



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У  
НОВОМ САДУ



---

Милица Матић

# Реализација OAuth 2.0 модула у оквиру IoT система

ДИПЛОМСКИ РАД  
- Основне академске студије -

Нови Сад, 2017



## КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Милица Матић		
Ментор, МН:	проф. др Иштван Пап		
Наслов рада, НР:	Реализација OAuth 2.0 модула у оквиру IoT система		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2017		
Издавач, ИЗ:	Ауторски репринт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страница/цитата/табела/слика/графика/прилога)	7/41/16/0/28/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кључне речи, ПО:	IoT, OAuth 2.0, повезивање налога		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У раду је описан проблем повезивања корисниковог налога на страној апликацији са OBLO налогом ради приступања ресурсима OBLO сервиса. Овај проблем решен је коришћењем OAuth 2.0 окружења за аутентификацију и његова два тока провере овлашћења страној апликацији.		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	доц. др Иван Каштелан	
	Члан:	доц. др Немања Лукић	Потпис ментора
	Члан, ментор:	проф. др Иштван Пап	



## KEY WORDS DOCUMENTATION

Accession number, <b>ANO:</b>			
Identification number, <b>INO:</b>			
Document type, <b>DT:</b>	Monographic publication		
Type of record, <b>TR:</b>	Textual printed material		
Contents code, <b>CC:</b>	Bachelor Thesis		
Author, <b>AU:</b>	Milica Matić		
Mentor, <b>MN:</b>	Ištván Papp Phd		
Title, <b>TI:</b>	Implementation of OAuth 2.0 module in IoT system		
Language of text, <b>LT:</b>	Serbian		
Language of abstract, <b>LA:</b>	Serbian		
Country of publication, <b>CP:</b>	Republic of Serbia		
Locality of publication, <b>LP:</b>	Vojvodina		
Publication year, <b>PY:</b>	2017		
Publisher, <b>PB:</b>	Author's reprint		
Publication place, <b>PP:</b>	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, <b>PD:</b> (chapters/pages/ref./tables/pictures/graphs/appendices)	7/41/16/0/28/0/0		
Scientific field, <b>SF:</b>	Electrical Engineering		
Scientific discipline, <b>SD:</b>	Computer Engineering, Engineering of Computer Based Systems		
Subject/Key words, <b>S/KW:</b>	IoT, OAuth 2.0, account linking		
<b>UC</b>			
Holding data, <b>HD:</b>	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, <b>N:</b>			
Abstract, <b>AB:</b>	This paper presents problem with linking user's account for foreign application and his OBLO account for the sake of accessing resources on OBLO service. This problem has been solved using OAuth 2.0 authorization framework and its authorization flows.		
Accepted by the Scientific Board on, <b>ASB:</b>			
Defended on, <b>DE:</b>			
Defended Board, <b>DB:</b>	President:	Ivan Kaštelan, Phd	
	Member:	Nemanja Lukić, Phd	Mentor's sign
	Member, Mentor:	Ištván Papp, Phd	

## **Zahvalnost**

Zahvaljujem se mentoru, prof. dr Ištvanu Papu, dr Mariji Antić i ostalim saradnicima na stručnoj pomoći prilikom izrade rada.

Posebnu zahvalnost dugujem svojoj porodici na neizmernoj podršci koju su mi pružili tokom školovanja.

# **SADRŽAJ**

1.	Uvod .....	1
2.	Teorijske osnove.....	2
2.1	OBLO sistem .....	2
2.1.1	Centralni uređaj .....	3
2.1.2	Okruženje u oblaku.....	4
2.1.3	Klijentske aplikacije .....	4
2.1.4	Komunikacija klijentskih aplikacija sa oblakom i centralnim uređajem.....	4
2.2	OAuth 2.0 okruženje za autentikaciju.....	5
2.2.1	Osnovne definicije.....	5
2.2.1.1	Vlasnik resursa .....	6
2.2.1.2	Poslužilac na kome se resurs nalazi .....	6
2.2.1.3	Autorizacioni poslužilac.....	6
2.2.1.4	Klijent.....	6
2.2.2	Proces autentikacije .....	7
2.2.3	Provera ovlašćenja.....	7
2.2.3.1	Korišćenjem koda ovlašćenja.....	8
2.2.3.2	Implicitni tok .....	9
2.2.3.3	Provera ovlašćenja putem lozinke vlasnika resursa .....	10
2.2.3.4	Provera ovlašćenja putem klijentovih podataka.....	11
3.	Koncept rešenja .....	13
3.1	Ideja realizacije .....	14
4.	Programsko rešenje.....	15
4.1	Delovi oblaka OBLO sistema .....	15
4.2	OAuth 2.0 u OBLO sistemu.....	16
4.2.1	Administratorski deo OBLO sistema .....	16
4.2.2	Implementacija OAuth servisa .....	17
4.2.2.1	Provera ovlašćenja korišćenjem koda ovlašćenja .....	19
4.2.2.2	Kod za osveženje pristupnog koda.....	20

