УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Борис Петељ

Механизам за обавештавање корисника о догађајима у ИоТ систему

МАСТЕР РАД

Нови Сад, 2016.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Дипломски – мастер рад	
Аутор, **АУ**:		**Борис Петељ**	
Ментор, **МН**:		**доц. др Иштван Пап**	
Наслов рада, **НР**:		**Механизам за обавештавање корисника о догађајима у ИоТ систему**	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		**2016**	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		**7/42/9/4/6**	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		**ИоТ, обавештење, кућна аутоматизација**	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		**У раду описана је реализација механизма за обавештавање корисника у системима паметне куће. Обавештења подразумевају достављање информација на мобилне уређаје коришћењем доступних метода као што су размене порука, е-поште, и метода иницираних са послужиоца мобилних платформи.**	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	**доц. др Милан Лукић**	
	Члан:	**доц. др Иван Каштелан**	Потпис ментора
	Члан, ментор:	**доц. др. Иштван Пап**	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		**Boris Petelj**	
Mentor, **MN**:		**PhD. Ištvan Pap**	
Title, **TI**:		**Mechanism for notifying users in event based IoT systems**	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		**2016**	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		**7/42/9/4/6**	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		**IoT, notification, home automation**	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		**This paper covers implementation of mechanism for notifying users in home automation system. Notifications imply delivering messages to mobile devices using available methods such as SMS, e-mail and push notification mechanisms.**	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	**PhD. Milan Lukić**	
	Member:	**PhD. Ivan Kaštelan**	Menthor's sign
	Member, Mentor:	**PhD. Ištvan Pap**	

Zahvalnost

Hvala svim osobama koje su mi pružile stručnu, moralnu ili duhovnu podršku u prevazilaženju prepreka do izrade ovog rada.

Posebno se zahvaljujem svojoj porodici na stalnoj podršci, strpljenju i veri.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

HTTP -Komunikacioni protokol za razmenu podataka (eng. **H**yper **T**ext **T**ransfer **P**rotocol)

JSON -Sintaksa za skladištenje i razmenu podataka – (eng. **JavaScript O**bject **N**otation)

BLE -Protokol za razmenu podataka - **B**luetooth **L**ow **E**nergy

APN -Posrednički poslužilac za dostavljanje *push* obaveštenja na iOS uređaje (eng. **A**pple **P**ush **N**otification)

GCM -Posrednički poslužilac za dostavljanje *push* obaveštenja na Android uređaje (eng. **G**oogle **C**loud **M**essaging)

IoT -Pojam uređaja povezanih na Internet (eng. **I**nternet **O**f **T**hings)

MQTT -Protokol za razmenu podataka po principu objave/pretplate (eng. **M**essage **Q**ueuing **T**elemetry **T**ransport)

NPM -Servis za manipulaciju JavaScript biblioteka (eng. **N**ode **P**ackage **M**anager)

1. Uvod

Povezivanje uređaja na Internet, ili IoT koncept, sve više dobija na značaju u ovoj tehnološkoj eri. Razlog za tako masovnu ekspanziju ovih uređaja pre svega jeste njihova potražnja na tržištu usled primene u domenu kućne automatizacije, ali i pojava veoma jeftinih hardverskih komponenti. Konstantan razvoj IoT tehnologije, koja podrazumeva povezivanje raznih perifernih uređaja kao što su senzori, aktuatori, prenosni uređaji, itd. u jedinstvenu mrežu sposobnu za prikupljanje i obradu podataka, u korak prati i razvoj sistema za kontrolu i personalizaciju ovih mreža.

Konkurentnih kompanija specijalizovanih za proizvodnju perifernih uređaja je sve više, iako univerzalni protokol za razmenu informacija ne postoji, već se koristi niz standarda. Najčešći su svakako Zigbee, Zwave, BLE, WiFi, 6LoWPAN, ili drugi vlasnički protokoli, međutim iako ih je malo, problem koji se javlja jeste njihova međusobna nekompatibilnost. Drugim rečima, ne postoji način za direktnom razmenom informacija između npr. ZigBee i Zwave perifernog uređaja. Sa druge strane odnos zastupljenosti ovih protokola u perifernim uređajima je takav da oslanjanje samo na neki od njih nije dovoljno za potrebnu pokrivenost perifernih uređaja u IoT sistemima.

Radi rešavanja pomenutog problema, sistemi „pametne“ kuće, uvode pojam centralnog uređaja (ili češće „gateway“) čija je uloga da obezbedi uniforman pristup perifernim uređajima različitih protokola, ali i pružanje podrške za dodatnom logikom na višem apstraktnom nivou unutar sistema. Stoga sistem kućne automatizacije predstavlja centralizovan sistem gde periferne jedinice različitih protokola čine jedinstvenu mrežu posredstvom „gateway“ uređaja.

Korisnička kontrola ovih sistema omogućena je korišćenjem mobilnih uređaja, tableta, ili zidnih računara koji u lokalnoj mreži direktno komuniciraju sa centralnim uređajem (preko mrežnih rutera), ili u slučaju kontrole izvan domašaja lokalne mreže, posredstvom *cloud* servisa. Stoga se sistemi „pametnih kuća“ prostiru od samih perifernih, preko centralnih uređaja, servisa

u oblaku (cloud rešenja) do korisničkih aplikacija u koje spadaju Web, aplikacije za rasprostranjene platforme mobilnih uređaja, desktop aplikacije, ili kontrole preko namenskih ugradnih jedinica.

1.1 Sigurnost i obaveštenja

Glavne karakteristike ovih sistema su komfor, ekonomičnost, automatizacija i sigurnost međutim nezaobilazni termin koji se po pravilu vezuje uz pojam „pametne“ kuće, obzirom na koncept iz kog proističe (IoT), jeste pristup preko Interneta ili udaljeni pristup.

Udaljena kontrola, a ujedno i nadgledanje, nudi mogućnost za pristupom ovoj mreži povezanih uređaja izvan objekta korišćenjem sigurne Internet veze. Podrazumeva komunikaciju mobilnih klijenata sa servisima u „oblaku“ koji su konstantno dostupni, obzirom da direktan pristup mrežnom ruteru izvan objekta najčešće nije moguć.

Sa druge strane, pitanje sigurnosti sistema tiče se najpre konkretnih senzora i aktuatora raspoređenih po objektu, a onda i logike za reagovanjem na detekciju anomalija, koja se definiše od strane samih korisnika sistema. Iako moderni sistemi za kontrolu ovih mreža svakako poseduju autonomnu logiku koja samostalno reaguje na ove događaje, svrha udaljenog pristupa i nadgledanja podrazumeva i mogućnost notificiranja korisnika o stanju objekta pri definisanim ponašanjima (detekcijama).

Veza između ovih funkcionalnosti, uzrokuje obaveznu podršku za slanjem obaveštenja korisnicima usled događaja vanrednih situacija kao što su detekcija pokreta, poplava, pojava dima ili slično u gotovo svim uslovima rada, pri čemu i sam korisnik dolazi u mogućnost za reagovanjem na konkretnu situaciju.

1.2 Mehanizmi za dostavljanje obaveštenja

Obzirom da se u slučaju postojećeg rešenja kontrole „pametnog“ objekta pod nazivom OBLO koriste isključivo mobilne klijentske aplikacije za kontrolu, postoji nekoliko varijanti za slanjem obaveštenja proteklih od centralnog uređaja, i to:

- Slanjem SMS poruke,
- Slanjem E-mail poruke,
- Korišćenjem mehanizma „pretplate/objavljivanja“ pod okvirom MQTT (Message Queuing Telemetry Transport) koji je korišćen kao primarni način za razmenu poruka u okviru OBLO sistema,
- Upotrebom *push* obaveštenja, ili „serverskih push“ obaveštenja, tj. slanje informacija na mobilne klijente bez njihovog prethodnog zahteva.

1.3 Problem dostave obaveštenja

Ovaj rad adresira problem pravovremenog i pouzdanog načina dostave obaveštenja korisnicima korišćenjem pomenutih metoda. Svaka od pomenutih metoda slanja obaveštenja ima svoje prednosti i mane, ali i ulogu u sistemu automatizacije kuće, međutim zaključak je da se *push* notifikacije zbog prirode funkcionisanja najbolje uklapaju u mehanizam za dostavljanje vremenski senzitivnih obaveštenja. Stoga, suština rada jeste eksperimentalna potvrda koncepta kroz integraciju mehanizma *push* obaveštenja u postojeće rešenje pametne kuće pod nazivom OBLO.

1.4 Integracija u OBLO sistem

U ovom radu detaljno je opisan koncept dostave obaveštenja na svim nivoima (centralni uređaj, servisi u oblaku, kao i aplikacije mobilnih uređaja) koji su uključeni u reagovanje na detekciju vanrednog stanja sistema.

Rešenje podrazumeva implementaciju mehanizma *push* obaveštenja na *cloud* platformi uz komunikaciju sa poslužiocima treće strane kao što su *Google Cloud Messaging* (*GCM*) za Android klijentske aplikacije, kao i *Apple Push Notification* (*APN*) za *iOS* klijentske aplikacije.

Takođe pokriveni su problemi koji se javljaju zbog prirode funkcionisanja ovog mehanizma, kao i načini za njihovo rešenje.

Rešenje je realizovano kao proširenje postojećeg servisa u oblaku pod nazivom Stratus2, pisanog u programskom jeziku JavaScript i okviru za razvoj poslužioca, Node.js.

2. Teorijske osnove

Imajući u vidu da rešenje predstavlja unapređenje postojećeg sistema za kontrolu IoT uređaja, u ovom poglavlju biće ukratko opisane korišćene tehnologije, kao i osnovni blokovi sa vezama i protokolima unutar projekta.

Razumevanje principa komunikacije u ovom sistemu neophodno je zbog uočavanja prednosti i mana svakog od navedenih načina za slanje obaveštenja.

2.1 OBLO, sistem "pametne" kuće

2.1.1 Centralni uređaj

Namena svake periferne jedinice (uređaji raspoređeni po objektu) najčešće je usko specificirana najpre zbog veće slobode raspoređivanja ali i jednostavnosti njihove izrade.

Uređaji sa npr. funkcionalnošću detekcije kontakta (senzori vrata/prozora) najčešće poseduju samo tu funkciju, te se svrstavaju u senzore. Sa druge strane, čak i ako uređaj poseduje skup funkcionalnosti one se po pravilu mogu kontrolisati ili očitavati nezavisno. To uzrokuje velik broj jednostavnih uređaja specificiranih za merenje (senzori) ili delovanje (aktuatori) u skladu sa njihovom namenom. Problem koji ostaje jeste reagovanje jednog uređaja u zavisnosti od stanja drugih, povrh različitih protokola komunikacije koje ova dva uređaja potencijalno koriste.

