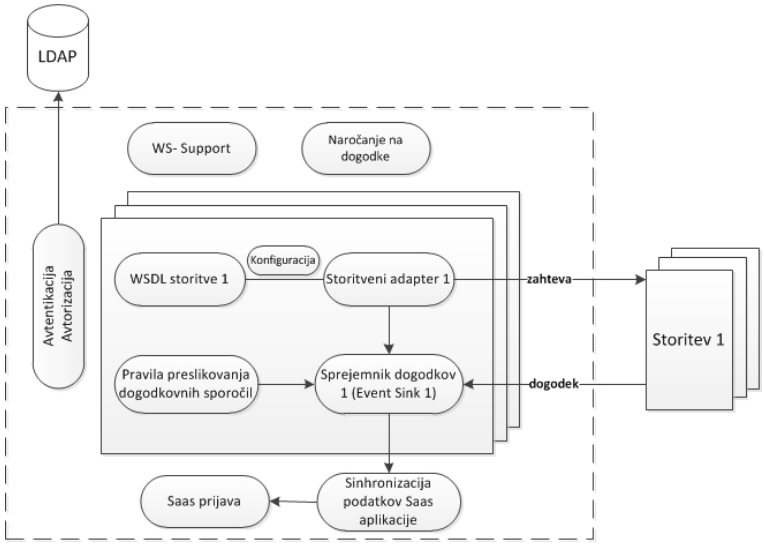
3.4.1 Implementacija - BPEL ali implementacija po meri

Business Process Execution Language (BPEL) je standardni jezik za modeliranje in izvajanje poslovnih procesov, ki temeljijo na spletnih storitvah. BPEL je tipična tehnologija v SOA, ki omogoča fleksibilno sestavljanje spletnih storitev v poslovne transakcije, vključno z komunikacijskimi storitvami. Namenjen je obdelavi dolgotrajnih transakcij oziroma procesov in tudi interakcij z uporabnikom. Droga možnost je implementacija z nizkonivojskim programskim jezikom, kjer je interakcija s storitvijo modelirana s klici API funkcij informacije o stanju pa se hranijo v spremenljivkah. Ko izbiramo način sestavljanja komunikacijskih storitev moramo upoštevati več dejavnikov:

1. Učinkovitost. Za realno časovne komunikacijske storitve je pomembno, da je odzivni čas kratek saj je omejen. Implementacija po meri je običajno bolj učinkovita od BPEL implementacije, ki ima pogosto veliko nepotrebnih dodatkov.

2. Razširljivost. Komunikacijske storitve uporabljajo napredne standarde spletnih storitev kot so WS-Addressing in WS-Eventing. Zaželeno je, da način sestavljanja storitev podpira vse standarde ali pa omogoča razširitev podpore potrebnim standardom. Kot visokonivojski orkestracijski jezik se BPEL fokusira na tok poslovnih procesov in ne podpira nizkonivojskih podrobnosti. Dodajanje podpore novim standardom je lahko težavno in običajno odvisno od ponudnika. Implementacija po meri je namenjena specifični aplikaciji, čeprav je lahko implementacija lahka (light weight) in enostavno podpira določene standarde.

3. Uporabnost. Zaželeno je tudi, da je razvoj novih orkestriranih storitev enostaven. BPEL omogoča grafično modeliranje poslovnih procesov za načrtovanje in razvoj aplikacij. Sestavljanje nove storitve z uporabo



Slika 3.3: Ogrodje SaaS adapter [**?**]

BPEL je relativno enostavno. Implementacija po meri zahteva programiranje podrobnosti na nizkem nivoju, je pa zaradi tega implementacija bolj napredna.

3.4.2 Arhitektura ogradja SaaS adapter

Ogrodje sestavljata dve vrsti storitvenih komponent: tiste, ki so specifične za neko aplikacijo in tiste ki so skupne vsem aplikacijam in storitvam. SAF ponudi skupne komponente kot vgrajene storitve, ki omogočajo razvijalcem, da se osredotočijo na storitveno logiko.

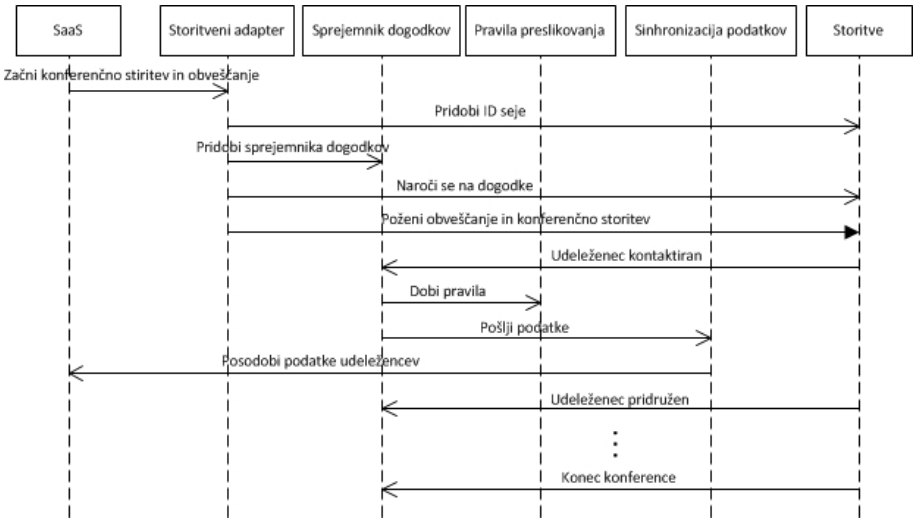
Komunikacijske storitve, prikazane ne desni strani slike 3.3, nudijo vmesnike spletnih storitev SAF-u. Komunikacijske storitve lahko vključujejo mehanizem za obveščanje o dogodkih, ki jih pošiljajo SAF komponenti. Za vsako storitev se v SAF arhitekturi generira nadomestek (stub) storitve, ki je dostopen vsem storitvenim adapterjem. Za vsak storitveni adapter se ustvari WSDL, ki opisuje operacije, katere so na voljo SaaS aplikacijam. Storitveni adapter je zgrajen na osnovi WSDL. Adapter implementira logiko sestavljanja (composition) in transformiranja storitve. Skrbi za klicanje zalednih

(backend) storitev preko nadomestkov (stub) storitev v imenu SaaS aplikacij in koordinacijo prejetih odgovorov glede na tok kompozicije, ki je specifičen za neko storitev. Ko se tok zaključi se končni rezultat pošlje SaaS aplikaciji.

SAF ponuja množico standardnih skladov spletnih storitev vključno z WS-Addressing in WS-Eventing preko skupnega WS-* Support modela. Komponenta Naročanje na dogodke (Event Subscription) storitvenemu adapterju omogoča, da se naroči na dogodke, ki so zanj pomembni, in upravlja prijavljanje na dogodke (obnovi, prekini, itn.).

Če storitveni adapter želi prejemati obvestila o dogodkih mora implementirati sprejemnika dogodkov (event sink) in pošiljatelju dogodkov posredovati informacijo o tem sprejemniku znotraj sporočila za naročanje na dogodek. Sprejemnik dogodkov implementira dogodkovni WSDL, ki ga ponuja vir dogodkov (event source) – ponudnik storitev. Ko se storitev naroča na dogodke, storitveni vmesnik proizvede lokacijo (URL) sprejemnika dogodkov. Ko sprejemnik prejme dogodek, ki je povezan s podatki SaaS aplikacije, mora posodobiti ustrezne podatke SaaS aplikacije. Na primeru konference, to pomeni, da ko se udeleženec pridruži konferenci, konferenčni strežnik proži ParticipantJoined dogodek z njegovim imenom in ID-jem konference, ki ga posreduje sprejemniku dogodkov. Ker SaaS aplikacija hrani status udeležencev interno, mora prejemnik dogodkov poskrbeti za spremembo statusa udeleženca na aplikaciji, ki teče na SaaS platformi.

Ta proces spreminjanja statusa udeleženca izvedejo Sprejemnik dogodkov(Event Sink), pravila preslikovanja dogodkovnih sporočil (Event Mapping Rules) in sinhronizacija podatkov SaaS apliakcije SaaS Data Synchronization (SDS) moduli. Pravila preslikovanja dogodkovnih sporočil so predstavljena v datoteki, ki vsebuje povezave (mapping) med dogodki in ustreznimi SaaS podatki. Povezovanje (mapping) je odvisno od SaaS platforme, povezovanje je lahko podobno SQL poizvedbam ali pa so to XML sporočila za posodabljanje podatkov SaaS platform. Po sprejetju dogodkov, sprejemnik izvede pregled (look up) v datoteki pravi preslikovanja (Mapping Rules). Če ne najde nobenega pravila se dogodek zavrže. V nasprotnem primeru se vrne

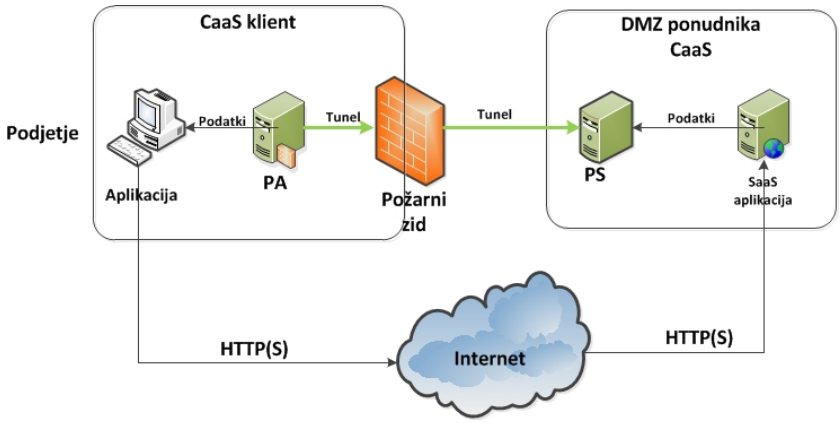


Slika 3.4: Sporočilni tok konferenčne storitve [2]

poizvedba ali pa predloga XML sporočila. Predlogo XML sporočila zapolni sprejemnik dogodkov glede na podatke sprejetega dogodka, nato sporočilo pošlje komponenti Sinhronizacija podatkov SaaS aplikacije (SaaS Data Synchronization).

SDS komponenta skrbi za posodabljanje SaaS podatkov za vse storitve. Zaradi zmanjšanja obremenitve prijavljanja v SaaS platformo, SDS vsebuje SaaS prijavni modul (SaaS Loging). SaaS Loging modul skrbi za prijavljanje v SaaS platformo in ohranjanje seje, tako SDS ne potrebuje vzpostavljati sejo vsakič, ko želi posodobiti podatke.

V primeru na sliki 3.4 vidimo kako poteka interakcija med SaaS aplikacijam in zalednim storitvam z uporabo ogrodja SAF, ter kako SAF komponente komunicirajo med sabo. SaaS aplikacija sproži zahtevo, ki jo pošlje SAF, da sproži konferenčno storitev. Adapter storitve (Service Adaptor) bo najprej izvedel klic operacije getSessionId, kar adapterju storitve omogoči prijavo na konferenčni strežnik, da potrdi prisotnost (avtentikacija) in vzpostavi sejo s konferenčnim strežnikom. Nato storitveni adapter izvede klic operacije za prijavo sprejemanja dogodkov iz strežnika. Zahteva za prijavo vsebuje naslov sprejemnika dogodkov. Ko se klic konča, pokliče naslednjo operacijo



Slika 3.5: Uvedba PASS [2]

za začetek konference in SaaS aplikaciji odgovori s potrditvenim sporočilom (acknowledgment). Ko se stanje udeleženca spremeni, se iz komunikacijskega strežnika, proti ustreznemu sprejemniku dogodkov, pošlje dogodek. Ta poišče sporočilo, ki in če dogodku ustreza in ga pošlje SDS komponente posreduje SaaS aplikaciji.