4.2.2.3	Implicitni tok provere ovlašćenja.....	21
4.2.3	Interakcija sa korisnikom .....	21
4.3	Čuvanje podataka u bazi podataka.....	22
5.	Verifikacija .....	24
5.1	Testiranje manipulisanja klijentima od strane administratora .....	24
5.2	Povezivanje naloga .....	25
5.3	Testiranje grafičke korisničke sprege .....	30
6.	Zaključak .....	31
7.	Literatura .....	32

## SPISAK SLIKA

Slika 2.1 OBLO sistem pametne kuće .....	2
Slika 2.2 Protokoli koje OHM podržava .....	3
Slika 2.3 Uprošćen prikaz koraka autentikacije.....	7
Slika 2.4 Tok provere ovlašćenja korišćenjem koda ovlašćenja .....	8
Slika 2.5 Implicitni tok provere ovlašćenja .....	10
Slika 2.6 Tok provera ovlašćenja putem lozinke vlasnika resursa .....	11
Slika 2.7 Tok provere ovlašćenja putem klijentovih podataka .....	11
Slika 3.1 Povezivanje OBLO sistema sa servisima za prepoznavanje govora .....	13
Slika 4.1 Primer podešavanja klijentovih podataka za Google Assistant aplikaciju .....	17
Slika 4.2 Funkcija provere postojanja klijenta i čuvanja transakcije.....	18
Slika 4.3 Funkcija utvrđivanja korisnikove odluke .....	18
Slika 4.4 Funkcija prijave tipa provere ovlašćenja .....	19
Slika 4.5 Deo funkcije koja poziva funkciju <i>authorization_code</i> .....	20
Slika 4.6 Funkcija slanja koda za osveženje pristupnog koda .....	21
Slika 4.7 Funkcija koja obavlja implicitni tok provere ovlašćenja.....	21
Slika 4.8 Informacije o korisniku .....	22
Slika 5.1 <i>thirdParties</i> kolekcija .....	24
Slika 5.2 <i>thirdParties</i> kolekcija .....	25
Slika 5.3 <i>thirdParties</i> kolekcija .....	25
Slika 5.4 Podešavanje Amazon Alexa aplikacije za povezivanje naloga uz proveru ovlašćenja korišćenjem koda ovlašćenja .....	26
Slika 5.5 Podešavanje Amazon Alexa aplikacije za povezivanje naloga uz proveru ovlašćenja korišćenjem implicitnog toka.....	27
Slika 5.6 Amazon web aplikacija .....	27
Slika 5.7 Stranica za odabir centralnog uređaja.....	28
Slika 5.8 Indikacija uspešnog povezivanja naloga .....	28
Slika 5.9 <i>profiles</i> kolekcija .....	29
Slika 5.10 <i>auths</i> kolekcija .....	29

Slika 5.11 Indikacija neuspešnog povezivanja naloga.....	30
Slika 5.12 Podešavanja korisničkog naloga.....	30

## SKRAĆENICE

<b>IoT</b>	- Uređaji povezani na Internet (engl. <b>Internet Of Things</b> )
<b>OHM</b>	- Centralni uređaj OBLO sistema (engl. <b>OBLO Home Manager</b> )
<b>IP</b>	- Protokol mrežnog sloja (engl. <b>Internet Protocol</b> )
<b>BLE</b>	- Komunikacioni protokol (engl. <b>Bluetooth Low Energy</b> )
<b>MQTT</b>	- Protokol za razmenu podataka po principu objave/preplate (engl. <b>Message Queuing Telemetry Transport</b> )
<b>TCP</b>	- Transmisioni kontrolni protokol (engl. <b>Transmission Control Protocol</b> )
<b>API</b>	- Aplikaciona programska sprega (engl. <b>Application Programming Interface</b> )
<b>URL</b>	- Web adresa koja se koristi za lociranje nekog resursa na Internetu (engl. <b>Uniform Resouce Locator</b> )
<b>HTTP</b>	- Protokol prenosa informacija na Internetu ( <b>HyperText Transfer Protocol</b> )
<b>JSON</b>	- Sintaksa za skladištenje i razmenu podataka ( <b>JavaScript Object Notation</b> )
<b>CRUD</b>	- Osnovne funkcije baza podataka ( <b>Create, Read, Update and Delete</b> )