Obzirom da sistem pametne kuće najpre podrazumeva grupisanje IoT periferija u jedinstvenu mrežu, uređaje je potrebno posmatrati na višem apstraktnom nivou, odnosno potrebno je generalizovati metode i omogućiti aplikativan pristup njihovim funkcionalnostima. Da bi se informacije u ovakvom sistemu prebrodile prepreku različitih protokola, razmena podataka oslanja se na centralni uređaj, koji je u stanju da komunicira putem svih podržanih protokola, ali i da periferne uređaje grupiše po njihovoj funkcionalnosti.

Pored apstrakcije pristupa uređajima, logika raspoređivanja uređaja po zonama, tajmera, scena ili definisanja ponašanja i konfiguracije sistema, takođe su smeštene u ovom delu sistema "pametne" kuće.

2.1.2 **Servisi u *oblaku***

Centralni uređaji najčešće su namenski, relativno malih gabarita i bez ekrana za interakciju sa korisnicima, te je korisnička sprega za kontrolu sistema morala biti izmeštena na druge uređaje, odnosno mobilne ili klijentske uređaje. Veza između centralnog i klijentskih uređaja po pravilu je bežična i u svom osnovnom obliku podrazumeva povezivanje u okviru lokalne mreže posredstvom mrežnih rutera.

Međutim, u scenariju gde se klijentski uređaj nalazi izvan lokalne mreže, mora postojati drugi način za pristup centralnom uređaju. Za tu namenu uvodi se "okruženje u oblaku" ili *cloud* rešenje koje predstavlja instancu poslužioca konstantno dostupnu na Internet-u povezanu sa centralinim uređajem. Međusobno nezavisni centralni uređaji različitih korisnika koriste isto *cloud* okruženje, koje vodi računa o propagiranju informacija između klijenata i odgovarajućih centralnih uređaja.

Pored funkcije kontrole, servisi u "oblaku" služe i za održavanje centralnih uređaja od strane administratora OBLO sistema.

2.1.3 **Klijentske aplikacije**

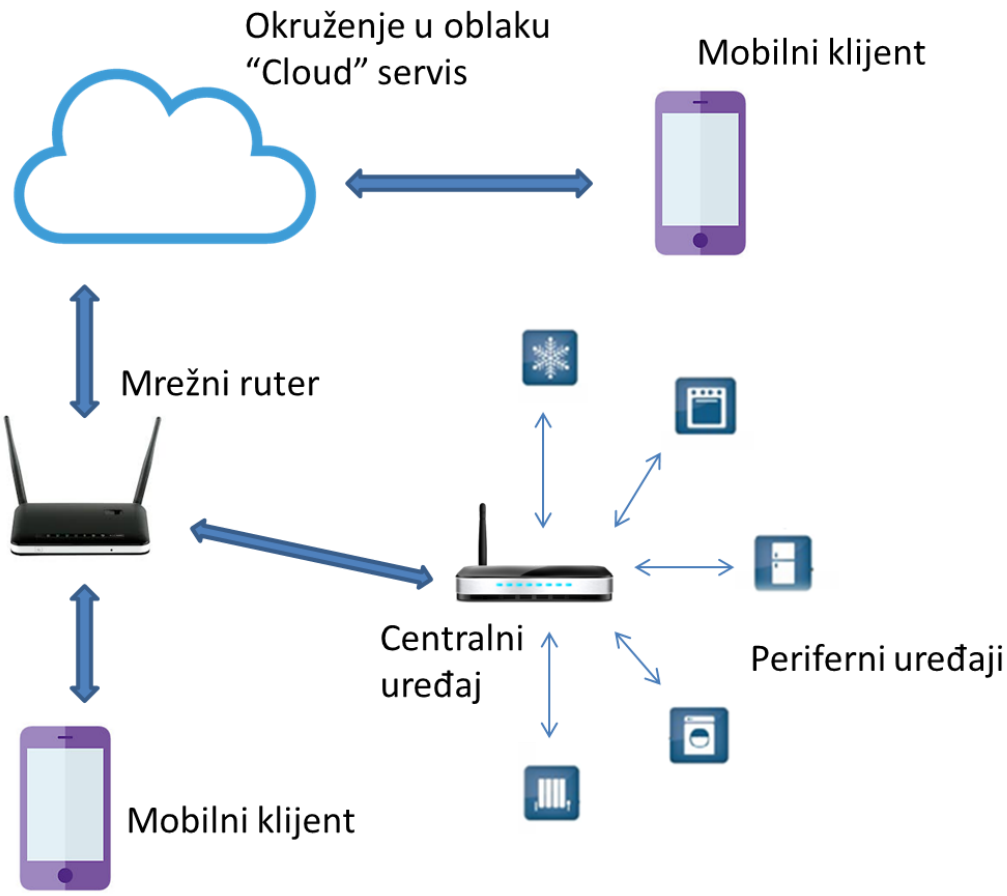
Sa korisničke strane, upotreba sistema svodi se na interakciju sa mobilnom aplikacijom koja se nakon pokretanja povezuje sa centralnim uređajem (posredstvom *cloud* servisa ili ne) radi dobavljanja osnovnih informacija o sistemu. Ove informacije sadrže uređaje povezane u mrežu kojima je tada moguće manipulisati individualno ili preko neke od dostupnih opcija kao što je grupna kontrola.

Korisnička sprega za korišćenje i konfiguracuju svih mogućnosti centralnog uređaja dostupna je kroz ovu softversku komponentu koja predstavlja primaran način korišćenja sistema od strane korisnika. U projektu, pokrivene su Android i iOS platforme mobilnih klijenata.

2.1.4 **Veza klijent-*cloud*-centralni uređaj**

Tok rada pri udaljenom pristupu mreži najpre podrazumeva povezivanje centralnih uređaja sa *cloud* platformom, nakon čega je uređaj evidentiran kao dostupan. Korisnik tada kroz klijentsku aplikaciju, a posredstvom *cloud* rešenja može da pristupi svom dodeljenom centralnom uređaju. Dodeljivanje je omogućeno na osnovu pristupnih prava, odnosno korisničkog naloga koji se kreira pri inicijalnoj konfiguraciji sistema. Komunikacija od strane centralnog uređaja, a i klijentske aplikacije prema *cloud* platformi, bazirana je na MQTT

protokolu po principu “pretplati se/objavi” zbog svoje arhitekture, načina prosleđivanja poruka i neophodnih resursa radi operabilnosti [1]. Slika 1 prikazuje vezu osnovnih blokova sistema za kućnu automatizaciju “OBLO”.



Slika 1: Sistem pametne kuće OBLO

2.1.5 **Servisi i scene unutar OBLO-a**

Mogućnosti centralnog uređaja bitne za temu obaveštenja korisnika jesu funkcionalnosti servisa i scena unutar OBLO sistema, iz razloga što predstavljaju mehanizam za generisanje ovih obaveštenja shodno sa korisničkom konfiguracijom sistema.

Pojam servisa podrazumeva grupisanje perifernih IoT uređaja po njihovoj funkcionalnosti nasuprot grupisanju po korišćenom protokolu za komunikaciju, dok scene predstavljaju mehanizam za utvrđivanjem ponašanja sistema u zavisnosti od svojstava ovih servisa [2]. Jednostavan primer konfiguracije scene bio bi: „upali svetlo na detektovan pomeraj“, gde senzor (pomeraja) i aktuator (prekidač) mogu biti uređaji koji koriste različite protokole za komunikaciju sa centralnim uređajem. Bitna napomena je da aktuatori na određeni događaj kao što je „detektovan pomeraj“ ne moraju biti samo fizički uređaji u sistemu, već i određene akcije sistema između kojih su i slanje obaveštenja korisniku. Pored ovih, korisnički utvrđenih

obaveštenja, sistem interno koristi mehanizam razmene standardnih poruka na analogan način, pomoću kojeg je moguće dobiti i uvid u ponašanje sistema u istoriji („Log“ obaveštenja).

Jasno je da se i jednostavni periferni uređaji, kao što su kontakt senzori vrata i prozora, mogu iskoristiti u svrhe sigurnosti „pametne“ kuće ručnim konfigurisanjem scena, te je neophodno obezbediti sistem pouzdanim načinom slanja obaveštenja na korisničku aplikaciju, posebno ukoliko ove scene detektuju neki oblik vanredne situacije.

2.2 Dostava obaveštenja

Dostavljanje obaveštenja na klijentski uređaj može se postići na više načina, međutim efekat prikaza, pouzdanost, stopa registrovanja od strane korisnika i slično čini bitnu razliku a zavisi od korišćene tehnologije.

Email, SMS i *push* obaveštenja osnovni su načini za komunikaciju sa korisnicima ukoliko su ciljni uređaji mobilne platforme. Međutim, imajući u vidu da su klijentske aplikacije takođe razvijene u okviru projekta, moguće je iskoristiti i protokol upotrebljen za standardnu razmenu poruka u sistemu OBLO (MQTT).

Svaki od navedenih načina ima prednosti, i nalazi svoju primenu u sistemu “pametnih” kuća te sledi opis i upotreba svakog od navedenih metoda.

2.2.1 Email obaveštenja

Iako predstavljaju jedan od najpouzdanijih metoda slanja poruka, pogotovo zbog mogućnosti konfiguracije skladištenja poruka na poslužiocima (*SMTP*), ovaj način nije najpogodniji za obaveštenja usled detekcije vanrednih situacija. Po navodima kompanija za analizu metrike digitalnog sadržaja, prosečno vreme otvaranja email poruke jeste 6,4h [3], dok je stopa otvorenih (pročitanih) email poruka oko 33% [4]. Stoga se primarna upotreba email poruka svodi na vremenski ne-senzitivna obaveštenja kao što su uspešne registracije, potvrde, narudžbe i slično. Takođe email predstavlja odličan izvor podataka kojima je potrebno pristupiti i obraditi u određenom vremenu nakon primanja poruke.