3.5 Obhod požarnega zidu in NAT

Iz zgornje analize, mora rešitev, ki omogoča prečkanje NAT in požarnega zida – 1) razrešiti notranji URL in ga pretvoriti v naslov ustrezen za usmerjanje, 2) omogočiti vhodne zahteve spletne storitve specifične SaaS aplikacije, 3) rešitev mora biti transparentna SaaS aplikaciji in 4) zahtevati minimalne ali nobenih sprememb konfiguracije požarnega zidu in NAT-a, ker bi s tem tvegali omrežno varnost podjetja.

Slika 3.5 prikazuje vmesno programsko (middleware) infrastrukturo za prečkanje požarnega zidu in NAT (PASS), ki je osnovana na proxy-ju. Infrastruktura nudi učinkovito rešitev za podporo dvosmerne komunikacije spletnih storitev za porazdeljene SaaS aplikacije in integracijo aplikacij tipa SaaS s storitvami, ki gostujejo na opremi znotraj organizacije (on-premise services). V primeru PASS se lahko SaaS povezujeje z aplikacijami in storitvami,

ki so locirane znotraj podjetja, brez da bi te bile izpostavljene javnemu internetu in brez oziroma z minimalnim spreminjanjem konfiguracije požarnega zidu. PASS sistem sestavljata dve vrsti komponent: PASS strežnik (PS) in PASS agent (PA). Tipični scenarij uvajanja je prikazan na sliki. PS je tipično uveden v javni domeni, to je v DMZ SaaS ponudnika. Vsaka stranka pa uvede PA znotraj omrežja svoje organizacije, PA je dosegljiv aplikacijam znotraj podjetja.

3.5.1 Ključne ideje PASS

1. PASS težave s požarnim zidom in NAT rešuje z varovanim tunelom. Tunel je obstojna povezava, ki je vzpostavljena iz strani PA, ki se nahaja v omrežju podjetja, k PS lociranega v javni domeni ali pa DMZ. Ker je povezava vzpostavljena iz notranje strani požarnega zidu, ta povezave ne bo blokiral. Ko je tunel vzpostavljen lahko zahteve SaaS aplikacije dosežejo PA znotraj omrežja podjetja preko PS brez spreminjanja konfiguracije omrežja.

2. PASS izvaja preslikavo ciljnega naslova sporočila z ustreznim tunelom za prečkanje NAT. Namesto usmerjanja sporočila neposredno do ciljnega naslova (ki ga ni mogoče usmerjati – not routable), PS preusmeri sporočilo k PA preko ustreznega tunela. Ko je enkrat zahteva posredovana v omrežje podjetja jo je mogoče usmerjati.

3. Pristop na osnovi namestniškega strežnika (proxy) je transparenten SaaS aplikacijam. PASS je SaaS aplikacijam izpostavljen kot HTTP(S) namestniški strežnik. Za pošiljanje sporočila proti cilju preko PASS mora biti SaaS aplikacija konfigurirana, da uporabi PASS kot svoj odhodni namestniški strežnik. Aplikacije, ki tečejo na strežnikih znotraj podjetja ne potrebujejo nobene spremembe v primeru, da nudijo vmesnike spletne storitve.

3.5.2 PASS agent

PA je nameščen znotraj omrežja podjetja. Njegova naloga je, da vzpostavi tunel s PS pred kakršnokoli izmenjavo podatkov. Pogajanje za odpiranje tunela se izvede z uporabo SSL preko TCP. Ko je tunel vzpostavljen lahko PA preko tunela sprejema zahteve iz PS. Ob sprejemu sporočila PA iz glave sporočila prebere URL naslov storitve, nato izvede poizvedbo v bazi registriranih storitev. Če je ujemajoča storitev najdena bo PA posredoval ustrezni aplikaciji, ki se nahaja v istem omrežju. PA servisira samo registrirane storitve, tiste ki jih podjetje namerava izpostaviti SaaS aplikacijam. Zahteve aplikacijam, ki niso registrirane bodo zavržene. Še bolje, zahteve neregistriranim storitvam ne bodo nikoli dosegle PA, zavrgel jih bo že PS. Tudi v primeru, če PS deluje napačno in posreduje zahtevo na neregistrirano storitev PA, bo storitveni dispečer blokiral zahtevo, ker storitev ni registrirana.

3.5.3 PASS strežnik

PS je vmesni člen, ki premosti komunikacijo med SaaS aplikacije in aplikacije, ki so locirane v podjetju. PS opravlja avtentikacijo in upravlja tuneliranje z večimi PASS agenti. Običajno posluša na vratih 443, ki so namenjena vzpostavljanju tunela. Za vsak vzpostavljen tunel PS določi ID za identifikacijo. Ko je tunel vzpostavljen je lahko uporabljen za posredovanje zahtev za storitve preko PS proti PA.

Iz zornega kota SaaS aplikacij se PS obnaša kot navadni namestniški strežnik. Vzdržuje dinamično usmerjevalno tabelo parov z naslovi URL storitve in ID-jem kanala oziroma tunela. Ob sprejemu zahteve za storitev, PS v svoji usmerjevalni tabeli poišče ustrezen par glede na URL storitve in tako odkrije naslov za naslednji skok – ID tunela. PS nato posreduje zahtevo PA-ju preko ustreznega tunela.

3.5.4 Potek izvajanja

PASS je popolnoma konsistenten z infrastrukturo internetnega dostopa. To ustvari uvajanje PASS-a dokaj enostavno. SaaS aplikacija mora biti konfigurirana tako, da uporablja PS kot izhodni namestniški strežnik, če želi pošiljati zahteve aplikacijam, ki tečejo na strežnikih znotraj podjetja. Na drugi strani pa mora biti storitev podjetja registrirana na PASS, ker tako postane dostopna SaaS aplikacijam.

Po uspešnem vzpostavljanju tunela lahko SaaS aplikacija pošilja zahteve preko PS, ki se obnaša kot spletni namestniški strežnik. PS izvede iskanje po podatkovni bazi registriranih storitev glede na URL storitve. Če najde ujemanje in je hkrati ustrezen tunel proti PA aktiven, posreduje zahtevo PA.

Ob sprejemanju podatkov iz drugega konca tunela PA preveri ali je zahtevana storitev registrirana lokalno. Če je, potem posreduje zahtevo gostitelju storitve, drugače zahtevo zavrže. Ko sprejme odgovor strežnika, odgovor pošlje nazaj po isti poti k SaaS aplikaciji.

3.5.5 Varnost

Varnost je najbolj kritični faktor v PASS. PASS na nivoju aplikacije omogoča (end-to-end) HTTPS iz točke v točko, od SaaS ponudnika do aplikacije podjetja. To tretjim strankam (neznanim), in tudi PASS agentom in strežnikom, onemogoča prisluškovanje. V primerjavi s tradicionalnimi metodami izpostavljanja storitev PASS ponuja dodatni nivo varovanja preko varovanega tunela. To zagotavlja, da lahko do aplikacij podjetja dostopa le ponudnik storitev. Dostop do namestniškega modula PASS strežnika je strogo nadzorovan, tunel lahko prečkajo le avtenticirane in avtorizirane aplikacije.

Poglavje 4

Aplikacija z uporabo CaaS: merjenje gledanosti vsebin v IPTV

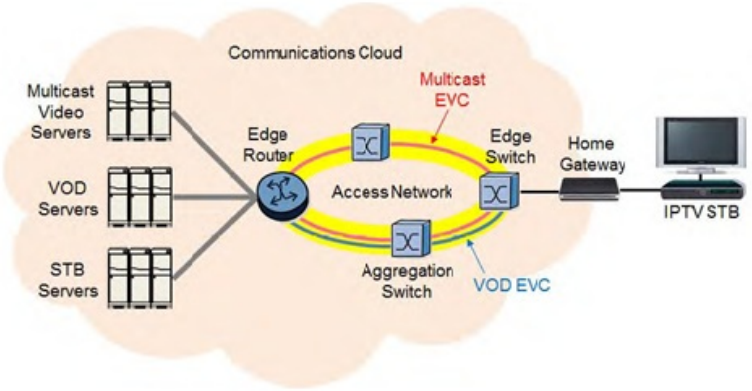
Preden smo se lotili razvoja lastnega sistema smo preučili primer aplikacije CaaS predlagane v članku [3], ki strankam ponudi operacije preko katerih dostopajo do raznih statistik o gledalcih televizijskih vsebin v IPTV komunikacijskih storitvah.

V tem poglavju vidimo, zakaj je pomembno dobro poznati arhitekturo omrežja v kateri bo CaaS rešitev vpeljana. Pri praktičnem delu se nismo usmerili v iskanje rešitve CaaS, podali smo samo predlog, kako bi CaaS integrirali z razvito aplikacijo. Kljub temu nam ta primer pomaga razumeti, kaj vse obsega razvoj zamišljenega sistema, ki bu uporabnikom internetne televizije omogočal naročanje, glasovanje, darovanje v dobrodelni namen, s pritiskom na gumb daljinskega upravljalnika.

4.1 Primer aplikacije za merjenje gledanosti vsebin IPTV

CaaS je, kot že omenjeno, definiran kot model nudenja organizacijskih komunikacijskih storitev navzven (outsourcing). CaaS ponudnik je odgovoren za operacije, administracijo, upravljanje in zagotavljanje z programsko in strojno opremo, ki je potrebna za dostavljanje VoIP, takojšnje sporočanje, sodelovanje, video konference in tudi IPTV. V viru [3] je opisan primer inovativne aplikacije za merjenja gledanosti v IPTV. Aplikacija opisuje, kako preko komunikacijske infrastrukture, ki je na voljo kot CaaS, kar pomeni, da je neodvisna od opreme ponudnika komunikacijskih storitev, merimo statistiko občinstvo televizijske vsebine. Ko govorimo o merjenju statistike občinstva mislimo na gledanost vsebin, čas gledanja vsebine, zbiranje podatkov o priljubljenih vsebinah, ipd.

Pri storitvah IPTV imamo na strani stranke napravo, ki ji pravimo set-top box (STB). Naprava je nameščena na domu uporabnika. Set-top box naprava je odgovorna za dekodiranje video toka, ki ga oddaja video strežnik. STB lahko uporabimo tako, da poroča TV kanal, ki ga uporabnik trenutno gleda, to bi delovalo tako, da STB na STB strežnik periodično pošilja informacijo o TV kanalu ali pa se STB-ju pošlje zahteva na katero odgovori s TV kanalom, katerega vsebino si trenutno ogleduje uporabnik. Temu načinu pravimo, da je osnovan na STB (STB-based). Alternativno lahko uporabimo omrežna stikala oz. Digital Subscriber Line Access Multiplexers (DSLAM) znotraj omrežja za dostopanje. Podatke o gledanju TV kanala v tem primeru poročajo stikala. Ta način je omrežno osnovan (network-based). V obeh primerih zadovoljimo zahteve digitalizacije, prilagajanje, avtomatizacijo, celovitost in natančnost merjenja IPTV občinstva. Vendar, glede na učinkovitost in razširljivost (skalabilnost), je STB osnovan način manj zaželen, ker poveča obremenitev STB naprave in STB strežnika.



Slika 4.1: Poenostavljeno IPTV omrežje v komunikacijskem oblaku [3]

4.2 Pregled CaaS storitev v IPTV porazdeljenem sistemu

Poenostavljena CaaS storitev v obliki IPTV porazdeljenega omrežja v komunikacijskem oblaku, je prikazana na sliki 4.1.

Sestavljajo ga naslednji elementi:

- Čelna stran (headend side): video strežniki za oddajanje več prejemnikom (multiacst), VOD strežniki in STB strežniki.
- Dostopno omrežje (Access network): robni usmerjevalnik, združevalno stikalo in robno stikalo.
- Uporabnikova stran (Customer-premise side): domači prehod (home gateway), IPTV STB in televizijski sprejemnik.

Na čelni strani se IPTV video tok usmerja iz video strežnikov za oddajanje več uporabnikom in VOD strežnikov proti robnim stikalom. Znotraj omrežja za dostop se video tokovi pošiljajo proti robnim stikalom, ki nudijo neposreden omrežni dostop do stranke. Na strani stranke domači prehod sprejme video tok in ga posreduje STB, ki ga dekodira.