## 1. Uvod

Poslednjih godina se sve više javlja potreba za povezivanjem uređaja u domaćinstvu na Internet, odnosno u sistem pametne kuće. Koncept pametne kuće odnosi se na sistem u kom centralni uređaj (*engl. gateway*) upravlja raznim električnim uređajima, sa ciljem njihovog kontrolisanja i/ili nadzora. Jedan od sistema za upravljanje pametnom kućom je OBLO sistem. Bitna karakteristika pametnih kuća je lako i intuitivno upravljanje uređajima koje se najčešće vrši preko mobilne ili web aplikacije.

U poslednje vreme raste popularnost sistema za prepoznavanje govora što dovodi do potrebe za omogućavanjem upravljanja uređajima u pametnoj kući glasovnim komandama. Danas postoji mnogo takvih servisa dostupnih za korišćenje. Neki od njih su Google Assistant, Microsoft Cortana, Amazon Alexa, Jasper, Julius, itd. Za spregu sa OBLO sistemom korišćeni su Google Assistant i Amazon Alexa zbog svojih brojnih prednosti. Oba servisa su dostupna na Internetu i koriste servise u oblaku koji rešavaju probleme prepoznavanja govora i slanja odgovora nazad do servisa koji ih poziva. Deo OBLO sistema koji komunicira sa oblakom Amazon, odnosno Google servisa, je takođe u oblaku. Kod komunikacije oblak-sa-oblakom dolazi do problema prepoznavanja korisnika koji šalje zahtev. Takođe, stranoj aplikaciji (Amazon ili Google servis) ne sme da se dopusti potpun pristup podacima OBLO sistema.

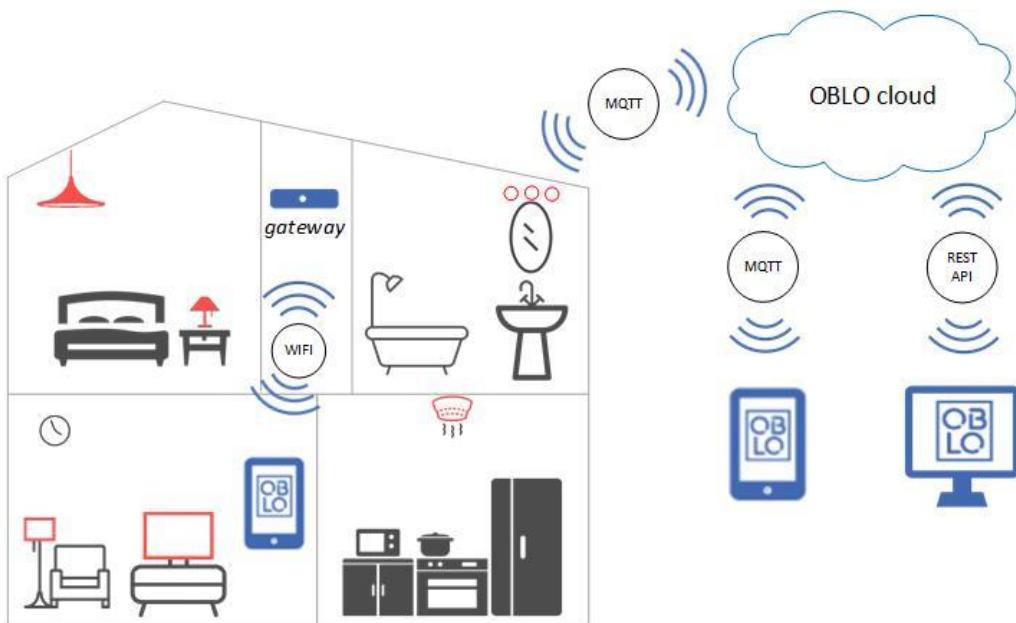
U ovom radu opisano je rešenje povezivanja (autentikacije) korisnikovih naloga na sistemu za prepoznavanje govora sa njegovim OBLO nalogom. Opisano je rešenje problema prepoznavanja korisnika koji šalje zahtev, razlikovanja različitih korisnika i njihovih uređaja u domaćinstvu, kao i rešenje ograničenja pristupa resursima OBLO korisnika, aplikaciji servisa za prepoznavanje govora.