Email kao mehanizam razmene poruka podrazumeva Internet konekciju sa email poslužiocima, i najčešće koristi mehanizam periodičnog propitivanja poslužioca o pristiglim porukama. Ovaj način uvodi vreme kašnjenja dostave poruke u zavisnosti od konfiguracije periode propitivanja. Međutim, moderni email poslužioci koriste i mehanizam pozadinskog *push* obaveštenja (opisano kasnije) radi obaveštavanja klijenata o pristigloj poruci bez periodičnog propitivanja. Na ovaj način izbegava se period čekanja između provere pristiglih poruka.

2.2.2 SMS poruke

Značajna prednost SMS poruka, koja ovu metodu izdvaja od ostalih je mogućnost rada bez Internet konekcije, kao i mogućnost slanja obaveštenja na različite platforme. Takođe, statistika po navodima kompanije "*Mobile Squared*" glasi da se 90% SMS poruka otvori u roku od 3 minuta od momenta slanja [5]. Međutim, imajući u vidu da korisnici OBLO sistema ne moraju uvek da poseduju i GSM broj uz kreiran nalog, ova metoda, nije dovoljna za sigurnu dostavu obaveštenja. Svakako, kako predstavlja jedan od najpouzdanijih metoda slanja obaveštenja njena podrška u sistemima kućne automatizacije je neizostavna.

2.2.3 MQTT

MQTT je protokol za razmenu poruka baziran na modelu "pretplati se/objavi". Neophodna komponenta u ovom mehanizmu je posrednik u razmeni poruka među klijentima, odnosno broker. "Pretplati se/objavi" mehanizam omogućava slanje poruka po principu jedan-ka-više, te logički razdvaja pošiljaoce od primaoca. Takođe, podržava slanje obaveštenja o nestandardnom raskidu veze sa korisnicima.

Jedinice koji učestvuju u komunikaciji su izdavači, pretplatnici i posrednici. Izdavači šalju poruke na određene naslove, a posrednik je zadužen za isporuku poruka zainteresovanim pretplatnicima. Obzirom na ograničene memorijski/procesorske resurse centralnih uređaja, kao i baterijska napajanja mobilnih klijentskih uređaja, MQTT protokol određen je kao najpogodniji od sličnih M2M protokola (AMQP, CoAP, XMPP) za postojeći OBLO sistem, te se koristi se za standardnu razmenu informacija u celokupnom sistemu koja podrazumeva komunikaciju između centralnih uređaja, *cloud* servisa i mobilnih aplikacija [1].

Problem koji se javlja pri upotrebi ovog mehanizma za slanje bitnih informacija na mobilne klijente kao što su vanredna obaveštenja je potreba za stalno aktivnim pozadinskim procesom ili trenutno aktivna aplikacija na samom klijentu da bi poruka bila dostavljena. Razlog je što se MQTT protokol oslanja na permanentnu TCP vezu. Klijentske aplikacije, korišćenjem MQTT klijent biblioteke, nakon pokretanja iniciraju uspostavljanje veze sa brokerom nakon čega je nadalje moguće objavljivati i primati poruke od značaja.

2.2.4 *Push* obaveštenja

Za razliku od "pull" obaveštenja, kod kojih klijent periodično mora zahtevati informacije od poslužioca, *push* obaveštenja su inicirana od strane poslužioca. Tipično, krajnji korisnik najpre mora konfigurisati mobilni uređaj tj. omogućiti primanje ovih obaveštenja za određenu aplikaciju, što se najčešće odvija u toku instalacionog procesa.

Najvažnija karakteristika, a i prednost *push* obaveštenja jeste što ova tehnologija ne zahteva od klijentskog uređaja bilo kakav specijalni pozadinski servis ili trenutno aktivnu aplikaciju. To znači da obaveštenja bivaju dostavljena čak i kada je ekran uređaja zaključan, ili aplikacija na koju se obaveštenje odnosi trenutno nije pokrenuta. Naravno, Internet konekcija je obavezna zbog komunikacije sa *cloud* servisima treće strane, neophodnim za realizaciju ovog vida komunikacije. Ukoliko je reč o dostavljanju obaveštenja na mobilne uređaje, u okviru samih operativnih sistema uređaja integrisana je komunikacija sa odgovarajućim posredničkim poslužiocima odnosno *GCM* za Android i *APN* za iOS operativne sisteme.

Push obaveštenja predstavljaju način za direktnu komunikaciju sa korisnikom. Obaveštenja završavaju kao "popup" poruke na klijentskom uređaju. Ne prolaze kroz potencijalne filtere i ne završavaju u datotekama primljenih poruka kao što je slučaj sa email porukama. Takođe, reakcija korisnika na registrovano obaveštenje je podesiva, te najpre vodi korisnika u asociranu aplikaciju gde se u slučaju pametne kuće mogu preduzeti dodatne akcije na događaj.

2.3 Node.js platforma

U IoT sistemima, neophodno je obezbediti logiku sposobnu za reagovanjem na događaje u sistemu koji su po svojoj prirodi asinhroni. U okviru OBLO projekta, *cloud* servis zasnovan je na Node.js platformi baziranoj na JavaScript programskom jeziku iz sledećih razloga:

- Model programiranja dizajniran upravo za potrebe poslužioca i komunikacije sa periferijama
    - Asinhroni programski model vođen događajima (eng. *event driven*), što je upravo slučaj sa IoT sistemima gde poslužilac većinu vremena provede čekajući na događaj od strane centralnog ili mobilnog klijentskog uređaja.
- Podrška za kompletno (eng. *end-to-end*) rešenje od perifernih uređaja, preko centralnog uređaja, mobilnih platformi, internet pretraživača i poslužioca.
    - OBLO rešenje takođe podrazumeva i Web stranice za kontrolu (*Selfcare*) i administrativan pristup (*BackOffice*) koji po prirodi koriste JavaScript programski jezik i *JSON* kao model podataka. Node.js nudi uniforman pristup pri rešenju pozadinske logike (eng. *backend*), delu zaduženom za korisničku interakciju (eng. *frontend*) kao i skladištenju podataka.
- Neblokirajuće operacije sa I/O periferijama.
    - Mogućnost nastavka obrade drugih zahteva ukoliko dođe do blokirajuće operacije kao što je pristup bazi sa trenutnim, što veći deo skupa zahteva podrazumeva.
- Podrška za komunikacijom koristeći razne protokole.

  - Primarno se oslanja na HTTP ali nije njime i ograničena. U daljem radu (poglavlje 4.1) opisana je arhitektura *cloud* servera po principu mikro servisa. Jednostavnost realizacije projekta u Node.js platformi pod ovakvim modelom je jedna od bitnijih stavki razmotrenih pri samom odabiru platforme.
- Velik ekosistem i sistem za upravljanje modulima.
  - Oko 250,000 modula dostupnih preko NPM (Node package manager) servisa u okviru kojih su i više paketa koji realizuju MQTT klijenta, kao i komunikaciju sa potrebnim posredničkim poslužiocima.
- Mogućnost paralelizacije poslova.
- Široka i stabilna podrška
  - Fundiran i podržavan od strane većih kompanija kao što su Walmart, Microsoft, Yahoo, Paypal, Voxer i drugi [6].

3. Koncept rešenja

Iz prethodno navedenog poređenja metoda slanja obaveštenja na mobilne klijente, zaključuje se da je mehanizam *push* notificiranja najpogodniji za potrebe slanja vremenski senzitivnih obaveštenja. Integracija ove vrste razmene poruka u postojeći OBLO sistem, omogućila bi slanje obaveštenja na mobilne klijente ukoliko aplikacija nije trenutno aktivna, izbegavajući potencijalne filtere (u slučaju email-a), i potrebu za periodičnim propitivanjem poslužioca.

U nastavku rad je fokusiran na detaljnijom razradom principa rada i metoda integracije *push* obaveštenja u OBLO sistem.

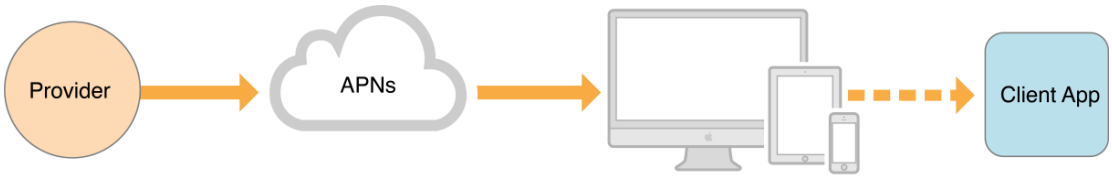
3.1 Slanje *push* obaveštenja

Postoje tri glavna činioca uključena u dostavljanje *push* obaveštenja:

- Vlasnički poslužilac ili snabdevač
  - Zadužen za odlučivanjem kada i kome obaveštenje treba poslati.
  - Mora podržati način za komunikacijom sa uređajima za koje je obaveštenje namenjeno.
  - Generiše i inicira slanje obaveštenja.
- Klijentska aplikacija
  - Specifična za konkretan OS uređaja.
  - Instalirana na mobilni uređaj koja prima *push* obaveštenja.
- Posrednički poslužilac ili poslužilac obaveštenja (Notification server)
  - Poslužilac koji konkretno obaveštenje prosleđuje mobilnim uređajima.

Može se primetiti da zarad dostavljanja obaveštenja, sa klijentskim uređajima komuniciraju i vlasnički, i posrednički poslužilac. Naime, vlasnički poslužilac koji je zadužen za generisanje, obaveštenje ne može i poslati direktno na klijentski uređaj. Razlog za to je što slanje

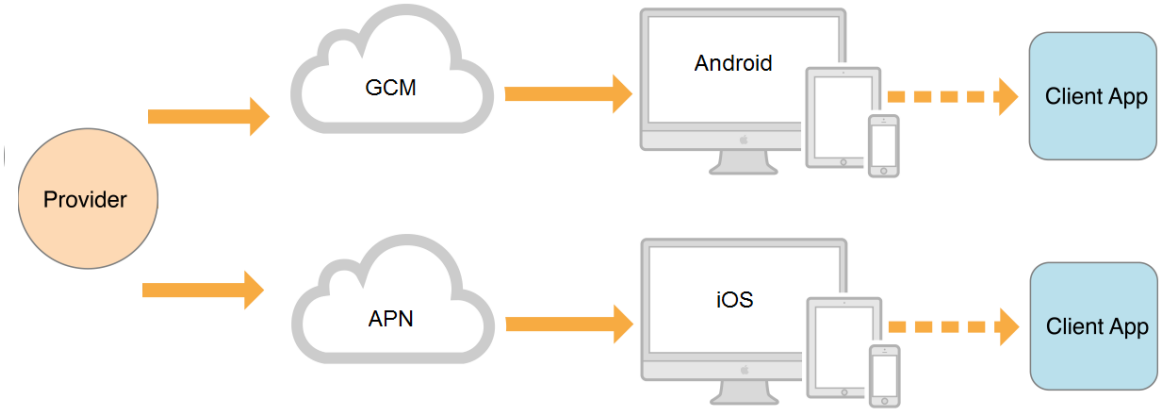
push obaveštenja na mobilni uređaj podrazumeva komunikaciju sa odgovarajućim servisom u okviru operativnog sistema mobilne platforme, generalno OSPNS (Operating system push notification service). Svaki operativni sistem mobilne platforme, uključujući iOS, Android, Fire OS, Windows i Blackberry poseduje implementaciju ovog servisa u okviru svoj operativnog sistema. Stoga u okviru svake od mobilnih platformi, odnosno proizvođača mobilnih uređaja, postoji specifični posrednički poslužioc za tu platformu.