Na drugem nivoju omrežja za dostop uporabljamo dve vrsti eternet virtualnih povezav (Ethernet virtual circuits - EVC) za prenos video toka: večtočkovna EVC za oddajanje več uporabnikom za vse IPTV naročnike in več povezav od točke v točko (point-to point) za video na zahtevo (VOD) EVC – eden za vsakega naročnika. EVC za oddajanje več uporabnikom je načrtovan za prenos video tokov za video kanale, ki si jih deli več naročnikov, medtem ko so VOD EVS-ji namenjeni individualnim naročnikom za prenos video tokov za prilagojene VOD storitve.

Omenili smo, da se video tokovi IPTV gledalcem porazdeljeni po principu oddajanje več odjemalcem (multicast) in ne oddajanje vsem odjemalcem (broadcast), tako so IPTV programi dostavljeni le specifičnim strankam (uporabnikom, gledalcem), tistim ki so naročeni na IPTV storitve. Zaradi upravljavskih razlogov se uporablja IGMP (Internet group management protocol), ki ga uporabljamo za upravljanje video tokov za oddajanje več odjemalcem na povezovalnem nivoju. Ko uporabnik izvede menjavo kanala STB sproži IGMP Membership Report in Leave Group sporočil robnemu stikalu. Ob sprejemu tega IGMP sporočila bo robni usmerjevalnik začel s pošiljanjem novega video toka in prekinil pošiljanje starega. Posledica tega je, da je vsebina video kanala prisotna samo takrat, ko je na kanal prijavljen vsaj en uporabnik. V nasprotnem primeru distribucija kanala ni potrebna, kar pomeni, da je poraba pasovne širine manjša pri EVC za oddajanje več uporabnikom. Te lastnosti ločijo IPTV od tradicionalnega TV oddajanja, ki vsebino oddaja ne glede na prisotnost uporabnika oz. gledalca. Še več, CaaS ponudniki (komunikacijski operaterji), ki igrajo vlogo ponudnika IPTV lahko vzpostavijo več multicast EVC in ne samo enega in tako povečajo fleksibilnost. Različni multicast EVC-ji lahko nosijo različne IPTV pakete (različno kombinacijo TV kanalov) ali pa tudi vsebino različnih IPTV ponudnikov. Ne glede na to koliko multicast EVC-jev obstaja v omrežju je IGMP protokol uporabljen za upravljanje prenosa video tokov za oddajanje več uporabnikom (multicast) v vsakem EVC.

Uporaba IGMP protokola za upravljanje multicast IPTV distribucije omogoča

učinkovito izrabljanje virov pasovne širine. Ampak, ker se zahteve po visoko ločljivi (HD) vsebini povečujejo, morajo CaaS ponudniki še bolj previdno upravljati uporabo multicast EVC-jev, ker HD vsebina zahteva več pasovne širine kot vsebina standardne ločljivosti (SD). Da bi pridobili natančno in podrobno poročilo o uporabi multicast EVC moramo pridobiti oceno gledanosti periodično, da pridobimo statistiko gledanosti programov – kdaj so gledani, kateri so gledani. Podatki in informacije IPTV občinstva so pomembni tudi za CaaS stranke – ponudniki vsebine, s katerimi prepričajo oglaševalce o populaciji, ki je v določenem času gledala določeno vsebino in katri oglas so pri tem videli. Kot rezultat za telekomunikacijske operaterje, ki želijo ponuditi CaaS storitve, je zelo pomembno, da v svoje storitve vključijo natančno in učinkovito metodo merjenja IPTV občinstva.

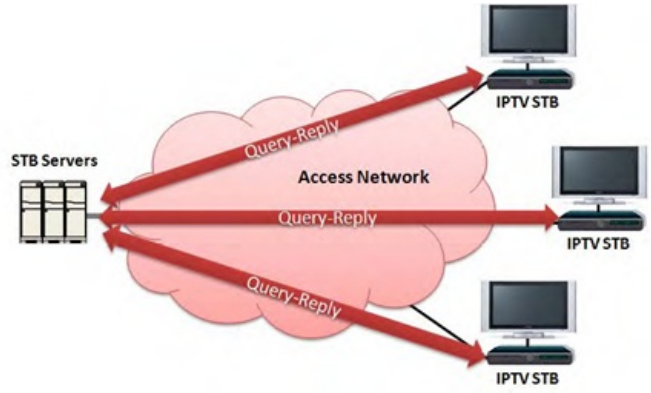
4.3 Arhitektura sistema za merjenje IPTV občinstva

Avtorji članka [3] predlagajo dve različni arhitekturi za pridobivanje informacij občinstva: lahko jih zbiramo iz STB naprav (STB osnovan) ali omrežnih stikal (DSLAMS) v omrežju za dostopanje (omrežno osnovan).

4.3.1 Merjenje občinstva IPTV na osnovi STB sistema

Pri tem načinu informacije o gledanju kanalov pridobimo iz STB naprave, ki se nahaja pri uporabniku. Bolj natančno, STB naprava lahko pošlje statistiko gledanja STB strežniku periodično ali na zahtevo. Arhitektura sistema je prikazana na sliki 4.2, kjer vidimo da STB strežnik pošilja poizvedbe STB napravam, te pa odgovorijo z informacijami o gledanosti. Izmenjava sporočil zahteva-odgovor je lahko implementirana s prenosnim sistemom (middleware) STB sistema.

Kljub temu, da lahko s STB osnovano shemo dosežemo cilje celovitosti in natančnosti pri merjenju občinstva, nastane težava pri razširljivosti (ska-



Slika 4.2: STB osnovana arhitektura za merjenje statistik občinstva [3]

labilnost). Ta težava nastane s povečevanjem IPTV naročnikov, zbiranje statistike gledanosti neposredno iz vseh STB naprav lahko in najverjetneje bo drastično povečalo obremenitev STB strežnikov. Težavo bi lahko reševali z distribuiranjem STB strežnikov, kar pomeni da lahko med strežnike razdelimo obremenitev, vendar obremenitev posameznega strežnika je še vedno sorazmerna s številom naročnikov IPTV. Poleg tega lahko v STB osnovani shemi nastanejo težave s kompatibilnostjo. IPTV ponudniki lahko razširijo svoje sisteme vsako leto, v sistemi imamo tako lahko STB naprave različnih ponudnikov. Iz tega sledi, da implementacija merjenja občinstva preko STB prenosnega sistema (middleware) verjetno z določeno verzijo STB naprave ne bo delovala pravilno.

4.3.2 Omrežno osnovano merjenje statistik občinstva IPTV

Slika 4.3 prikazuje sistemsko arhitekturo tako imenovanega omrežno osnovanega sistema za merjenje IPTV občinstva, domači prehod je zaradi enostavnejše predstavitve izpuščen. V IPTV porazdeljeni platformi so robna stikala odgovorna za posredovanje video toka gledalcem, kar pomeni, da lahko informacije o gledanih kanali pridobimo iz teh stikal. Stikala omogočajo nepo-

sreden omrežni dostop do strank (customer premises). Posledica tega je, da so podatki o gledanih vsehinah na voljo "na vrata" (vemo katera vsebina se oddaja na portu) na robnem stikalu.

Ključni del IPTV platforme za distribuiranje je, da vsako robno stikalo hrani IGMP snooping tabelo za sledenje multicast stanja vsakih vrat naročnika. Natančneje IGMP snooping tabela prikazuje trenutno IGMP skupino na kateri so prijavljena določena vrata. Podatke IGMP snooping tabele lahko zbiramo periodični na osnovi katerih pridobimo naslednje informacije:

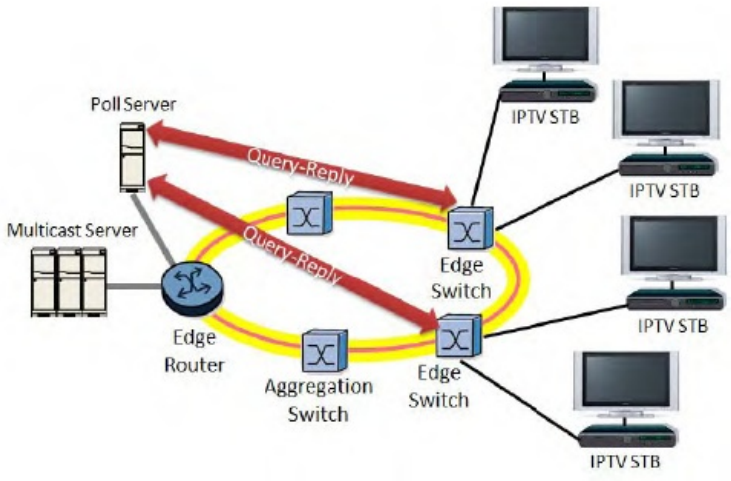
- Identiteto naročnika, ki gleda določen TV kanal.
- Ura in dolžina časa, ki ga je naročnik porabil za gledanje določenega TV kanala.
- Katere kanale je gledal določen naročnik.
- Število gledalcev določenega TV kanala.

Tu moramo biti pazljivi, namreč nekateri podatki so osebne narave in zato občutljivi. Stranke IPTV storitev ali CaaS ponudniki brez dovoljenja IPTV naročnikov ne smejo izpostaviti njihovi osebnih informacij.

4.3.3 Primerjava shem za merjenje občinstva

Prednost merjenja občinstva IPTV po omrežno osnovanem načinu pred STB osnovanem načinu je večkratna:

1. Omrežno osnovan način lahko vpeljemo v vse IPTV distribucijske platforme, ki uporabljajo IGMP, ne glede na programsko opremo STB sistemov.
2. Uporaba omrežno osnovanega načina ne zahteva nobenih posodobitev obstoječega omrežja, med tem, ko je pri STB osnovanem načinu to potrebno.



Slika 4.3: Omrežno osnovana arhitektura za merjenje občinstva [3]

3. STB napravam ni potrebno sodelovati pri zbiranju podatkov, zato je ta način naročniku popolnoma transparenten.

4. Pri omrežno osnovanem načinu statistiko ogledov pridobimo iz robnih stikal, ki tipično servisira 24 do 48 naročnikov, kar pomeni, da je ta način bolj razširljiv kot STB osnovan pristop.

Poglavje 5

Tehnologije uporabljene pri razvoju lastne aplikacije

Odločili smo se, da bomo tudi sami zasnovali prototip sistema, ki bi naročnikom internetne televizije ponudil interaktivne storitve, kot naročanje oglaševanega produkta, izvedba darovanja v dobrodelni namen in glasovanje ob gledanju resničnostnega šova s pritiskom na gumb daljinskega upravljalnika.

Jedro našega sistema predstavlja aplikacija tipa SaaS, ki se povezuje z aplikacijami strank in aplikacijami ponudnikov internetne televizije. Aplikacija je sestavljena iz množice vmesnikov, ki zunanjim aplikacijam izpostavijo metode, preko katerih na primer aplikacije ponudnikov internetne televizije shranjujejo naročila, glasove in obvestila o darovanjih v dobrodelne namene, aplikacije prodajalcev pa imajo na voljo metode preko katerih sprejemajo naročila in ostala sporočila. Za implementacijo smo uporabili tehnologije Java EE.

5.1 Pregled uporabljenih tehnologij

Vmesniki aplikacije so implementirani kot spletne storitve v Java EE tehnologiji z uporabo JAX-WS (Java API for XML Web Services). Prototip aplikacije teče na IBM WebSphere v8.5 aplikacijskem strežniku.

Spletne storitve nudijo sredstva za doseganje interoperabilnost med različnimi aplikacijami, ki tečejo na različnih platformah. Spletne storitve omogočajo komunikacijo med napravami preko interneta, med katerimi s prenašajo sporočila, ki jih določimo s "pogodbo" spletne storitve, pri opisu vmesnika. Vmesnik spletne storitve opišemo z WSDL (Web Service Description Language), pri razvoju svoje aplikacije smo uporabili WSDL 1.2 [9]. Komunikacija med sistemi poteka preko SOAP (Simple Object Access Protocol) sporočil, ki se prenašajo z uporabo protokola aplikacijske plasti - največkrat HTTP. Format SOAP sporočil je osnovan na XML (Extensible Markup Language).