## 2. Teorijske osnove

U ovom poglavlju izložene su teorijske osnove koje su neophodne za razumevanje programske podrške u okviru koje je realizovan modul za povezivanje korisnikovog naloga na sistemu za prepoznavanje govora sa njegovim nalogom na OBLO sistemu. Na početku je opisan OBLO sistem, njegove komponente i način komunikacije među tim komponentama. Zatim je dat opis OAuth 2.0 okruženja za autentikaciju koje je korišćeno za povezivanje naloga.

### 2.1 OBLO sistem

OBLO sistem [1] predstavlja primenu IoT tehnologija u rešenju pametne kuće. U njemu se preko bežične mreže kontrolišu uređaji u njoj.



Slika 2.1 OBLO sistem pametne kuće

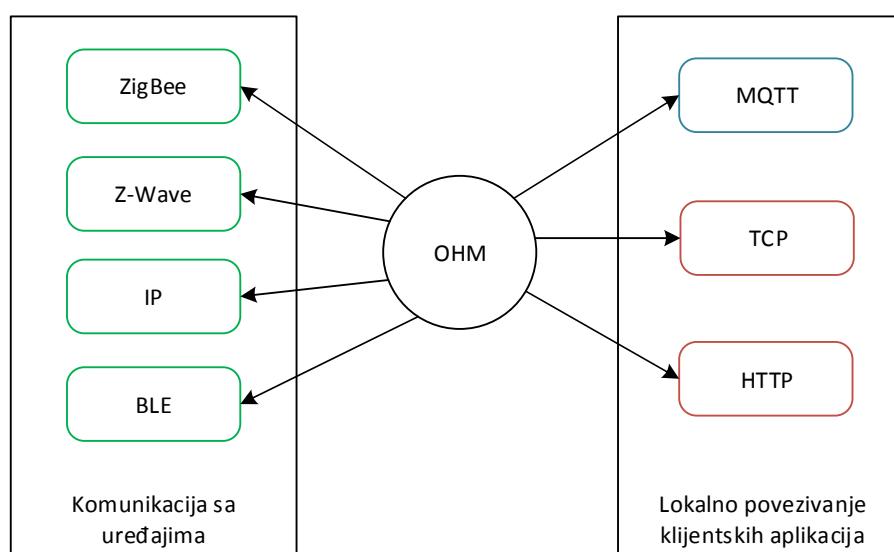
Komponente OBLO sistema predstavljaju:

- Centralni uređaj koji kontroliše ostale uređaje u sistemu
- Oblak sistema koji služi za udaljenu komunikaciju aplikacija sa centralnim uređajem
- Klijentske aplikacije koje predstavljaju grafičku spregu korisnika sa sistemom
- Nodovi, odnosno uređaji u domaćinstvu

### 2.1.1 Centralni uređaj

Uređaji u OBLO sistemu podeljeni su na senzore i aktuatori. Senzori su uređaji koji mere stanje sistema (temperatura, vlažnost, senzori za indikaciju poplave, detektovanja otvorenosti vrata, odnosno prozora...). Senzorima ne može da se upravlja. Sa druge strane, aktuatori predstavljaju uređaje za delovanje. Tu spadaju pametne utičnice, prekidači i sijalice sa potenciometrom, kontroleri scena, odnosno uređaji na čije stanje korisnik može da utiče.

Problem koji se javlja jeste povezivanje različitih uređaja u jedinstvenu mrežu, zbog različitih protokola komunikacije koje ovi uređaji potencijalno koriste. Da bi se ovaj problem rešio, uvodi se pojam centralnog uređaja. Centralni uređaj može da komunicira preko svih podržanih komunikacionih protokola (Zigbee [2], Z-Wave [3], IP, BLE), ali i da uređaje grupiše po njihovoj funkcionalnosti. On kreira mrežu u kojoj su svi uređaji grupisani na osnovu svojih funkcionalnosti, što omogućava primenu scena na sve uređaje, nezavisno od komunikacionog protokola na kom je svaki uređaj baziran. Programska podrška, koja se nalazi na centralnom uređaju i obezbeđuje funkcionalnost kućne automatizacije naziva se OHM (*engl. OBLO Home Manager*). OHM se na oblak sistema povezuje preko MQTT protokola [4], a pruža i HTTPS, TCP i MQTT API za lokalno povezivanje klijentskih aplikacija.