Slika 2: Tok *push* obaveštenja

Vlasnički poslužioc, ukoliko želi da dostavi obaveštenje na ciljani mobilni uređaj određene platforme mora da, nakon generisanja obaveštenja isto prosledi odgovarajućem posredničkom poslužiocu u zavisnosti od platforme uređaja.

U slučaju OBLO projekta, gde su mobilne aplikacije razvijane za Android i iOS mobilne platforme, posrednički poslužioci su *GCM* (Google Cloud Messaging) i *APN* (Apple Push Notification) respektivno. Sa druge strane OBLO *cloud* servisi razvijeni su upotrebom Node.js platforme. To znači da iz prethodno navedenih činioca, uloga klijentske aplikacije deli se na podršku za Andoid i iOS uređaje, a ulogu vlasničkog poslužioca preuzima *cloud* servis (OBLO Stratus2).



Slika 3: Tok *push* obaveštenja u slučaju više ciljnih mobilnih uređaja

3.2 Utvrđivanje korisničkog identiteta

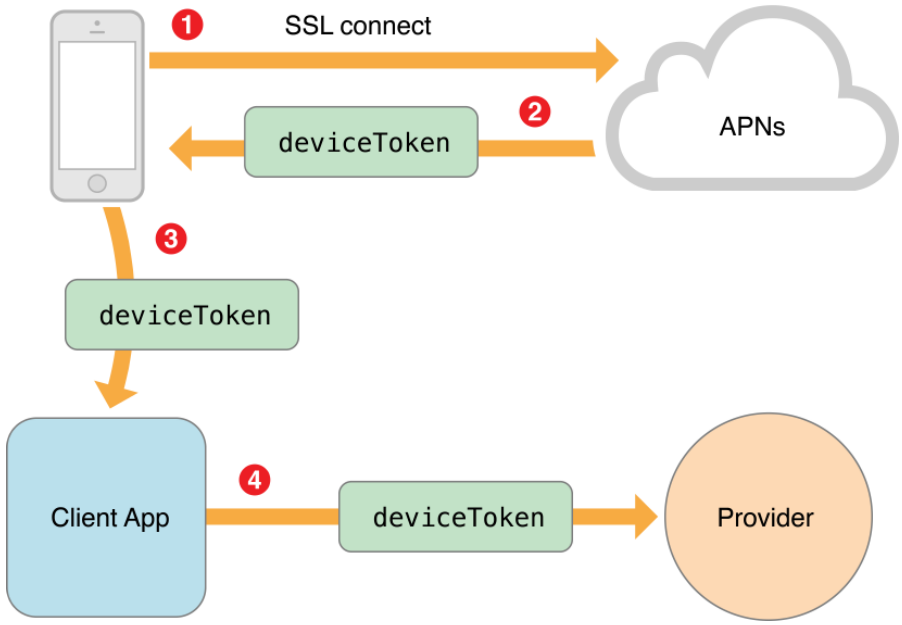
Obaveštenje, između ostalih informacija mora sadržati i identitet korisnika kome je namenjeno. Uzimajući u obzir da *cloud* servis (Stratus2) generise ova obaveštenja, identifikacija korisnika u okviru OBLO sistema, kao i posredničkih poslužioca mora biti uniformna. Sa druge strane, posredničkih poslužioca je više, odnosno putanja obaveštenja podrazumeva kako *GCM*, tako i *APN* poslužioce u zavisnosti od ciljnog uređaja.

Jednoznačno utvrđivanje korisnika vrši se upotrebom takozvanih tokena, što je utvrđen mehanizam na oba posredna poslužioca.

3.2.1 Tokeni

Klijentska aplikacija, najpre mora dobaviti ovaj token od odgovarajućeg posredničkog poslužioca. Ovaj deo specifičan je za mobilnu platformu i koristi pomenuti OSPNS kao podsistem za komunikaciju. Nakon toga, token se asocira sa aplikacijom koja je prosledila zahtev na samom uređaju. Zbog toga mobilni uređaj je u stanju da pristiglo *push* obaveštenje prikaže kao informaciju iz same aplikacije. Ovaj token tada se prosleđuje na *cloud* servis i asocira sa nalogom korisnika aplikacije u OBLO sistemu.

Sa klijentskih aplikacija, pored tokena šalje se i informacija o platformi na kojoj je aplikacija pokrenuta, odnosno, ukoliko je zahtev za tokenom protekao od Android mobilne platforme, pri prosleđivanju tokena *cloud* servisu prosleđuje se i ta informacija. Na ovaj način pri slanju obaveštenja kontaktira se tačno određeni posrednički poslužilac u zavisnosti od informacija pristiglih uz token.



Slika 4: Dobavljanje tokena od posredničkog poslužioca

3.2.2 **Multi-korisnička aplikacija i tokeni**

Token u domenu *push* obaveštenja predstavlja varijablu koja jednoznačno određuje konkretnu aplikaciju na tačno određenom mobilnom uređaju.

Iako se na prvi pogled čini da je token dovoljan za utvrđivanje identiteta korisnika, ovo nije slučaj. Razmotrimo sledeću situaciju u kojoj zarad objašnjenja imamo dva korisnika koji koriste isti uređaj i istu mobilnu aplikaciju za udaljen pristup svojim centralnim uređajima:

- Korisnik "A" se loguje na aplikaciju, te se u pozadini od posredničkog poslužioca dobavlja token koji određuje mobilni uređaj i aplikaciju.
- Ovaj token se prosleđuje Stratus2 *cloud* servisu, koji taj token asocira sa korisnikom "A". Nakon toga ukoliko dođe do slanja *push* obaveštenja iniciranog sa centralnog uređaja korisnika "A", korisnik "A" će ih primiti, između ostalih, i na tom uređaju.
- Nakon toga, korisnik "A" napušta aplikaciju (Log-out), a korisnik "B" se loguje na istom uređaju. Pri logovanju, aplikacija kontaktira posrednički poslužilac i dobija identičan token kao i korisnik "A" obzirom da se radi o istom uređaju i istoj aplikaciji.
- Ovaj token se po istom principu kao i u slučaju sa korisnikom "A" prosleđuje Stratus2 *cloud* servisu, međutim ovaj token je sada asociran i sa korisnikom "B".
- Ukoliko sada dođe do slanja *push* obaveštenja iniciranog sa centralnog uređaja korisnika "A", korisnik "B" koji je trenutno ulogovan u aplikaciju dobiće obaveštenje namenjeno korisniku "A".

Da bi se ovaj problem izbegao postoji više načina gde svi podrazumevaju dodatnu logiku kako ka *cloud* servisu, tako i na mobilnim aplikacijama. Predlozi su sledeći:

- Filtriranje poruka na klijentskom uređaju
    - Nakon što je korisnik "A" napustio aplikaciju, a korisnik "B" se ulogovao, u inicijalnom slučaju obaveštenja od korisnika "A" bi i dalje stizala na uređaj. Jedan način je da se pristigle poruke filtriraju ukoliko se ne podudaraju sa trenutno ulogovanim korisnikom.
        - Napomena je da je ovo filtriranje moguće na mobilnim uređajima Android platforme ali ne i uređajima iOS platforme. Razlog je interna realizacija ovih sistema.
- Ažuriranje tokena na *cloud* servisu
    - Pri ulasku u aplikaciju (login) dobavlja se token od posredničkog poslužioca, te se šalje na *cloud* servis. Ukoliko je u navedenom slučaju korisnik "B" poslao isti token kao i korisnik "A" na *cloud* servis, moguće

je izvršiti proveru u okviru servisa da li je pristigli token već asociran sa drugim korisnikom. Ukoliko jeste, token treba "prebaciti" na novog korisnika. Međutim, jedan graničan slučaj je kada korisnik napusti aplikaciju, ali se niko ne uloguje nakon toga. *cloud* servis tada nije svestan ove činjenice, te će obaveštenja i dalje stizati na uređaj iako je korisnik odjavljen sa sistema.

- Slanje informacije da je korisnik napustio aplikaciju.
    - Ukoliko bi se informacija o napuštanju aplikacije od strane nekog korisnika, zajedno sa trenutnim tokenom na uređaju prosledila na *cloud* servis, ovaj token bi se mogao obrisati iz liste asociranih tokena sa tim korisnikom. Na taj način *cloud* servis više ne bi slao obaveštenja na taj uređaj.
    - Slično, ova informacija bi se mogla slati i na posrednički poslužilac, koji bi tada označio token kao nevažeći. Pri eventualnom slanju *push* notifikacije za taj token sa *cloud* servisa, posrednički poslužilac bi odgovorio da je nevažeći te bi se i na *cloud* servisu token obrisao.

Problem pri ovom pristupu javlja se ukoliko korisnik moblinog uređaja u trenutku izlaska iz aplikacije nema internet konekciju. Obzirom da je jedini validan način za prestankom slanja obaveštenja upravo slanje informacije pri napuštanju aplikacije, one se moraju realizovati tako da korisniku onemoguće izlazak (Logout) ukoliko ne postoji Internet veza, i jasno daju do znanja da je ostao ulogovan iako aplikacija više nije aktivna.

Sve opcije imaju svoje prednosti ali ukoliko se realizuje samo jedna od njih, svi granični slučajevi neće biti pokriveni. Zato je u implementaciji *push* mehanizma u okviru OBLO projekta upotrebljena kombinacija ovih predloga zarad stabilnosti sistema.

"Ažuriranje tokena na *cloud* servisu" upotrebljeno je radi pokrivanja slučaja kada isti uređaj (i aplikaciju) koriste više korisnika.