Spletne storitve so tehnologija za integracijo šibko sklopljenih komponent. To pomeni, da se lahko sporočila prenašajo med različnimi platformami. Prenos sporočil se izvja preko klicev ustreznih operacij, ki jih izpostavlja vmesnik. Izmenjava sporočil med komponentami lahko poteka na dva načina: sinhrono, to pomeni, da klicatelj čaka na odgovor, in asinhrono, klicatelj ne čaka odgovora in nadaljuje svoje delo. Pri asinhronem načinu, spletna storitev odgovori klicatelju, ko dokončno obdela zahtevo. Prednost asinhronih klicev je, da storitev ne blokira klicatelja, vendar ni možno vedno uporabiti asinhronih klicev, če kličoča aplikacija za nadaljevanje izvajanja potrebuje podatke spletne storitve, mora uporabiti sinhron klic. Storitve lahko komunicirajo po večih principh: zahteva/odgovor - za vsako zahtevo se tvori odgovor, enosmerno, obvestilo, spodbujen odgovor. Prototip uporablja princip zahteva/odgovor.

K razvoju spletnih storitev lahko pristopimo na dva načina: "od spodaj navzgor" in od "zgoraj navzdol". Pri načinu "od spodaj navzgor" najprej napišemo programsko kodo storitev, implementiramo metode, nato iz te kode generiramo WDSL dokument, ki izpostavi te metode. Na koncu implementiramo še razrede, ki serializirajo parametre in rezultate metod v SOAP sporočila. Pristop "od zgoraj navzdol" najprej zahteva definicijo podatkov, njihovo arhitekturo in vmesnike – najprej napišemo WSDL dokument, v naslednjem koraku iz WSDL dokumenta generiramo metode in jih implementiramo, na koncu se definira varnostna politika za dostop do storitev.

Pri razvoju prototipa smo uporabili pristop "od zgoraj navzdol". Najprej smo definirali sporočila in njihovo strukturo v WSDL, nato pa smo generirali metode. Za ta pristop smo se odločili zato, ker se je bilo v zgodnji fazi načrtovanja lažje osredotočiti na sporočil, ki se prenašajo, kot pa na kodo, ki jih obdeluje. Prototip smo razvili v razvojnem okolju IBM Rational Application Developer v8.5 (RAD).

5.1.1 JAX-WS

Java API for XML Web Services (JAX-WS) je API za programski jezik Java s katerim ustvarjamo spletne storitve. JAX-WS je del Java EE platforme, ki kot večina ostalih APIjev Java EE platforme uporablja anotacije, ki poenostavljajo razvoj odjemalcev spletne storitve in končnih točk (endpoint). Več o JAX-WS na [**?**].

Storitve implementirane v prototipu se generirajo z generatorjem, ki uporablja JAX-WS verzijo 2.2. Storitve tečejo v IBM WebSphere JAX-WS izvajalnem okolju na aplikacijskem strežniku IBM WebSphere 8.5.

Primer kode vmesnika in implementacije vmesnika končne točke spletne storitve vidimo spodaj, uporabljene so JAX-WS anotacije.

Vmesnik:

```
package paket;

import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public interface Greeting {
        @WebMethod String sayHello(String name);
}
```

Implementacija kode vmesnika:

```
package paket;
import javax.jws.WebService;

@WebService(endpointInterface = "paket.Greeting")
public class GreetingImpl implements Greeting {

        @Override
        public String sayHello(String name) {
                return "Hello, Welcom to jax-ws " + name;
        }
}
```

5.1.2 JAXB

Java Architecture for XML Binging (JAXB) omogoča pretvarjanje Java objektov v XML format in obratno. Z uporabo JAXB nam ni potrebno implementirati operacije za shranjevanje in branje XML formata. JAXB poenostavi delo z XML saj se razvijalcem ni potrebno poglabljati v tehniko XML.

Ob generiranju spletne storitve definirane v WSDL dokumentu smo generirali tudi serializacijske JAXB razrede, ki predstavljajo sporočila definirana v WSDL 1.2.

Primer sporočila definiranega v WSDL dokumentu in generiranega Java razreda.

Primer definicije elementa sporočila v WSDL 1.2

```
<xsd:element name="orderRequest">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="buyer" type="tns:userType"/>
```

```
        <xsd:element name="productId" type="xsd:int"/>
        <xsd:element name="quantity" type="xsd:int"/>
        <xsd:element name="paymentMethod" type="xsd:int"/>
        <xsd:element name="orderTime" type="xsd:dateTime"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Zgornji element je del zahteve za naročilo. Zahteva za naročilo mora vsebovati naslednje elemente: 1) podatki o kupcu (userType - definiran v WSDL), 2) ID izdelka - productId, 3) količina- quantity, 4) način plačila - paymentMethod in čas naročila - orderTime.

Primer definicije sporočila v obliki javanskega razreda

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "buyer",
    "productId",
    "quantity",
    "paymentMethod",
    "orderTime"
})
@XmlRootElement(name = "orderRequest")
public class OrderRequest
    implements Serializable
{
    @XmlElement(required = true)
    protected UserType buyer;
    protected int productId;
    protected int quantity;
    protected int paymentMethod;
    @XmlElement(required = true)
```

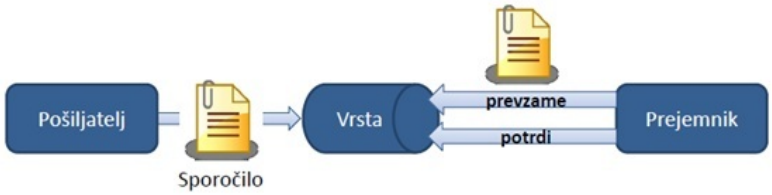
```
    @XmlSchemaType(name = "dateTime")
    protected XMLGregorianCalendar orderTime;

    /**
     * Getters and setters methods
     */
}
```

5.1.3 JPA

Pri razvoju prototipa smo uporabili tudi JPA tehnologijo. Java Persistence API (JPA) je ogrodje za upravljanje relacijskih podatkov. Glavna komponenta JPA so entitetna zrna. Entitetna zrna so Java razredi, katerih stanja so trajno hranjena v relacijski podatkovni bazi. Instanca entitetnega zrna predstavlja vrstico v tabeli. Med entitetami tipično obstajajo relacije, ki jih definiramo v kodi entitetnega razreda z uporabo anotacij. Relacije so lahko definirane tudi v ločeni XML datoteki. Anotacije se uporabljajo tudi za definiranje entitet. JPA omogoča enostavno izvajanje osnovnih CRUD (Create, Read, Update and Delete) operacij - štiri osnovne operacije persistenčne shrambe, to so shranjevanje v bazo – persist(), posodabljanje – merge() in pridobivanje – find() ter odstranjevanje – remove() objektov iz podatkovne baze. Vse CRUD operacije se izvajajo znotraj transakcijskega konteksta, to pomeni, da se stanje entitete spremeni samo v primeru uspešno izvedene transakcije.

Pri razvoju prototipa smo najprej postavili MySQL podatkovno bazo, določili entitetne tipe in relacije med njimi. Nato smo iz obstoječih tabel v okolju RAD z uporabo orodja JPA Tools generirali entitetne razrede in preko čarovnika določili njihove medsebojne odvisnosti.

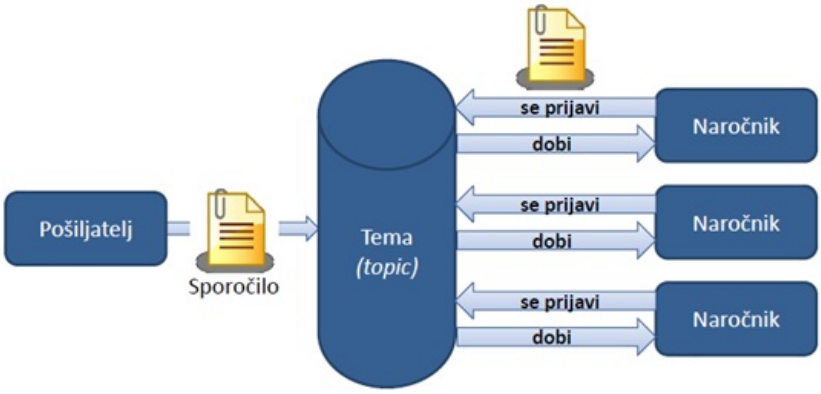


Slika 5.1: JMS vrsta

5.1.4 Java Message Service - JMS

Java Message Service je Java EE, ki skrbi za komunikacijo s sporočilnim sistemom, t.i. Message-oriented Middleware (MOM). Sporočilni sistemi omogočajo asinhrono komuniciranje med enotami sistema. JMS podpira dva glavna načina pošiljanja sporočil: Točka v točko (P2P) in objavi/naroči (Publish/Subscribe) - članek [8], poglavje Message Delivery Models. Pri P2P načinu imajo sporočila samo enega naslovnika. Sporočila se nalagajo v FIFO vrsto, kjer čakajo na dostavo, primer delovanja vrste je viden na sliki 5.1. Sporočilo se v vrsti hrani dokler prejemnik ne potrdi prejema. Pri objavi/naroči načinu je naročilo lahko dostavljeno večim prejemnikom. Prejemniki, ki želijo prejemati sporočila, se morajo naročiti na temo (topic). Primer uporabe teme je prikazan na sliki 5.2.

JMS omogoča razvoj asinhronih spletnih storitev. S pomočjo JMS lahko poslovno logiko ločimo od vmesnika končne točke spletne storitve (web service endpoint). Vmesnik pri asinhronih spletnih storitvah preveri vsebino prejetega sporočila, če ta ustreza, odjemalcu potrdi sprejetje in sporočilo posreduje JMS vrsti. Komponenta, ki izvaja poslovno logiko, sprejme sporočilo iz JMS vrste in ga obdela. Tipično asinhrone spletne storitve delujejo tako, da po obdelavi odjemalcu vrnejo rezultat, ker je našem primeru rezultat zapis v podatkovno bazo to ni potrebno.



Slika 5.2: JMS tema

5.1.5 Sejna in sporočilna zrna

EJB – Javanska strežniška zrna so komponente za modularno izgradnjo poslovnih aplikacij. EJB je strežniški model, ki povzema poslovno logiko aplikacije. EJBji nudijo standarden način za implementacijo zaledne (backend) poslovne kode. Od EJB verzije 3.0 naprej poznamo dva tipa javanskih strežniških zrn: sejna zrna in sporočilna zrna (MDB). Sejna zrna opravljajo poslovne operacije, orkestrirajo transakcije ter upravljajo z dostopi. Poznamo dve vrsti sejnih zrn: sejna zrna brez stanja (stateless), ki ne vzdržujejo stanja in sejna zrna s stanjem (stateful), kjer je instanca sejnega zrna povezana s specifično zahtevo (običajno sejo).

Sporočilna zrna (MDB) so namenjena asinhronemu sprejemanju sporočil iz JMS vrst ali tem. Sporočilno zrno implementira metodo onMessage(), ki jo proži EJB vsebnik ob prispetju sporočila. V primeru prototipa sporočilno zrno skrbi za zapis prejetega sporočila v podatkovno bazo.

Poglavje 6

Načrtovanje in razvoj aplikacije

Ko smo preučili primer rešitve za merjenje raznih statistik v IPTV sistemu, smo se vprašali kako lahko tak sistem nadgradimo oziroma razširimo. V poglavju 3 smo pogledali, kako med sabo integriramo porazdeljene sisteme, torej kako SaaS aplikacije (aplikacije v oblaku) integriramo z aplikacijam, ki se nahajajo na strežnikih v podjetju. Izvedeli smo tudi kaj je CaaS, kaj nam CaaS storitve nudijo in za kaj so uporabne. Z vsem tem znanjem smo se lotili zasnove arhitekture sistema, ki bi interaktivno IPTV popeljala na višjo raven.