Slika 2.2 Protokoli koje OHM podržava

## 2.1.2 Okruženje u oblaku

Centralni uređaji su najčešće namenski sistemi, integrисани na različite platforme kao što su kontroleri kućne automatizacije, STB uređaji (*set-top-box* – uređaji koji se koriste za prijem i obradu digitalnih zemaljskih televizijskih ili satelitskih signala), računari ili WiFi ruteri. Oni nemaju ekran za interakciju sa korisnicima, te je korisnička sprega za kontrolu sistema izmeštena na druge uređaje, kao što su mobilni uređaji. Veza između centralnih i klijentskih uređaja je bežična. Po pravilu, ona podrazumeva povezivanje ovih uređaja u okviru lokalne mreže posredstvom mrežnih ruteru.

Međutim, u scenariju u kom se korisnik nalazi van lokalne mreže potrebno je obezbediti pristup centralnom uređaju. To omogućava okruženje u oblaku (*engl. cloud*), koje predstavlja korisničku spregu konstantno dostupnu na Internet-u. Svi centralni uređaji različitih korisnika koriste isto okruženje u oblaku, koje vodi računa o razlikovanju korisnika i njihovih odgovarajućih centralnih uređaja.

Oblak se može koristiti na tri načina:

- Od strane administratora – za proveru statusa sistema, upravljanje bazama korisnika, ažuriranje programske podrške, kao i za udaljenu dijagnostiku i podršku korisnicima
- Od strane klijentskih uređaja – za pristupanje uređjima u kući iako korisnik nije na lokalnoj mreži, konfiguraciju centralnog uređaja ili prijem obaveštenja
- Od strane centralnog uređaja – za autentikaciju, ažuriranje trenutne konfiguracije, skladištenje i obradu podataka prikupljenih sa uređaja, slanje obaveštenja korisniku o stanjima uređaja

## 2.1.3 Klijentske aplikacije

Sa korisničke strane, upotreba sistema svodi se na interakciju sa Web aplikacijom, mobilnom aplikacijom (Android ili iOS platforma), ili sa aplikacijom na uređaju za detekciju i raspoznavanje govora (npr. Amazon Echo [5], Google Home [6]). Klijentske aplikacije se povezuju sa centralnim uređajem radi dobavljanja osnovnih informacija o sistemu i slanja zahteva za upravljanjem sistemom.

## 2.1.4 Komunikacija klijentskih aplikacija sa oblakom i centralnim uređajem

Po ulasku centralnog uređaja u mrežu, potrebno je da se on poveže na oblak, nakon čega je korisniku omogućen pristup uređajima u sistemu sa klijentskim aplikacijama. Komunikacija centralnog uređaja odnosno klijentske aplikacije sa platformom u oblaku je bazirana na

MQTT protokolu po principu „preplati se/objavi“ [7]. Centralni uređaj se putem MQTT protokola povezuje na oblak i tako omogućava pristup njegovim funkcijama (npr. daljinska kontrola).

## 2.2 OAuth 2.0 okruženje za autentikaciju

Sa sve bržim razvojem Internet-a raste i potreba za sigurnošću računara, a sa druge strane sigurnosni problemi koji se javljaju postaju sve složeniji. Jedan od bitnijih razloga za povećanu važnost sigurnosti računara predstavljaju povezane aplikacije (*engl. mashups*) koje za svoj rad koriste podatke drugih aplikacija. Ukoliko su za rad jedne aplikacije potrebni podaci iz druge aplikacije javlja se sigurnosni problem – kako omogućiti prvoj aplikaciji ogranicen pristup podacima druge aplikacije? Na primer, ukoliko aplikacija omogućava korisniku da za profilnu sliku iskoristi sliku sa Facebook naloga (koja nije javno dostupna), potrebno je omogućiti da ta aplikacija može samo pristupati slikama sa korisnikovog Facebook naloga. Ukoliko se to ne omogući, aplikacija bi mogla postavljati objave na korisnikovom Facebook nalogu bez njegovog znanja. Neke aplikacije ovaj problem rešavaju tražeći da korisnik obezbedi aplikaciji korisničko ime i lozinku Facebook naloga. Međutim, u tom slučaju aplikacija bi imala pristup korisnikovom nalogu sve dok on ne promeni lozinku, što opet dovodi do problema sa iskorišćavanjem naloga bez korisnikovog znanja. Takođe, ukoliko aplikacija čuva korisnikove podatke u svojoj bazi, potencijalni napadač na aplikaciju bi mogao da „ukrade“ te podatke, te da napadne korisnikov Facebook nalog.