"Slanje informacije da se korisnik izlogovao", koristi se radi pokrivanja slučaja kada se korisnik izloguje iz aplikacije (a nakon toga nije bilo logovanja).

"Filtriranje poruka na klijentskom uređaju" se koristi u slučaju kada korisnik na Android mobilnoj platformi (obzirom da je u mogućnost) obriše sve podatke aplikacije (clear data). Obzirom da u sistemu ne postoji način za detektovanjem ovog događaja, a sa brisanjem podataka obrisao se i nalog trenutno ulogovanog korisnika, sva *push* obaveštenja pristigla nakon ovog događaja treba zanemariti. Obzirom da korisnik na mobilnom uređaju u suštini

više ne postoji, mogućnost za kontaktiranjem *cloud* servisa nije moguća. U tom slučaju na nivou aplikacije potrebno je filtrirati pristigle poruke tako da se ne prikazuju. Ovo je moguće jer *push* obaveštenja na Android platformi najpre završavaju u samoj aplikaciji koja definiše njeno dalje ponašanje, za razliku od iOS sistema koji interno rukuje obaveštenjima pre nego što ona završe u aplikaciji. Međutim, problem brisanja podataka aplikacije na iOS-u ne postoji jer takva opcija ne postoji.

3.3 Deinstalacija aplikacije

Slučaj u kom korisnik deinstalira aplikaciju sa svog mobilnog uređaja prethodno asociranog na *cloud* servisu mora biti propraćena na odgovarajući način.

Token na *cloud* servisu više nema potrebe čuvati jer aplikacija kojoj bi obaveštenja bila namenjena više ne postoji. U tu svrhu posrednički poslužioci obezbeđuju određene mehanizme za ažuriranje tokena na vlasničkim poslužiocima.

3.3.1 *GCM*

Što se tiče *GCM* poslužioca, tok detektovanja deinstalirane aplikacije je sledeći:

- Krajnji korisnik briše aplikaciju sa uređaja,
- Zarad objašnjenja, recimo da tada dolazi do generisanja i slanja obaveštenja sa OBLO *cloud* servisa na *GCM*,
- *GCM* šalje obaveštenje na odgovarajući mobilni klijent,
- Mobilni uređaj, odnosno OSPNS na mobilnom uređaju dobija obaveštenje i proverava da li postoje aplikacije asocirane sa pristiglim obaveštenjem (preko tokena). Za ovaj korak OSPNS dobija negativan odgovor jer u tom momentu na uređaju ne postoji odgovarajuća aplikacija,
- OSPNS na uređaju informiše *GCM* poslužilac da je aplikacija deinstalirana, te je token na *GCM*-u markiran za brisanje.
- Obzirom da povratna informacija o pristiglom *push* obaveštenju na mobilne uređaje ne postoji, tek pri sledećem pokusaju slanja obaveštenja sa OBLO *cloud* na *GCM*, očekuje se prijava da je aplikacija na samom uređaju deinstalirana, te se tada odbacuje i token na OBLO *cloud* servisu.

3.3.2 *APN*

Što se tiče *APN* servisa, upotrebom povratne sprege tj. tzv. „*feedback*“ mehanizma moguće je na *cloud* strani detektovati slučaj deinstalacije aplikacije. U okviru *APN* poslužioca, postoji tzv. “*feedback*” servis koji pruža informacije o neuspešnim pokušajima dostave obaveštenja. Po navodima *APN*-a, ovaj servis sadrži listu tokena uređaja koji više ne sadrže instaliranu

odgovarajuću aplikaciju. Međutim, bitna stavka jeste da ukoliko se token na *APN*-u ponovno registruje (deinstalacija i nakon toga ponovna instalacija aplikacije) velika je šansa da će uređaj dobiti isti token pri oba pokušaja registracije.

Obzirom da je taj token već registrovan u "*feedback*" listi, neophodno je uporediti vremenski pečat za konkretan token iz "*feedback*" liste, sa vremenskim pečatom tokena asociranog sa korisničkim nalogom u okviru OBLO *cloud* servisa. Samo ukoliko je vremenski pečat iz "*feedback*" servisa svežiji, tada je potrebno ukloniti token sa OBLO *cloud* servisa. U suprotnom token je obnovljen i može se koristiti za dalje slanje *push* obaveštenja.

3.4 Enkripcija poruka

Što se sigurnosti poruka tiče, enkripcija upotrebom TLS kanala prisutna je na oba posrednička poslužioca (*APN*/*GCM*). Drugim rečima komunikacija između *cloud* servisa ili mobilnih aplikacije prema posredničkom poslužiocu enkriptovana je. I dalje, ovo nije enkripcija "sa kraja na kraj" (end-to-end) gde podatke mogu razumeti samo strane koje razmenjuju poruke tj. u slučaju *push* notifikacija: *cloud* servis i mobilna aplikacija. To znači da poruke može pročitati, pored mobilne aplikacije i naravno *cloud* servisa koji poruku generiše, i posrednički poslužilac (*APN* i *GCM*). Pitanje je samo da li je validno da informacije u okviru IoT domena prolaze kroz ove posredničke poslužioce, odnosno da li verujemo da se informacije neće širiti ili iskoristiti u nepoželjne svrhe.

Jedan deo komunikacije koji nije pomenut a tiče se sigurnosti jeste slanje tokena sa mobilne aplikacije na *cloud* servis. Ovaj deo rešen je upotrebom sigurne veze preko MQTT protokola koji se koristi kao mehanizam za standardnu razmenu poruka u sistemu, preko koga se šalje i token sa mobilnog uređaja.

3.5 Node.js biblioteke za komunikaciju sa *APN*/*GCM* servisima

U okviru OBLO projekta, *cloud* servis realizovan je pod Node.js platformom. Servis je zadužen za pružanjem usluga kako korisnicima tako i administratorima OBLO sistema.

U osnovnom obliku namena je udaljena kontrola, međutim sadrži i funkcije kao što su zadavanje ažuriranja softvera na centralnom uređaju, uvid u stanje više povezanih centralnih uređaja asociranih na nalog, udaljeno preuzimanje izveštaja sa centralnog uređaja i slično.

Komunikacija sa centralnim uređajima, što se strane *cloud*-a tiče, omogućena je upotrebom klijentske biblioteke za MQTT pod nazivom *Mosca* [7].

C*loud* servis nakon što primi informaciju od centralnog uređaja značajnu za korisnika, mora kreirati *push* poruku sa zaglavljem odgovarajućim za ciljani posrednički poslužilac. Deo odlučivanja kojem poslužiocu (*APN* ili *GCM*) poruku treba proslediti rešen je pri dostavljanju

tokena iz same aplikacije, uz kojeg se salje i platforma uređaja. Ostaje problem komunikacije sa različitim poslužiocima tj. implementacija klijentskih biblioteka za ovu spregu.

Obzirom na popularnost i prevashodnu namenu platforme Node.js, zasebna platforma za deljenje i manipulisanje JavaScript komponentama NPM (Node Package Manager) većim delom je sadržana upravo modulima za Node.js. Po zvaničnim navodima kompanije, trenutno je oko 250,000 modula na NPM-u što ga čini najvećim registrom softverskih biblioteka na svetu.

U tako popularnom ekosistemu modula, a uzimajući u obzir namenu platforme prevashodno za razvoj poslužioca, našao se i skup biblioteka za komunikaciju sa *GCM* I *APN* poslužiocima. Sledeća table prikazuje poređenje po tri biblioteke sortirane po popularnosti na NPM-u, kao zvaničnom repozitorijumu modula, respektivno:

***GCM* klijent**	**node-*GCM***	***GCM***	**android-*GCM***
Licenca	MIT	MIT	MIT
Preuzimanja sa NPM u proteklih mesec dana (datum provere 29.6.2016)	64740	5319	160
Problema na GitHub (rešenih/otvorenih)	128/26	3/1	1/1
Ukupno zavisi od drugih biblioteka	3	5	2
Zavisnost drugih biblioteka od ove	46	3	1

Tabela 1: Poređenje biblioteka za komunikaciju sa *GCM* posredničkim poslužiocem

***APN* klijent**	***APN***	***APN*agent**	***APN*s**
Licenca	MIT	MIT	Free
Preuzimanja sa NPM u proteklih mesec dana (datum provere 29.6.2016)	90410	1859	235
Problema na GitHub (rešenih/otvorenih)	296/20	6/14	1/0
Ukupno zavisi od drugih biblioteka	3	10	1
Zavisnost drugih biblioteka od ove	63	3	0

Tabela 2: Poređenje biblioteka za komunikaciju sa *APN* posredničkim poslužiocem

Iz tabela vidi se da su “node-*GCM*” za komunikaciju sa *GCM* poslužiocem, odnosno “*apn*” za komunikaciju sa *APN* poslužiocem drastično ispred drugih po pitanju referenci. Zavisnost velikog broja drugih biblioteka, pored broja svlačenja na mesečnom nivou, predstavljaju najbitnije tačke pri odlučivanju korišćenja biblioteke zbog adresiranja i rešavanja potencijalnih problema, kao i dokaza stabilnosti u radu.

4. Programsko rešenje

Programsko rešenje se proteže na *cloud* servis i mobilne aplikacije OBLO projekta, međutim ovaj rad se fokusira na realizaciju podsistema za komunikaciju sa posredničkim poslužiocima (*GCM*/*APN*) u okviru *cloud* servisa. Sama implementacija na mobilnim platformama nije pokrivena, iako je radi sagledavanja problema sa ovim mehanizmom razrađena na konceptualnom nivou (odeljci 3.2.2 i 3.3).