Zamislili smo si sistem, ki ne bi samo zbiral podatke o gledanosti kanalov in o vsebini, ki je najbolj priljubljena med naročniki IPTV, ampak bi naročnikom IPTV omogočil izvajanje različnih akcij, na primer na nekem kanalu se predvaja TV oddaja v kateri gledalce pozivajo k telefonskemu glasovanju, vendar namesto, da mora uporabnik opraviti klic, glasovanje opravi preko daljinskega upravljalnika. Sistem bi v tem primeru zbiral dogodke, ki jih prožijo naročniki preko različnih akcij. Če uporabnik glasuje za nek dogodek, sistem sprejme njegov glas in ga shrani. Ko se glasovanje zaključi sistem naročniku glasovanja pripravi rezultate. Ta funkcionalnost je lahko uporabljena tudi za izvajanje anket na primer med oglasi. Torej imamo sistem, ki povezuje uporabnika internetne televizije in stranko, ki želi infrastrukturo IPTV uporabiti v svojo korist, v tem primeru kot medij za izvajanje glaso-

vanja, ki bi ga uporabila televizijska hiša oziroma ponudnik TV programa. V nadaljevanju bomo podrobneje predstavili sistem, njegove funkcionalnost in kako se integrira z ostalimi sistemi.

Glavna lastnost takega sistema je, da se nahaja v oblaku. To pomeni, da je sistem dostopen vsakomur, ki sistem najame. Jedro sistema je SaaS aplikacija, ki se na eni strani povezuje s strankam (podjetja, televizijske hiše, itn.) in ponudniki internetne televizije (ISP), posredno naročniki internetne televizije. Pri praktičnem delu smo se osredotočili na jedro sistema torej SaaS aplikacijo. Podali smo predlog, kako lahko IPTV ponudnik realizira pošiljanje zahtev uporabnikov SaaS aplikaciji, vendar nismo iskali optimalno rešitev, ki bi bila takšna, da čim manj obremenjuje obstoječo IPTV infrastrukturo.

Naučili smo se, da dostop do komunikacijskih storitev podjetja, v tem primeru je to internetni ponudnik, omogočimo preko CaaS. Kot smo videli v primeru aplikacije za merjenje občinstva, CaaS strankam ponudi storitve preko katerih dostopajo do rezultatov meritev. Naš sistem bo enostavnejši. Vsako akcijo, ki jo proži uporabnik, ponudnik internetne televizije posreduje SaaS sistemu, tako SaaS aplikacije ne potrebujejo dostopa do infrastrukture ponudnikov internetne televizije, vsaj ne v fazi razvoja prototipa. V vsakem primeru pa mora ponudnik internetne televizije postaviti sistem, ki zna SaaS aplikacijam posredovati zahteve, katere prožijo uporabniki internetne televizije.

V primeru aplikacije za merjenje gledanosti smo opazili, da sistem skoraj nič ne obremenjuje obstoječe IPTV infrastrukture. To želimo tudi pri naši aplikaciji, vendar naš sistem vseeno zahteva nekaj nadgradenj. Ponudnik internetne televizije mora uporabnika, če ima te interaktivne storitve vklopljene, opozoriti na možnost akcije. Predpostavimo, da IPTV ponudnik ima informacije o tem, kdaj je katera akcija na voljo in za koliko časa, na primer: neka televizijska hiša naroči IPTV ponudniku, da v nekem časovnem intervalu omogoči akcijo »glasuj«. Druga stvar, za katero mora IPTV ponudnik poskrbeti je uporabniški vmesnik, ki se prikaže ob proženju neke akcije. Ob proženju akcije »glasuj« se gledalcu prikaže seznam iz katerega izbere vre-

dnost za katero želi glasovati. Tretja stvar pa je posredovanje zahtev SaaS aplikaciji. Sistem za IPTV ponudnika nima nobene koristi, če mora IPTV ponudnik vse zahteve, preden jih posreduje SaaS aplikaciji, obdelati sam. Naletimo na iste težave kot pri STB osnovanem načinu merjenja gledanosti, obremenjujemo IPTV infrastrukturo. Tu tega ne moremo popolnoma obiti. Predlog ne najboljše rešitve je, da IPTV ponudnik preko aplikacije znotraj infrastrukture posreduje zahtevo SaaS aplikaciji. Razlog, zakaj STB naprave ne more sama poslati zahteve, je ta, da STB naprava nima vseh potrebnih informacij, ki jih mora v zahtevo vključiti Na primer identifikator dogodka za katerega uporabnik glasuje ali pa identifikator produkta za katerega želi uporabnik izvesti nakup. Predpostavimo, da se ti podatki nahajajo v podatkovni bazi znotraj infrastrukture ponudnika. Dostop do teh podatkov ima strežnik na katerem teče aplikacija, ki zbira in posreduje zahteve uporabnikov SaaS aplikaciji v oblaku.

6.1 Funkcionalnosti

Pri snovanju sistema smo se osredotočili na tri glavne funkcionalnosti: nakupovanje preko IPTV, glasovanje preko IPTV in nakazilo denarne pomoči preko IPTV. Vse funkcionalnosti so del nekega vmesnika spletne storitve SaaS aplikacije. V nadaljevanju bomo predstavili, kako posamezne funkcionalnosti delujejo, kaj omogočajo in kako so funkcionalnosti razdeljene po vmesnikih.

6.1.1 Scenarij “nakup”

Na nekem televizijskem kanalu se predvaja oglas za nek produkt. Ko uporabnik preklopi na ta kanal se mu prikaže opozorilo, na primer »Za nakup na daljinskem upravljalniku pritisnite rdeč gumb«. Če se uporabnik odloči za izvedbo te akcije se prikaže vmesnik, kjer uporabnik izbere način plačila in količino. Ob potrditvi se zahteva posreduje aplikaciji znotraj infrastrukture IPTV. Aplikacija zahtevo opremi z dodatnimi podatki: ime in priimek uporabnika, naslov za dostavo in identifikator produkta, vse te podatke o svojih

naročnikih IPTV ponudnik ima, nato pa jo posreduje SaaS aplikaciji, ki bo naročilo predala prodajalcu. Podatke o produktu aplikacija pridobi glede na čas, ko je bila zahteva oddana in glede na kanal, na katerem je bila zahteva oddana. Aplikacija v podatkovni bazi preveri kateri produkt se na kanalu prodaja v času proženja zahteve.

6.1.2 Scenarij "glasuj"

Gledalec je pozvan h glasovanju. Ob pritisku na gumb se gledalcu na zaslon prikaže seznam možnosti za izbiro, gledalec izbor potrdi in pošlje zahtevo. Zahteva se ponovno opremi z ostalimi parametri o uporabniku in dogodku za glasovanje, nato se posreduje SaaS aplikaciji. Informacija, ki mora biti shranjena v tem primeru je identifikator dogodka za glasovanje, sistem to informacijo ponovno pridobi glede na čas oddaje zahteve in kanala. Glasovanje se uporabniku zaračuna preko položnice njegovega ponudnika internetne televizije.

6.1.3 Scenarij "daruj"

Gledalec ima ob pozivu za darovanje v humanitarne namene možnost darovanja. Darovanje deluje tako, da uporabnik ob pritisku na gumb vnese vsoto, ki jo želi darovati. Darovana vrednost se uporabniku zaračuna na položnici ponudnika internetne televizije. Zahteva se v tem primeru opremi z informacijami o tem komu je namenjena humanitarna pomoč.

Poleg teh glavnih funkcionalnosti imamo še tri pomožne.

- Status nakupa: Uporabnik lahko preveri stanje naročila. Naročilo ima lahko naslednja stanja: naročilo še ni prejeto, naročilo v obdelavi, naročilo odposlano.

- Preklic nakupa: Ta akcija je možna dokler prodajalec naročila ne odpremi.

- Pregled opravljenih akcij v tekočem mesecu: Uporabnik lahko preveri katere akcije je opravil v tekočem mesecu, kdaj jih je opravil in koliko je posamezna akcija stala.

6.1.4 Scenarij "sprejmi nova naročila"

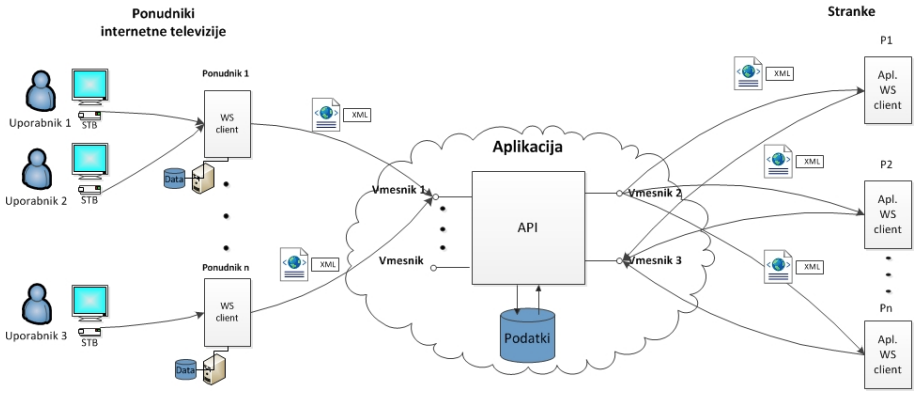
Naročila, ki so jih prožili gledalci se shranijo v podatkovno bazo SaaS aplikacije. Potrebujemo rešitev, ki bo prodajalcu dostavila nova naročila. Sprejem novih naročil deluje tako, da stranka odda zahtevo, ko želi nova naročila, SaaS aplikacija pa odgovori s seznamom novih naročil. Prototipni sistem deluje tako, da aplikacija podjetja povprašuje po novih naročilih periodično, na primer vsakih 30 minut. Drugi načini za dostavljanje novih naročil bomo opisali v nadaljevanju.

6.1.5 Scenarij "rezultati glasovanja"

Naročniku glasovanja omogočimo sprejetje rezultatov preko vmesnika. Sistem sprejema glasove gledalcev, ki jih zapisuje v podatkovno bazo. Tudi v tem primeru stranka rezultate pridobi tako, da na primer po koncu glasovanja pošlje zahtevo po rezultatih. Odgovor na zahtevo je seznam parov vrednost in število glasov.

6.1.6 Scenarij "seznam donacij"

Funkcionalnost stranki omogoči pridobiti seznam vrednosti, ki so jih uporabniki darovali za nek humanitarni dogodek. Gre za preprost obvestilni sistem darovanja v dobrodelne namene. Zamisel pri darovanju je, da se uporabniku darovano vrednost zaračuna kot dodatna uporabljena storitev na položnice, ki mu jo izda ponudnik internetne televizije. V tem primeru prenos nakazila opravi ponudnik internetne televizije.



Slika 6.1: Osnovni koncept sistema

6.2 Arhitektura sistema

Slika 6.1 prikazuje osnovni koncept sistema. Sistem lahko ločimo na tri dele: levi del – ponudniki internetne televizije, centralni del – aplikacija, vmesniki spletnih storitev, ki se nahajajo v oblaku in desni del – stranke, to so ponudniki televizijskih programov, oglaševalci, prodajalci, ipd. Na sliki 7 lahko hitro opazimo, zakaj je smiselno, da se takšna aplikacija nahaja v oblaku. Če se poglobimo v delovanje sistema ugotovimo, da so posamezni vmesniki zelo obremenjeni.

Poglejmo si vmesnik 1, ki skrbi za sprejem naročil, glasov in zbiranje pomoči. Ta vmesnik lahko uporablja kateri koli ponudnik internetne televizije. Vsak internetni ponudnik ima množico IPTV naročnikov, ki lahko prožijo katero koli od ponujenih akcij, vse akcije pa se zbirajo na enem vmesniku.

Poglejmo si glasovanje, predvsem ker je verjetnost, da bo ta akcija izvedene večkrat kot nakup in akcija daruj. V neki oddaji pozivajo h glasovanju. Tipično je glasovanje časovno omejeno, na primer glasovanje je mogoče znotraj časovnega intervala 15 minut. To pomeni, da mora vmesnik SaaS aplikacije v kratkem času obdelati veliko zahtev, obdelati mora zahteve uporabnikov vseh IPTV ponudnikov, ki omogočajo ta način glasovanja.