4.1 Blokovi *cloud* servisa

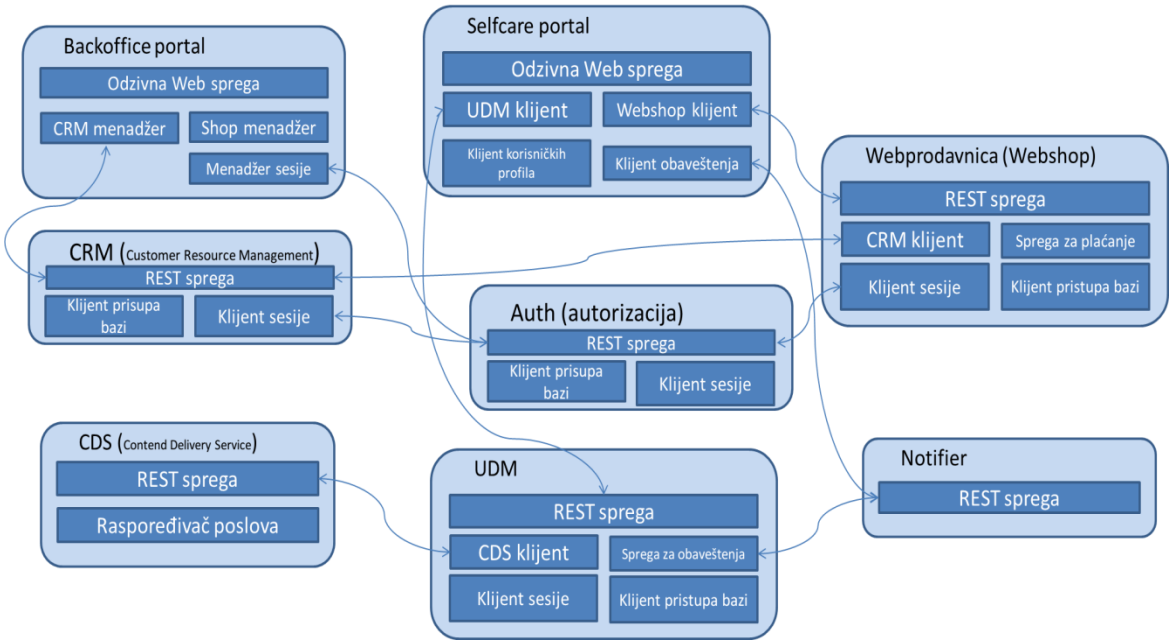
Glavna karakteristika *cloud* servisa je njegova arhitektura po principu mikro servisa, gde različiti moduli odvojene logike predstavljaju fizičke instance servera koji međusobno komuniciraju HTTP protokolom. Za razliku od servisne arhitektire (SOA - "service-oriented-architecture"), ovi blokovi su limitirani po pitanju veličine, tj. poslovne logike koju obrađuju.

Osnovne funkcionalnosti *cloud* servisa u projektu OBLO, razdvojene se po sledećim mikro servisima:

- Selfcare
    - Portal namenjen korisnicima. Kroz ovu komponentu korisnici su u mogućnosti za kontrolom i konfiguracijom svojih IoT sistema. Neke od mogućnosti su uvid u stanje uređaja, kontrola uređaja, ažuriranje verzija softvera, kupovina proizvoda, lista događaja i slično.
- Backoffice
    - Portal namenjen administratorima OBLO sistema. Administratori su podeljeni po nivou dozvoljenog pristupa preko koga mogu dobiti uvid u korisničke naloge, kupce, kategorije proizvoda u prodavnici i slično.
- Webshop

- Prodavnica proizvoda dostupnih korisnicima sistema kao i potencijalnim kupcima.

- CRM (Customer relationship management)
  - Deo sistema zadužen za manipulaciju korisničkih naloga. Nalozi kreirani u ovom modulu sadrže identifikaciju koja korisnike jednoznačno određuje na nivou celog sistema.
- Auth (Authorization)
  - Autentifikacioni servis koji je zadužen za proveravanje prava pristupa korisnika (putem Selfcare portala) i centralnih uređaja.
- Notifier
  - Servis zadužen za dostavljanje obaveštenja korisnicima.
- CDS (Content Delivery Service)
  - Servis zadužen za koordinisanje digitalnog sadržaja namenjenog korisnicima sistema.
- CDN (Content Delivery Node)
  - Iako fizički odvojen, usko povezan sa CDS servisom. Služi za skladištenje i dostavljanje sadržaja korisnicima.
- UDM (User Device Management)
  - Servis zadužen za komunikaciju sa centralnim uređajima korisnika.



Slika 5: Osnovni *cloud* servisi

Na slici je načelno opisana komunikacija među internim modulima *cloud* servisa. Zarad komunikacije sa mobilnim klijentima i centralnim uređajima, koristi se namenski servis koji predstavlja MQTT broker pod nazivom Mosca. Nakon pokretanja *cloud* servisa, centralni uređaji korisnika, kao i interni moduli povezuju se sa Mosca-om. Dalja razmena poruka vrši se posredstvom ovog servisa.

4.2 Format poruka

Ukoliko su događaji generisani na centralnom uređaju od značaja korisnicima, pre slanja na *cloud* servis putem MQTT protokola, formatiraju se u predefinisan oblik koji će se razložiti radi odluke za načinom slanja na određeni klijentski uređaj. Takođe, poruke mogu biti različitog tipa, ali od značaja za mehanizam *push* obaveštenja, zadržavamo se na porukama tipa "evt" odnosno događaj (event) i imena "NotificationEvent". Format poruka ovog tipa sadrži sledeća polja:

Naziv polja	Tip	Obavezno	Opis
user_id	String	Da	Identifikacija korisnika koja odgovara CRM servisu, kao i korisniku ulogovanom u aplikaciju
push_id	String	Da	Token dostavljen od strane samog klijenta. Jednoznačno određuje aplikaciju na uređaju. *Jedinstven u šemi podataka
device_type	String	Da	Tip ondosno platforma uređaja (Android ili iOS)
device_model	String	Ne	Model mobilnog uređaja.
device_name	String	Ne	Naziv mobilnog uređaja

Tabela 3: Format *push* poruke

Nakon pristizanja poruka na *cloud* servis, njihova obrada vrši se u zavisnosti od tipa i navedenih polja. Ukoliko je format takav da određuje poruke koje je potrebno proslediti

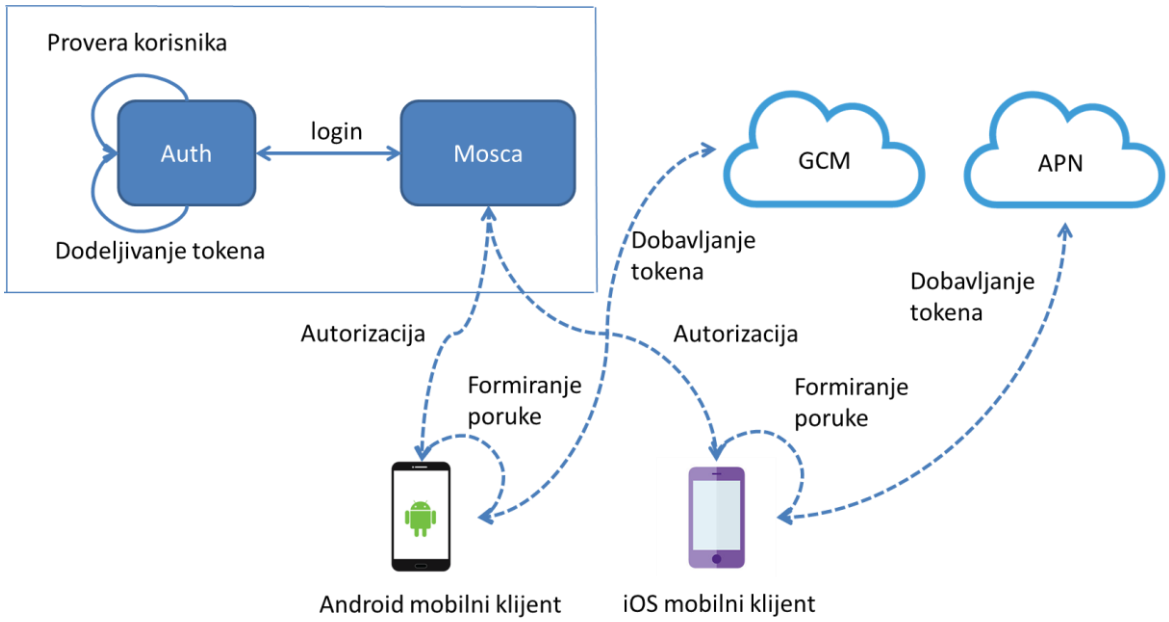
korisnicima, poruka se dopunjuje dodatnim informacijama iz sistema neophodnim za dostavu na korisnički mobilni uređaj.

4.3 Sekvenca registrovanja tokena

Registrovanje tokena na *cloud* servisu vrši se u toku registrovanja korisnika na sistem (login procedura). Kao što je objašnjeno u poglavlju 3.2.2, pri svakom pristupu *cloud* servisu, mobilni klijent mora obnoviti token (komunikacija sa posredničkim poslužiocima).

Nakon pristizanja zahteva na *cloud* servis, *Auth* servis zadužen je za kreiranje sesije za konkretnog korisnika, pri čemu dodatnu operaciju predstavlja asociranje prosleđenog tokena sa identifikacijom korisnika u bazi. To znači da se svi podaci o korisnicima, zajedno sa svim tokenima čuvaju u ovom *cloud* servisu.

Pri autorizaciji korisnika na sistem koristi se pojašnjen format poruke. Najpre klijent dobavlja token od posredničkog poslužioca, te se njegova vrednost smešta u polje push_id. Ostala polja popunjavaju se u zavisnosti od unetog naloga korisnika (user_id) i platforme uređaja (device_type) na kome se aplikacija izvršava. Ovako formirana poruka šalje se uz zahtev za autorizacijom na *cloud* poslužilac. Nakon pristizanja zahteva, najpre se proverava identitet korisnika (da li je aktiviran) te se pre dodavanja tokena proverava da li postoji drugi nalog kome je ovaj token već dodeljen. Ovo predstavlja jedan od pomenutih graničnih slučaja u višekorisničkim aplikacijama. Ukoliko je token asociran na drugog korisnika, "prebacuje" se na korisnika koji je inicirao proces autorizacije. U suprotnom, prosleđen token se svakako dodaje nalogu korisnika, ali drugi nalozi ostaju nepromenjeni.



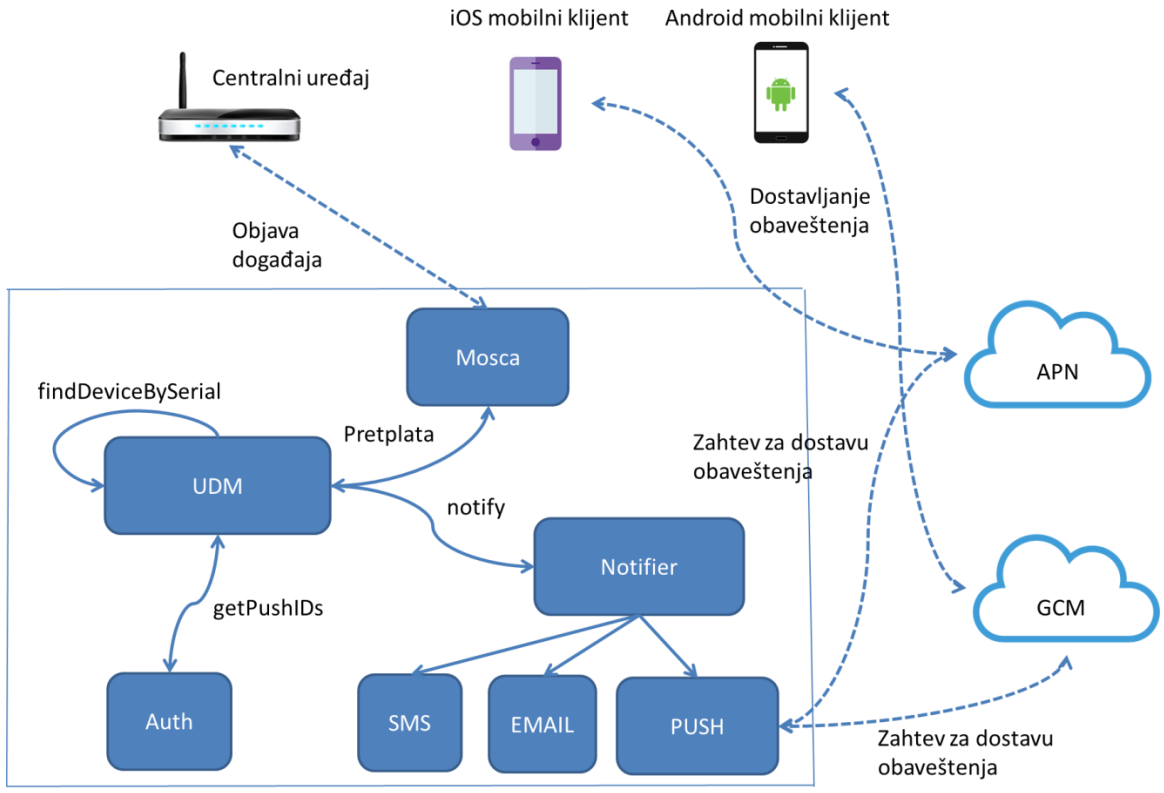
Slika 6: Sekvenca registrovanja tokena

4.4 Dopunjavanje poruke

UDM je modul zadužen za komunikaciju sa centralnim uređajima (preko brokera Mosca), te on predstavlja ulaznu tačku obaveštenja na *cloud* servis. Nakon pristizanja obaveštenja, ovaj modul mora obezbediti dobavljanje dodatnih informacija koje će dopuniti inicijalnu poruku radi dostavljanja na mobilni uređaj (tačka 3, poglavlje 1.1). Ovo podrazumeva podatke o korisniku kojem je obaveštenje namenjeno, dobavljanje tokena asociranih uz tog korisnika, filtriranje poruke u zavisnosti od konfiguracije preko Selfcare portala, i kreiranje objekta koji grupiše ove informacije. Taj objekat se prosleđuje Notifier servisu, koji je zadužen za njegovu dalju obradu, odnosno delegiranjem određenom skupu pod-servisa u zavisnosti od metode slanja (email, *push*, SMS ili kombinacija). Dopunjavanje informacijama je neophodno radi identifikacije mobilnih uređaja i tokena klijenata kojima se obaveštenje dostavlja.

4.5 Sekvenca slanja obaveštenja

Tok slanja obaveštenja od centralnog uređaja do mobilnih uređaja prikazan je na slici. U integraciji ovog mehanizma, postojeća logika na prikazanim blokovima *cloud* sistema iskorišćena je ili proširena da podrži operacije neophodne za dostavljanje obaveštenja.



Slika 7: Sekvenca slanja obaveštenja

Sekvenca je sledeća:

- Centralni uređaj nakon detektovanja određenog stanja, generiše poruku za *cloud* servis u pomenutom formatu i objavljuje je na MQTT broker. Obzirom da je UDM servis tada pretplaćen na poruke pristigle od centralnog uređaja, dalja obrada vrši se u ovom modulu. Najpre se poruka filtrira po tipu. Tip od interesa za ovaj rad je "NotificationService" koji predstavlja poruku namenjenu za prosleđivanje krajnjim korisnicima.
- UDM je zadužen za dopunjavanje pristigle poruke dodatnim informacijama iz sistema radi užeg definisanja daljeg toka poruke, te dobavlja informacije o centralnom uređaju koji je obaveštenje inicirao po serijskom broju. Ove informacije između ostalog sadrže i identifikaciju asociranih korisnika. Metoda koja obrađuje pristiglo obaveštenje je *handleNotification* iz modula UDM:

```
1.  handleNotification: function (topicData, msg) {
2.    return Device.findOneBySerialNumberQ(topicData.serialNumber).then(function (device) {
3.      if (device != undefined && device.userId != undefined) {
4.        if (msg.type == "push") {
5.          .
6.          .
7.          .
8.        } else if (msg.type == "email") {
9.          .
10.         .
11.         .
12.       } else if (msg.type == "sms") {
13.         .
14.         .
15.         .
16.       }
17.     }
18.   });
19. }
```

- Pomoću ID broja korisnika dobijenog preko serijskog broja centralnog uređaja, dobavljaju se njemu asocirani klijentski uređaji u vidu tokena generisanih od strane samih aplikacija. Ovi podaci grupišu se u novi objekat koji se prosleđuje *Notifier* servisu metodom "notify" gde se dalje odvija komunikacija sa različitim posredničkim poslužiocima u zavisnosti od tipa obaveštenja (Email, SMS, *push*). Slanje obaveštenja na više uređaja (tokena) iste platforme šalje se kao jedan zahtev posredničkom poslužiocu. Stoga postoji odvojeno grupisanje tokena za *Android* i *iOS* platforme. Dobavljanje i grupisanje neophodnih podataka vrši se na sledeći način:

```
return api.communicator.strat_auth.getPushIDs.invoke({userid: device.userId}).then(function (data) {
    var iosIDs           = [];
    var androidIDs       = [];
    data.pushdata.forEach(function (notificationObj) {
      if (notificationObj.devicetype == "ios") {
        iosIDs.push(notificationObj.pushid);
      } else if (notificationObj.devicetype == "android") {
        androidIDs.push(notificationObj.pushid);
      } else {
        logger.info("Unknown device type for push notification " + notificationObj.devicetype);
      }
    });
    var curTimestamp     = getDateTime();
    var notificationData = {
      notifications: []
    };
    if (iosIDs.length > 0) {
      notificationData.notifications.push({
        deviceType: "ios",
        contactType: "pushMobile",
        title: msg.short_desc,
        message: msg.desc,
        timestamp: curTimestamp,
        gateway: device.name,
        registrationIds: iosIDs
      });
    }
    if (androidIDs.length > 0) {
      notificationData.notifications.push({
        deviceType: "android",
        contactType: "pushMobile",
        title: msg.short_desc,
        message: msg.desc,
        timestamp: curTimestamp,
        gateway: device.name,
        registrationIds: androidIDs
      });
    }
    return api.communicator.strat_notifier.notify.invoke(notificationData).then(function (data) {
      logger.info("Notification data sent to notifier");
    });
});
```

- Konkretno slanje obaveštenja na posredničke poslužioce realizovano je u modulu *Notifier*. *Notifier* u zavisnosti od prosleđenog metoda dostavljanja obaveštenja, prepakuje poruku u oblik predefinisan od strane posredničkih poslužioca, i ovako formulisanu poruku šalje na iste. Ukoliko je poruka namenjena *Andoid* uređajima, slanje na *GCM* posrednički poslužilac realizovano je u metodi *pushAndroid:*

```
function pushAndroid(notification, callback) {
  var message = new gcm.Message({
    data: {
      title: notification.title,
      message: notification.message,
      timestamp: notification.timestamp,
      gateway: notification.gateway
    }
  });
  var sender  = new gcm.Sender(params.pushMobile.android.androidAppID);
  sender.send(message, {registrationIds: notification.registrationIds}, function (err, result)
  {
    if (err) {
      logger.error('Error while trying to send push notification: ', err);
      callback(err);
    } else {
      if (result.results[i].error && result.results[i].error == "NotRegistered")
          deleteDeviceId(notification.registrationIds[i]);
      var resp = _formatPushNotificationResponse(notification, result);
      callback(null, resp);
    }
  });
}
```

Dok je za *iOS* mobilne uređaje funkcija koj prosleđuje obaveštenje *APN-u* pushApple:

```
function pushApple(notification, callback) {
  logger.info("Push apple");
  note.setAlertText(notification.message);
  note.alert = {
    title: notification.title,
    body: notification.message,
    timestamp: notification.timestamp,
    gateway: notification.gateway
  };
  note.badge = 1;
  note.sound = "default";
  apnConnection.pushNotification(note, notification.registrationIds);

  _formatPushNotificationResponse(notification, null);
  callback(null, notification);
}
```

- Nakon odgovora da je zahtev uspešno razložen, te da su tokeni validni, Notifier je zadužen za skladištenje ovih obaveštenja radi mogućnosti pregleda istorije od strane korisnika, i vraćanjem statu sa o dostavljanju obaveštenja posredničkim poslužiocima. Stoga se poruke formatiraju u oblik za skladištenje u bazu, koji podrazumeva status o dostavi poruke, tip ovaveštenja (SMS, *push*, *email*), i tip uređaja pored same poruke. Metoda *formatPushNotificationResponse* formira objekat pogodan za upis u bazu:

```
function pushApple(notification, callback) {
  logger.info("Push apple");
  note.setAlertText(notification.message);
  note.alert = {
    title: notification.title,
    body: notification.message,
    timestamp: notification.timestamp,
    gateway: notification.gateway
  };
  note.badge = 1;
  note.sound = "default";
  apnConnection.pushNotification(note, notification.registrationIds);

  _formatPushNotificationResponse(notification, null);
  callback(null, notification);
}
```

  - Glavna razlika metoda *pushApple* i *pushAndroid* je što se za slanje na *GCM* odgovor o nevalidnom tokenu (aplikacija obrisana) dobija sinhrono, dok za *APN* ovaj odgovor stiže preko povratne sprege (eng. *feedback*). Pri detekciji nevalidnog tokena, potrebno je isti obrisati iz baze na *cloud* servisu, za šta je zadužena metoda *deleteDeviceID:*

```
function deleteDeviceId(deviceID) {
  api.communicator.stratus.crm.removeNotificationId.invoke({
      pushId: deviceID
    })
    .then(function (value) {
      logger.info('DeviceID removed from ACS');
    }, function (reason) {
      logger.info('Error occured while trying to delete deviceID from ACS');
    });
};
```

4.6 Brisanje tokena

Brisanje tokena sa *cloud* servisa radi se u dva slučaja. Ukoliko je korisnik ručno napustio aplikaciju (logout), ili ukoliko su korisnički tokeni na posredničkim poslužiocima označeni kao nevalidni što se događa ukoliko korisnik obriše aplikaciju sa uređaja. U slučaju nevalidnih tokena, pokušaj slanja obaveštenja na neki od posrednički poslužioca rezultuje greškom u odgovoru što se tiče *GCM* poslužioca, odnosno pokretanjem funkcije povratnog poziva (eng. *callback*) na *APN*. Oba slučaja predstavljaju indikaciju da token više nije validan te ga je potrebno odbaciti na *cloud* servisu (pogledaj poglavlja 3.2 i 3.3).