Časovno omejeno glasovanje povzroči veliko koncentracijo zahtev v kratkem časovnem obdobju, čemur rečemo »peak hour«. Kljub temu, da je glasovanje hkrati možno tudi na drugem kanalu, se v tem primeru uporabniki porazdelijo, lahko predpostavimo, da imamo v trenutku neko povprečno množico aktivnih uporabnikov. Te težave moramo upoštevati že v samem začetku načrtovanja programske arhitekture vmesnika za sprejem zahtev.

6.2.1 Opis delovanja posameznih komponent

6.2.1.1 Strežnik za posredovanje zahtev

Strežnik za posredovanje zahtev vsebuje vsak IPTV ponudnik, ki omogoča zgoraj opisane akcije. Strežnik se nahaja znotraj sistema IPTV ponudnika. Ta strežnik sprejme zahteve naročnikov IPTV, ki so prožili določeno akcijo. Zahteva, ki jo je poslal STB vsebuje čas, številko kanala in dodatne podatke, če jih akcija zahteva. Na primer pri glasovanju je to vrednost za katero uporabnik glasuje. Aplikacija na strežniku identificira akcijo in zahtevi doda informacije, ki so povezane s to akcijo. Podatkovna baza znotraj infrastrukture vsakega IPTV ponudnika vsebuje podatke o tem, katera akcija je na nekem kanalu na voljo v nekem časovnem intervalu. Glede na ta dva parametra aplikacija iz podatkovne baze pridobi informacije glede na akcijo, o produktu, o dogodku za glasovanje oziroma o dogodku za humanitarno pomoč. Pridobljene informacije se vključijo v zahtevo, poleg teh se dodajo še informacije o uporabniku, pri nakupu so to ime, priimek in naslov. Zahteva se preko odjemalce spletne storitve pošlje SaaS aplikaciji.

6.2.1.2 SaaS aplikacija

SaaS aplikacijo sestavljajo vmesniki spletnih storitev, aplikacij v splošnem razdelimo na dva dela. Prvi del skrbi za sprejem zahtev, ta zahteve obdela in zapiše v podatkovno bazo. Prvi del SaaS aplikacije predstavlja vmesnik 1. Drugi del skrbi za dostavo naročil, rezultatov glasovanja in obvestil o darovanju končnim strankam, to je vmesnik 2 na sliki 6.1. Aplikacije ponu-

dnikov internetne televizije, ki skrbijo za posredovanje zahtev SaaS aplikaciji tako komunicirajo s prvim delom, to je z vmesnikom 1 6.1. Vmesnik 2 pa predstavlja dostopno točko strankam.

Ko govorimo o zahtevam mislimo na zahteve http, pa katerih se prenašajo XML sporočila.

6.2.1.2.1 Sprejem zahtev

Za sprejem zahtev skrbi ločen vmesnik spletne storitve, prikazan je na sliki koncepta sistema 6.1. Gre za vmesnik 1, s katerim se povezujejo ponudniki internetne televizije. Vmesnik vsebuje metode, ki odgovarjajo na zahteve določene akcije. Vsaka metoda je namenjena ločeni akciji. Metoda, ki sprejme zahtevo preveri, če zahteva vsebuje zahtevane parametre in identifikacijo klicatelja. Klicatelj se identificira tako, da v zahtevi poda svoj id, ta način v praksi ni uporaben, potrebna je razširitev spletne storitve z uporabo WS-Security, ki podpira storitve varnosti kot avtentikacija, enkripcija in digitalni podpis, več na [**?**]. Če ima vse potrebne parametre se zahteva posreduje komponenti, ki poskrbi za zapis v podatkovno bazo. Aplikacija je tako razdeljena na sprednji (sprejemni) del (front end), ki sprejema in preverja zahteve, ter zaledni del (back end), ki poskrbi za izvedbo poslovne logike, to je v tem primeru zapis v podatkovno bazo.

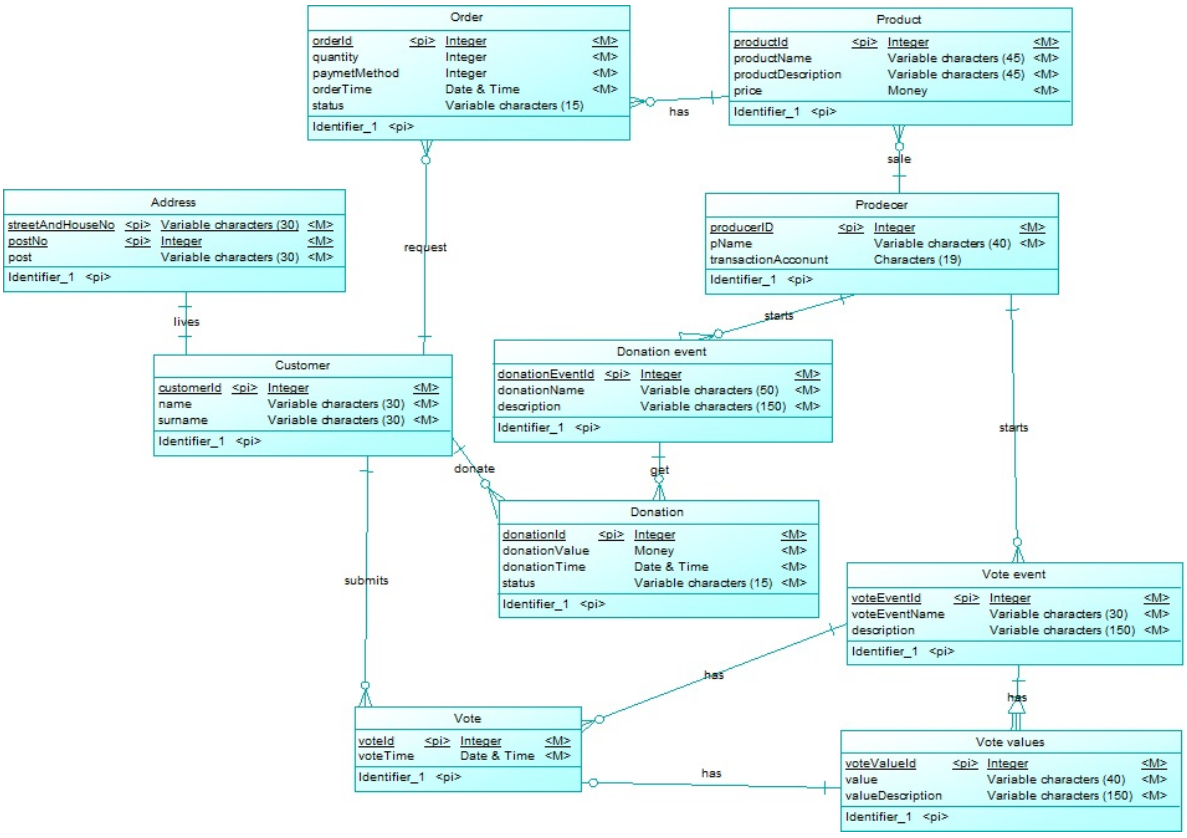
Kateri stranki je namenjena zahteve se ugotovi posredno preko podatkov o produktu, dogodku za glasovanje ali dogodka za humanitarno pomoč. Zahteva za nakup na primer vsebuje informacijo o produktu – njegov id. Podatke o produktu in lastniku produkta oziroma prodajalcu ima tudi aplikacija. Produkte, ki jih bo mogoče kupiti preko internetne televizije, mora aplikaciji dostaviti stranka. To omogoča ločen vmesnik. Poleg podatkov o produktih, dogodkih za glasovanje in darovanje v dobrodelne namene, ima aplikacija še podatke o stranki. Katere podatke vse hrani aplikacija in kakšna je njihova medsebojna odvisnost bomo prikazali v konceptualnem modelu podatkovne baze. Zahteva za naročilo se dodeli stranki, ki jo prodaja. To je zelo pomembno, saj ne želimo,da naročila neke stranke prejme druga stranka,

ki jih ne bi smela. Zahteva za nakup se tako shrani v podatkovno bazo. Ko bo stranka zahtevala nova naročila, se identificira, aplikacija pa ji odgovori s seznamom novih naročil.

6.2.1.2.2 Dostavljanje zahtev strankam Za dostavljanje zahtev strankam skrbi vmesnik 2 viden na sliki 6.1. Tudi ta vmesnik sestavljajo metode, ki se odzivajo na zahteve posamezne akcije, ki jih v tem primeru prožijo aplikacije strank.

V predlaganem sistemu bomo poskrbeli za dostavljanje naročil, rezultatov glasov in obvestil o darovanjih v najvišji meri avtomatiziramo. Ob prvem pomisleku se nam najbolj zdi, da zahteve ob sprejetju samodejno posredujemo stranki, vendar temu ni tako. Poglejmo si primer nakupnih naročil. Predpostavimo, da se na nekem kanalu oglašuje zelo zaželen produkt in v kratkem časovnem intervalu SaaS aplikacija sprejme več deset tisoč zahtev za nakup. Sistem na katerem teče SaaS aplikacija mora biti zmožen obdelati vse te zahteve. Če aplikacija deluje kot takojšni posrednik to velja tudi za sistem stranke, kar ni dobro. Sistem od strank ne sme zahtevati večjih sprememb njihovih sistemov. Alternativa temu je, da aplikacije strank izvajajo periodično povpraševanje po novih naročilih. Prototip trenutno omogoča omejeno število novih naročil na zahtevo, ravno zaradi obremenjevanja ostalih sistemov, omrežje. V predlaganem sistemu bo stranka preko nastavitvenega vmesnika nastavila, koliko novih naročil želi sprejeti ob eni zahtevi, vendar samo do neke meje. V primeru, da se prenašajo obsežna XML sporočila v odgovoru na zahtevo, se pojavijo nove težave. Na primer, kaj storiti, če sredi prenosa izgubimo povezavo?

Stranka se v svoji zahtevi za nova sporočila identificira. Na zahtevo se odzove določena metoda, na primer, če stranka zahteva nova naročila, se odzove metoda getNewOrders(). Metoda preveri ali je stranka registrirana v sistemu, pridobi nova naročila, pri čemer ohranja časovni vrstni red, in obvesti stranko, če ima še kakšno čakajočo naročilo. V tem primeru lahko stranka takoj pošlje novo zahtevo, razlog za to je prej omenjena omejitev o



Slika 6.2: Konceptualni model podatkovne baze

število naročil na zahtevo.

6.3 Arhitektura prototipne aplikacije

6.3.1 Konceptualni model podatkovne baze

Aplikacija vsa prejeta sporočila, to so naročila, glasove in obvestila o darovanju, shrani v podatkovno bazo z uporabo JPA. Iz konceptualnega modela podatkovne baze na sliki 6.2 je razvidno, katere podatke hranimo za delovanja osnovnih funkcionalnosti prototipa. Podatkovna baza je bila postavljena an strežniku MySQL 5.5.

6.3.2 Arhitektura aplikacije

Snovanje arhitekture aplikacije je zelo pomembno. Ustrezna aplikacijska arhitektura omogoči boljšo fleksibilnost, boljšo učinkovitost delovanja, razširljivost in lažje nadgrajevanje aplikacije.

Dobra praksa načrtovanja poslovnih aplikacij je ločitev poslovne logike od vmesnika, s tem omogočimo ponovno uporabo poslovne logike, zmanjšamo ponavljanje iste kode hkrati pa je lažje vzdrževanje (popravljanje) in nadgrajevanje kode. Pri načrtovanju prototipa aplikacije smo definirali dve arhitekturi, obe arhitekturi imata ločeno poslovno logiko od vmesnika, ki skrbi za sprejem sporočil.