5. Rezultati i merenja

Ispitivanje projekta izvršeno je konfigurisanjem OBLO sistema, tako da događaji kao što je promena stanja kontakt senzora aktivira određenu scenu. Ti događaji upotrebljeni su kao okidači scena koje reaguju slanjem *push* obaveštenja na korisničke mobilne uređaje. Obaveštenja se šalju na sve uređaje asocirane sa korisnikom čiji centralni uređaj je povezan sa kontakt senzorom.

Pored funkcionalnih testova slanja obaveštenja, testovi merenja vremena dostave u zavisnosti od Internet konekcije (preko lokalnog rutera ili GSM mreže), kao i testovi brisanja aplikacije i granični slučajevi sa višekorisničkom upotrebom aplikacije takođe su pokriveni.

5.1 Performanse posredničkih poslužioca

Što se tiče merenja performansi posredničkih poslužioca, zvanični navodi vremenskih garancija Apple i Google kompanija ne postoje. U toku istraživanja uzeta su u obzir i anketiranja, koja pokrivaju slučajeve hiljade korisnika mobilnih uređaja sa različitu vrstu Internet veze (WiFI, Cellular). Zaključak u konkretnom slučaju istraživanja *GCM* posredničkog poslužioca je da ovaj servis nije pogodan za vremenski senzitivna obaveštenja. Razlog je, pre svega taj što ne postoji garancija za dostavom obaveštenja čak i ukoliko je mobilni uređaj u vezi sa *GCM* poslužiocem. U istraživanju zaključeno je da ukoliko je ova veza uspostavljena, slanje obaveštenja velikom broju korisnika odjednom rezultuje u 80% dostavljenih poruka i to u vremenskom periodu od 10 sekundi [9].

Takođe, po navodima istraživača, *GCM* može biti pogodan za situacije gde je nasumično dostavljanje poruka grupi korisnika (uređaja), što je upravo slučaj sa situacijom obaveštenja korisnika o događajima u okviru njihove “pametne” kuće.

5.2 Funkcionalni testovi

Slučaj	Android	iOS
Registracija tokena na *cloud* servis pri ulasku u aplikaciju (eng. *login*)	✅	✅
Primanje obaveštenja nakon ulaska u aplikaciju	✅	✅
Distribucija obaveštenja na više uređaja asociranih sa jednim korisnikom (nezavisno od platforme)	✅	✅
Prestanak primanja obaveštenja na uređaju nakon napuštanja aplikacije (logout)	✅	✅

Tabela 4: Funkcionalni testovi

5.3 Testovi graničnih slučajeva

Primanje obaveštenja samo poslednje ulogovanog korisnika (u slučaju gde korisnik "A" napusti aplikaciju a umesto njega se uloguje korisnik "B")	✅	✅
Prestanak primanja obaveštenja nakon brisanja podataka aplikacije (clear data)	✅	*Na iOS uređajima operacija brisanja podataka aplikacije nije moguća
Detekcija nevalidnih tokena sa posredničkih poslužioca (*GCM*/*APN*)	✅	✅
Brisanje tokena na *cloud* servisu nakon deinstalacije aplikacije na mobilnom uređaju	✅	✅

Tabela 5: Testovi graničnih slučajeva

Tabele funkcionalnih i graničnih slučajeva pokazuju da su osnovne funkionalnosti, kao i svi adresirani problemi pri korišćenju *push* obaveštenja rešeni upotrebom dodatne logike raspoređene kako na mobilnim aplikacijama tako i na *cloud* serveru. Ovi slučajevi su detaljno opisani u poglavlju "Utvrđivanje korisničkog identiteta pod oznakom 3.2" i "Deinstalacija aplikacije" pod oznakom 3.3.

5.4 Merenje vremena dostave obaveštenja u OBLO sistemu

U testu merenja vremena dostave, konfiguracija je takva da jedan korisnik poseduje Android i iOS klijentske uređaje asocirane sa centralnim uređajem. Vrednosti vremena predstavljaju raspon od iniciranja slanja obaveštenja (promena stanja određenog uređaja), do momenta stizanja obaveštenja na mobilni uređaj.

Ovo podrazumeva vreme potrebno da se događaj prosledi od perifernog do centralnog uređaja, aktivaciju scene, slanje informacije na cloud servis, te njegovu obradu pre slanja obaveštenja na posrednički poslužilac koji push obaveštenje prosleđuje mobilnom uređaju. Vreme je prikazano u zavisnosti od Internet veze samog mobilnog klijenta, odnosno prikazana je razlika između WiFi konekcije u lokalnoj mreži naspram GSM konekcije pri dostavi obaveštenja. Razlika u putanjama je ta da se u slučaju slanja obaveštenja na Android uređaj kontaktira *GCM*, a u slučaju slanja na iOS uređaj *APN* posrednički poslužilac.

Mreža	Android	iOS
WiFi	**~1-5 sec**	**~1-5 sec**
GSM	**~2-15 sec**	**~2-15 sec**

Tabela 6: Vremena dostave obaveštenja u OBLO sistemu

Vidi se da Internet veza donekle utiče na vreme dostave obaveštenja. Ovaj test pokazuje okvirno vreme dostave obaveštenja na grupu mobilnih uređaja nekog korisnika (naloga), gde se informacije prostiru od perifernih do mobilnog uređaja. Ovo je standardni slučaj slanja obaveštenja u OBLO sistemu.

6. Zaključak

Iako se mehanizam *push* obaveštavanja najbolje uklapa u IoT sisteme, zbog svojih karakteristika kao što su iniciranje sa strane poslužioca, funkcionisanje bez pozadinskih servisa ili aktivnih aplikacija i izbegavanja potencijalnih filtera, ovaj mehanizam nije dovoljan za sigurno obaveštavanje korisnika o događajima u sistemu. Razlog za to je neizbežna zavisnost od posredničkih poslužioca, koji u ekstremnim slučajevima ne garantuju ni isporuku ni maksimalno vreme isporuke obaveštenja [9]. Stoga jedino pravo rešenje za obaveštavanjem korisnika jeste kombinacija pomenutih mehanizama (Email, SMS, *push*) u slučajevima definisanim poslovnom logikom projekta.

Ovaj rad bavi se realizacijom *push* obaveštenja u okviru postojećeg rešenja ”pametne” kuće OBLO. Uz to, adresirani su problemi koje ovaj mehanizam uzrokuje u korišćenju pri graničnim slučajevima kao što su višekorisničke aplikacije, deinstalacija aplikacija, ažuriranje i asocijacija tokena sa korisnicima na *cloud* servisu.

Svi pomenuti problemi su rešeni upotrebom dodatne logike kako na *cloud* servisu, tako i na mobilnim aplikacijama, međutim i dalje postoje otvorena pitanja i problemi sa posredničkim poslužiocima detektovanim od strane zajednice [8]. Ovo je još jedan dokaz da oslanjanje isključivo na ovaj mehanizam nije pouzdan način za dostavu obaveštenja, međutim njegova podrška u sistemu je neophodna zbog pogodnosti koje nudi.

U OBLO sistemu, pomenuti mehanizmi slanja obaveštenja koriste se u zavisnosti od korisničke konfiguracije sistema te predstavljaju opciju što zbog slobode korišćenja sistema, što zbog sigurnosti poruka, obzirom da enkripcija nije “sa-kraja-na-kraj” (end-to-end) te pristup porukama imaju i posrednički poslužioci.

Još jedna napomena je da se oba korišćena posrednička poslužioca i dalje aktivno razvijaju te je u toku izrade rada došlo do izmena kako na *GCM*, tako i na *APN* poslužiocu. Izmene svakako uvode poboljšanja što se tiče korišćenja sa strane *cloud* servisima, ali i mobilnih

aplikacija. Naime, *GCM* je sada postao deo većeg servisa pod nazivom FCM (Firebase cloud messaging) što uzrokuje neophodne izmene u komunikaciji mobilnih aplikacija i ovog posredničkog poslužioca. Sa druge strane, na *APN*-u uvedene su promene koje adresiraju problem neaktivnih tokena (”Feedback” servis zamenjen standardnim HTTP odgovorom). U daljem radu ove izmene posredničkih poslužioca moraju biti ispraćene prilagođenjem OBLO sistema.

7. Literatura

[1] Milan Tucić: IoT proširenje za povezivanje sistema za automatizaciju kuće korišćenjem M2M protokola, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka, 2015.

[2] Ivan Lazarević, Modular home automation software with uniform cross component interaction based on services, Berlin, 6-9 Sept. 2015, Page(s): 363 – 365.

[3] Istraživanje metoda oglašavanja, http://www.zipstripe.com/text-vs-email-sms-marketing/, Jul, 2015

[4] Statistika marketinškog oglašavanja preko email, dostupno na: https://www.vision6.com.au/metrics/, Jul 2016

[5] „Govorno oglašavanje “, dostupno na: http:// mobilesquared.co.uk/ media/27820/ Conversational-Advertising_SinglePoint_2010.pdf, Jul 2016

[6] Node.js fondacija, dostupno na: https://nodejs.org/en/foundation/members/, Jul 2016

[7] JavaScript MQTT broker pod nazivom Mosca, dostupno na: https://github.com/mcollina/mosca, Jul 2016.

[8] Više validnih tokena za aplikaciju/telefon, dostupno na: https://forums.developer.apple.com/thread/25914, Jul 2016.

[9] Istraživanje performansi GCM posredničkog poslužioca, dostupno na; http://www.cse.buffalo.edu/~demirbas/publications/*GCM*.pdf, Jul 2016.