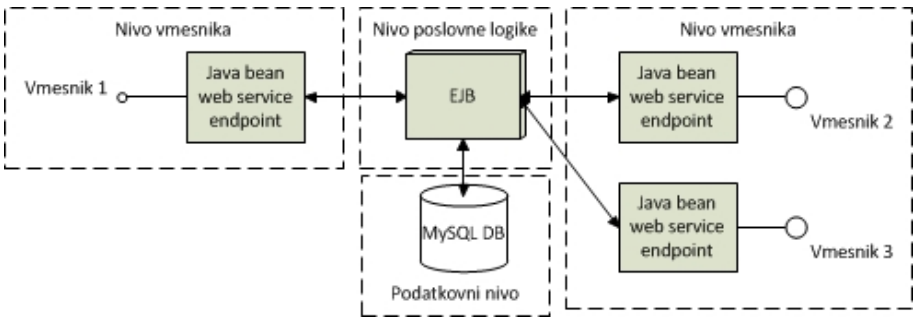
Kar moramo upoštevati pri gradnji arhitekture naše aplikacije je tudi to, da se nahaja v oblaku. Eden glavnih razlogov, da bo naša aplikacija umeščena v oblak, so lastnosti oblaka. Programska oprema kot storitev temelji na večkratno-najemni arhitekturi, čemur ustreza tudi naša aplikacija. Oblak omogoča podporo skalabilnosti (scalability), to pomeni, da se aplikacija prilagodi številu odjemalcev, ko le ta naraste. Ko število odjemalcev aplikacije, ki teče na enem strežniku, preseže zmogljivosti enega strežnika, se dinamično doda nova instance. Aplikacija potem hkrati teče na obeh strežnikih (horizontalno skaliranje), tudi na večih, če je to potrebno.

Razložili smo že delovanje sistema, kjer smo omenili, da bodo operacije gledalcem televizije na voljo v nekem časovnem okviru - na primer akcija bo možna v 15 minutnem časovnem okviru. Časovni okvir nam samodejno ustvari časovno obdobje z visokim prometom - t.i. peak hour, zato je ključno, da aplikacija živi v skalabilnem okolju. K boljši skalabilnosti aplikacije veliko pripomoremo z ustrezno arhitekturo aplikacije.

6.3.2.1 Sinhrona spletna storitev

Gradniki:

- Java bean web service endpoint - vmesnik spletne storitve, ki sprejema zahteve odjemalcev.



Slika 6.3: Osnovna arhitektura prototipne aplikacije

- EJB - Javansko strežniško zrno, ki izvaja poslovno logiko. Gre za zrna brez stanja, t.i. stateless javanska zrna.
- MySQL DB - Podatkovna baza.

Slika 6.3 predstavlja osnovno arhitekturo prototipne aplikacije. Prototip sestavljajo trije vmesniki, ki so implementirani kot spletne storitve, končne točke spletnih storitev na sliki predstavljajo gradniki "Java bean web service endpoint". Gre za javansko zrno, ki je izpostavljeno kot spletna storitev. Vmesniki skrbijo za sprejem zahtev in preverjanje pravilnosti vsebine zahtev. Če vsebina zahteve ni korektna bo vmesnik zahtevo zavrnil in o tem obvestil odjemalca. V primeru, da je zahteva ustrezna se izvede poslovna logika. Poslovno logiko izvede gradnik EJB. Na sliki je simbolično prikazan le en EJB gradnik, čeprav prototip sestavlja več EJB komponent, na primer: OrdersManagerEJB, ki izvaja vso poslovno kodo v povezavi z naročili; DonationsManagerEJB, poslovna logika za obdelavo obvestil o darovanju v dobrodelne namene in VotesManagerEJB, ki izvaja poslovno logiko v povezavi z glasovanjem. Metode EJB komponent preko JPA izvajajo operacije nad podatkovno bazo. Aplikacije ponudnikov internetne televizije uporabljajo Vmesnik 1 za dostavo zahtev uporabnikov.

Aplikacije ponudnikov internetne televizije uporabljajo Vmesnik 1 za dostavo zahtev uporabnikov.

Vmesnik 1 nudi naslednje metode:

```
public OrderResponse order(OrderRequest orderIn)
public OrderCancelResponse cancelOrder
(OrderCancelRequest orderCancelIn)
public VoteResponse vote(VoteRequest voteIn)
public DonationResponse donate(DonationRequest donationIn)
public ActionsResponse getPerformedActions
(ActionsRequest actionsIn)
```

Vse metode vrnejo odgovor, ki je odvisen od uspešnosti izvedbe programske kode. Vse metode izvedejo validacijo prejetega sporočila. Odgovor vsebuje status uspešnosti izvedbe akcije. Če se uporabnik odloči, da bo naročilo, oddan glas ali dobrodelni prispevek plačal preko položnice, ki mu jo izda ponudnik internetne storitve. V tem primeru odgovor vsebuje tudi številko transakcijskega računa prodajalce, v primeru da gre za naročilo, oziroma trr. stranke. Predlagamo, da uporabnikov ponudnik internetne televizije opravi transakcijo in jo preko položnice zaračuna uporabniku.

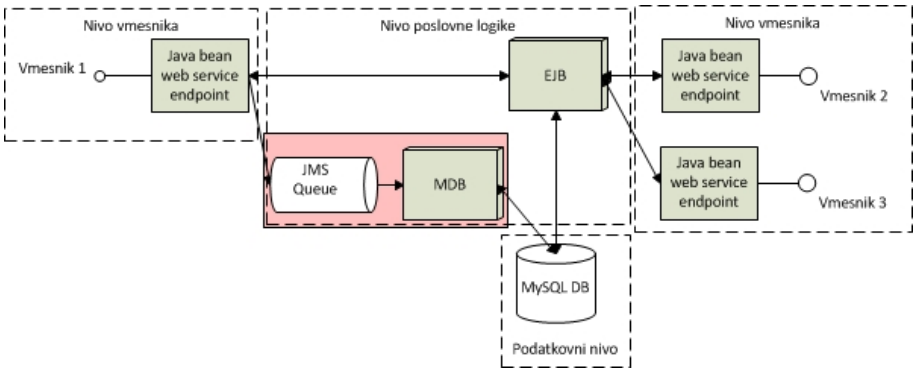
Vmesnika 2 in 3 pa sta namenjena strankam. Vmesnik 2 nudi operacije, s katerimi stranka pridobi nova naročila shranjena v podatkovni bazi. Vmesnik 3 strankam omogoča prijavo novih produktov, dogodkov za glasovanje in humanitarne dogodke. Te podatke mora stranka dostaviti tudi ponudnikom internetne televizije.

Metode vmesnika 2:

```
public GetNewOrdersResponse getNewOrders
(GetNewOrdersRequest newOrdersRequest)
public GetVoteResultsResponse getVoteResoults
(GetVoteResultsRequest voteResultsRequest)
public GetNewDonationsResponse getNewDonations
(GetNewDonationsRequest getNewDonationsRequest)
public UpdateOrderResponse updateOrderStatusShipped
(UpdateOrderRequest updateOrderRequest)
```

Vmesnik 3 v prototipu ni implementiran, predvideva pa naslednje operacije:

- Dodajanje novega produkta za nakup preko IPTV.



Slika 6.4: Razširjena arhitektura prototipne aplikacije

- Dodajanje novega dogodka za glasovanje.
- Dodajanje novega humanitarnega dogodka.

Arhitektura prikazana na sliki 6.3 predstavlja sinhrono spletno storitve. To pomeni, da odjemalec, ki je sprožil zahtevo, čaka na odgovor dokler zahteva ni obdelana, v tem primeru dokler zahteva ni shranjena v podatkovno bazo, ko se zahteva shrani se pošlje odgovor o uspešnosti izvedbe operacije.

6.3.2.2 Asinhrona spletna storitev

Arhitektura na sliki 6.4 je razširja arhitektura prikazana na sliki 6.3 Arhitektura 6.4 vsebuje dva nova gradnika:

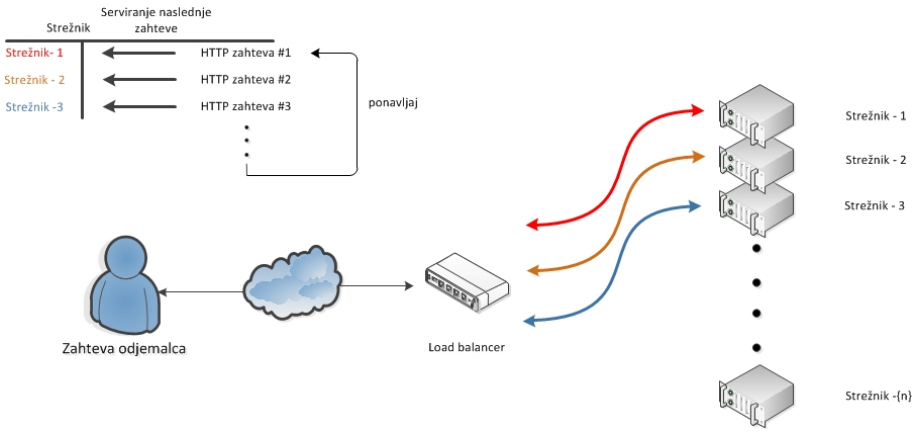
1. MDB - Sporočilno zrno, ki se uporablja za asinhrono sprejemanje sporočil iz JMS vrste. Prejeto sporočilo shrani v vrsto.
2. JMS vrsta (JMS Queue) - Sporočilna vrsta, kamor se shranjujejo prispela sporočila.

JMS vrsto in MDB je smiselno uporabiti za sprejem zahtev za naročilo produkta, glasa in obvestila o darovanju v dobrodelni namen. Asinhrona obdelava zahtev pride do izraza pri dolgotrajnih poslovnih operacij. Če uporabnik odda zahtevo za naročilo ne želi čakati na potrditev, da je sporočilo

shranjeno v podatkovno bazo. Arhitektura z uporabo JMS deluje tako, da vmesnik kličoči aplikaciji v primeru, da je zahteva korektna, takoj ko zahtevo posreduje JMS vrsti potrdi sprejem. Za obdelavo zahteve nato poskrbi komponenta MDB, ki sporočilo shrani v podatkovno bazo. Zahteva uporabnika je sprejeta šele takrat, ko se shrani v podatkovno bazo, saj jo šele takrat lahko prejme stranka. V tem primeru sistem potrdi samo prejem zahteve, kar pomeni, da mora sistem zagotoviti zapis v podatkovno bazo. To omogočijo transakcije, dokler sporočilno zrno metode, ki prevzema sporočila, ne izvede uspešno - metoda skrbi za zapis v podatkovno bazo, ne sme sporočila vzeti iz JMS vrste.

Uporaba JMS vrste omogoči šibko sklopljenost med komponentami aplikacije. Komponenta, ki v JMS vrsto vstavlja sporočila in komponenta, ki iz vrste odstranjuje oz. prejema sporočila in jih obdeluje, sta popolnoma ločeni. Komponenti lahko tečeta na ločenih sistemih, kar je pri naši aplikaciji zelo pomembno. Vmesnik spletne storitve lahko tako teče na ločenem strežniku kot MDB komponenta, ki sporočila obdeluje. Lahko imamo tudi več vzporedni vmesnikov (vmesnik se zaradi obremenjenosti paralelizira - skalabilnsot), ki lahko sporočila vstavljajo v isto JMS vrsto. Poleg tega imamo lahko več MDB komponent, ki obdelujejo sporočila. Vsaka MDB komponenta lahko teče na ločenem strežniku, narava JMS vrste pa bo poskrbela za izenačitev obremenitve posameznega naročnika (load balancing), saj je sporočilo dostavljeno samo enkrat.

Arhitektura na sliki 6.4 ravno tako omogoča, da komponente tečejo na ločenih strežnikih, komponente EJB morajo nuditi oddaljen vmesnik (remote interface), tako postanejo dostopni iz oddaljene aplikacije. Pri implementaciji smo uporabili EJB komponente z lokalnimi vmesniki, kar pomeni, da morata vmesnik spletne storitve in EJB komponente teči na istem strežniku. Paralelizacija v tem primeru je možna tako, da kompletno aplikacijo izvajamo na večih strežnikih med katere se zahteve dostavljajo preko komponente, ki opravlja izenačevanje obremenitve (load balancer). Razporejanje zahtev lahko potema po metodi Round Robin, kot je prikazano na sliki **??**.



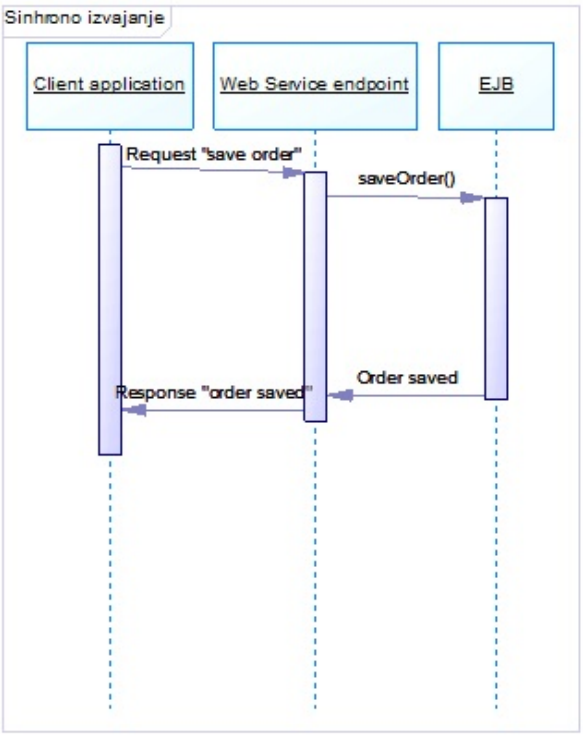
Slika 6.5: Uravnoteževanje obremenitve

6.3.2.2.1 Primerjava izvajanja klica po arhitekturah

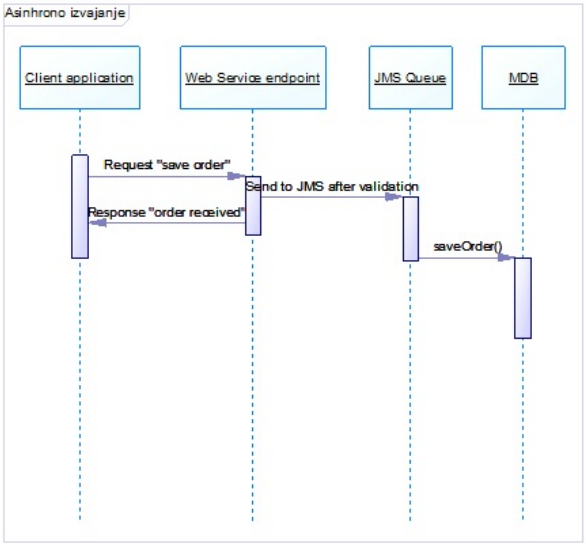
Ko primerjamo sliki 6.6 in 6.7 hitro vidimo bistveno razliko izvajanja. Na spodnji sliki vidimo, da odjemalec potrdilo o prejetju dobi preden je naročilo shranjeno v podatkovno bazo, med tem ko na zgornji sliki potrdilo dobi šele ko je izvajanje operacije shranjevanja končano. Tu je glavna prednost arhitektura z asinhronim pristopom, namreč odjemalna aplikacija ni blokirana in ne čaka na odziv. Odjemalna aplikacija lahko uporablja asinhrone klice spletne storitve in tako ni blokirana, vendar je prednost asinhronega pristopa v tem, da manj časa zaseda vmesnik spletne storitve (web service endpoint).

6.3.3 Aplikacija pošiljatelja zahtev - IPTV ponudnika

Aplikacija, ki jo implementira ponudnik internetne televizije, posreduje zahteve uporabnikov internetne televizije. Gre za odjemalca spletne storitve Vmesnik 1. Za testiranje prototipa smo razvili primer odjemalca spletne storitve Vmesnik 1, ki generira zahteve in jih posreduje SaaS aplikaciji.



Slika 6.6: Izvajanje sinhrone spletne storitve



Slika 6.7: Izvajanje asinhrone spletne storitve

6.3.4 Aplikacija stranke za sprejem naročil/rezultatov glasov/obvestil o darovanju

Dostavljanje sporočil deluje po principu povpraševanja. Aplikacija stranke periodično povprašuje po novih naročilih ali novih obvestilih o darovanju.

Tudi tu gre za odjemalca spletne storitve, le da v tem primeru kličemo operacije, ki jih nudi Vmesnik 2. Pridobivanje rezultatov glasovanja je dokaj enostavno, aplikacija mora v zahtevo vključiti le svoj id. Spletna storitev stranko identificira in vrne rezultate glasovanja. Pridobivanje rezultatov glasovanja je enkratna operacija, ko se glasovanje zaključi stranka zahteva rezultate. Drugače je pri naročilih in obvestilih o darovanju. V tem primeru zahteve za naročila in obvestila o darovanju nenehno prihajajo. Aplikacija mora tako spletno storitev periodično povpraševati po novih sporočilih.

Pri razvoju prototipa odjemalca smo uporabili časovno storitev (Timer Service) v EJB. Implementirali smo metodo, ki smo ji dodali anotacijo @Schedule in določili parametre. Anotacija @Schedule omogoča proženje metode s pomočjo časovnika. Časovnik tako omogoča enostavno implementiranje povpraševanja po novih naročilih, na primer vsako minuto.

Primer metode, ki jo proži časovnik

```
@Schedule(second="*/15", minute="*", hour="*")
        public void polling() {
                //String res = proxy.newOperation("test");
                Response<NewOperationResponse> response =
                proxy.newOperationAsync("test");
                while (!response.isDone()) {
                        // Do something while we wait.
                }
                NewOperationResponse res = null;
                try {
                        res = response.get();
                } catch (InterruptedException e) {
                        e.printStackTrace();
                } catch (ExecutionException e) {
```

```
                e.printStackTrace();
            }
            System.out.println("Odgovor : " + res.getOut());
        }
```

Koda zgornje metode se proži vsakih 15 sekund. Aplikacija za pridobivanje sporočil je lahko avtomatizirana, kot je opisano zgoraj. Implementacija avtomatizirane aplikacije je smiselna, ko ima prodajalec produkta svoj naročilni sistem, kar je v večini primerov. V nasprotnem primeru lahko stranka implementira spletni vmesnik, preko katerega stranka ročno proži zahtevo za nova naročila. Kako stranka implementira aplikacijo za pridobivanje novih sporočil je njena odločitev. V vsakem primeru uporabi vmesnik spletne storitve, ki omogoča dostop do strankinih sporočil.

Poglavje 7

Sklepne ugotovitve

V diplomski nalogi smo predstavili kako omogočimo dostop do komunikacijskih storitve aplikacijam, ki bi rade te storitve uporabile pri svoji implementaciji. Ugotovili smo, da za dostop do komunikacijskih storitev, ki se nahajajo znotraj infrastrukture nekega podjetja, potrebujemo rešitev, ker se te nahajajo znotraj varovanega omrežja, ki največkrat ni dostopno zunanjim uporabnikom. Primer take rešitve je PASS, ki zunanjim, največkrat SaaS storitvam omogoči uporabo komunikacijskih storitev, ki so ravno tako izpostavljene kot storitve z uporabo spletnih storitev. Primer CaaS aplikacije je aplikacija za merjenje občinstva internetne televizije. CaaS aplikacija je ponudila storitve za dostop do podatkov raznih statistik občinstva internetne televizije. CaaS aplikacija je podatke o gledanosti zbirala preko komunikacijskih storitev – spremlja število naročenih uporabnikov na kanal na robnem stikalu.

V praktičnem delu diplomskega dela smo se lotili izdelave prototipa sistema, ki bi uporabnikom internetne televizije ob gledanju razne vsebine omogočil akcije kot naročanje izdelka, ki se v nekem trenutku oglašuje, izvedbo oddaje glasu, ko je uporabnik pozvan h glasovanju in izvedbo darovanja v dobrodelni namen s pritiskom na gumb daljinskega upravljalnika. Cilj sistema je, da prodajalcem in ostalim strankam dostavi naročila, rezultate glasovanja in obvestila o darovanju. Največ pozornosti smo namenili razvoju

SaaS aplikacije, ki bi skrbela za dostavo zahtev teh akcij. Predlagali smo prototip sistema in razložili njegovo delovanje. Veliko pozornosti smo namenili tudi načrtovanju arhitekture SaaS aplikacije na platformi Java EE.

Da mora biti arhitektura take aplikacije ustrezno načrtovana zahteva sam način delovanje in narava funkcionalnosti, ki jih podpira. Prototip aplikacije smo razvili z uporabo Javanskih tehnologij: JAX-WS, EJB, MDB in JMS, ki omogočajo razvoj naprednejših poslovnih aplikacij. Prototipna SaaS aplikacija podpira le osnovne funkcionalnosti sistema, za komercialno uporabo so potrebne dodatne razširitve. V delu se nismo poglobili v to, kakšna mora biti informacijska rešitev na strani ponudnika internetne televizije, podali smo zgolj predlog odjemalne aplikacije, ki SaaS rešitvi posreduje naročila. Ker je rešitev sestavljena iz večih informacijskih sistemov, ki so med sabo povezani je potrebno pomisliti tudi na sinhronizacijo podatkovnih baz. Ponudniku internetne televizije je potrebno dostaviti podatke o tem, kdaj naj omogoči akcijo. Pri akciji "naroči"je potreben točen čas, kdaj se oglas vrti in koliko časa traja. V delu smo predpostavili, da te podatke ponudnik internetne televizije že ima, vendar za praktično uporaben sistem je potrebno podati rešitev tudi za to. V prototipu smo že predlagali vmesnik, ki bi strankam omogočil, da ponudniku internetne televizije naročijo novo akcijo, vendar teh podatkov nismo dostavili ponudniku internetne televizije. Potrebna je CaaS rešitev na strani ponudnika internetne televizije, kar bi SaaS aplikaciji omogočilo posredovanje potrebnih podatkov ponudnikom, katere si je zaželela stranka.

Možnosti, ki jih omogoča SaaS aplikacija, je veliko. Strankam - prodajalcem, naročnikom anket itd. lahko ponudimo podatke o tem, kdaj je najbolj smiselno oglaševati nek produkt, kdaj je največ nakupov, kakšen je odziv na nek produkt itn. Pri vsem tem moramo biti pazljivi pri varnosti. Podatke posameznega gledalca ne smemo deliti brez njegovega dovoljenja. Ker se SaaS aplikacija nahaja v oblaku in je namenjena uporabi večim strankam, pri tem moramo biti pazljivi, do katerih podatkov ima neka stranka dostop, na primer, če naročilo ni namenjeno določeni stranki ta do njega ne sme imeti

dostopa.

Razvoj takšne SaaS aplikacije zahteva, da je ta skalabilna, kar vpliva na arhitekturo sistema. Naslednja stopnja razvoja arhitekture bi bila, kako pohitriti vmesnike spletnih storitev, na primer z uporabo predpomnjenja.

Literatura

[1] J. Hofstader (2007), "Communications as a Service" Dostopno na: http://msdn.microsoft.com/en-us/library/bb896003.aspx

[2] Feng Liu Li Li, Wu Chou (2009), "Communications Enablement of Software-as-a-Service (SaaS) Applications" Dostopno na: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5425403

[3] Li-Chia Yeh Telecommuication Labs., Chunghwa Telecom, Taoyuan, Taiwan Ching-Sheu Wang ; Chi-Yi Lin ; Jia-Siang Chen (2011) ,"An innovative application over communications-asa-service: Network-based multicast IPTV audience measurement" Dostopno na: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6077040

[4] Sun Microsystems , "Web Service Interaction Architectures" Dostopno na: http://java.sun.com/blueprints/webservices/using/webservbp3.html

[5] Ecma International ,"Computer Supported Telecommunications Applications" Dostopno na: http://www.ecma-international.org/activities/Communications/TG11/cstaIII.htm

[6] S. Seely, "Understanding WS-Security" Dostopno na: http://msdn.microsoft.com/en-us/library/ms977327.aspx

[7] "JAX-WS 2.2 Runtime Library" Dostopno na: http://jax-ws.java.net/nonav/jaxws-api/2.2/index.html

[8] "Getting Started with Java Message Service (JMS)" Dostopno na: http://java.sun.com/developer/technicalArticles/Ecommerce/jms/

[9] "Web Services Description Language (WSDL) Version 1.2" Dostopno na: http://www.w3.org/TR/2003/WD-wsdl12-20030303/