OAuth 2.0 okruženje za autentikaciju [8] omogućava sigurnije deljenje podataka među različitim aplikacijama (web, desktop ili mobilne aplikacije). On dozvoljava aplikaciji da pristupi podacima druge aplikacije, ali u ograničenom broju i u ograničenom vremenskom periodu. Takođe, može se ograničiti i način pristupa, na primer pristup se može ograničiti samo na čitanje podataka, ali ne i na njihovo menjanje.

Cilj autentikacije je dobijanje pristupnog koda (*engl. access token*), koji se šalje između dve aplikacije kao dokaz da su one povezane i da mogu da razmenjuju resurse. Pored pristupnog koda, kako bi se povećala zaštita, moguće je koristiti i kod za osveženje pristupnog koda (*engl. refresh token*). Posle određenog vremenskog perioda, pristupni kod ističe i tada aplikacija koja je primila kodove šalje kod za osveženje prvoj aplikaciji, koja joj zatim generiše novi pristupni kod.

### 2.2.1 Osnovne definicije

Da bi se razdvojilo ko zahteva, a ko odobrava pristup resursima, definisane su četiri uloge u procesu:

1. Vlasnik resursa (*engl. resource owner*)
2. Poslužilac na kome se resurs nalazi (*engl. resource server*)
3. Autorizacioni poslužilac (*engl. authorization server*)
4. Klijent

### **2.2.1.1 Vlasnik resursa**

Vlasnik resursa je korisnik koji odobrava aplikaciji da pristupi resursima koji su povezani na njegov nalog.

### **2.2.1.2 Poslužilac na kome se resurs nalazi**

Poslužilac koji treba da dostavi resurs ukoliko je ceo proces autentikacije uspešno obavljen. U opštem slučaju, poslužilac na kome se resurs nalazi ne proverava ovlašćenja korisnika.

### **2.2.1.3 Autorizacioni poslužilac**

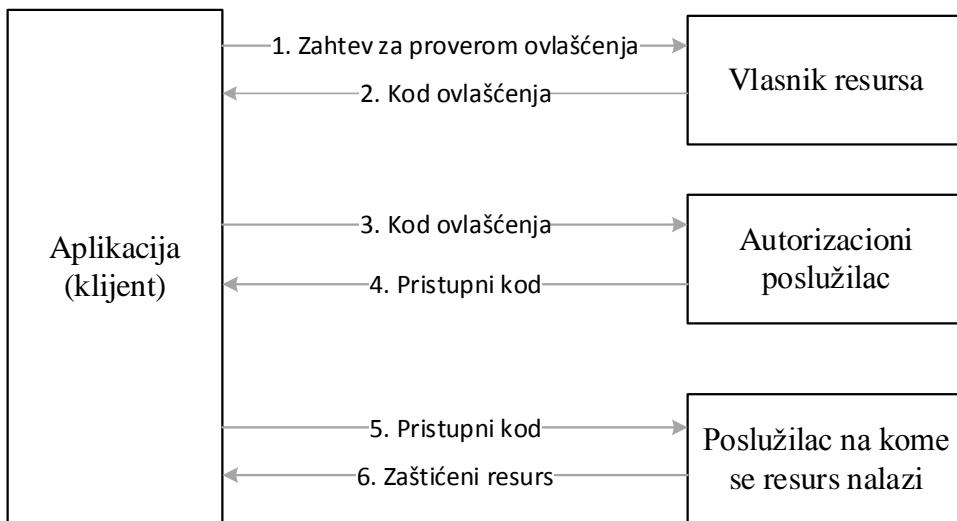
Autorizacioni poslužilac je poslužilac koji proverava ovlašćenja korisnika i izdaje pristupni kod klijentu.

### **2.2.1.4 Klijent**

U OAuth 2.0 terminologiji, klijent je aplikacija koja sa odobrenjem vlasnika resursa šalje zahtev za pristup resursu. Pri registraciji na autorizacioni poslužilac, klijent mora da definiše sledeće:

- Identifikator klijenta (*engl. clientID*) – jedinstveno ime koje je javno i koristi se da identificuje aplikaciju.
- Klijentov tajni ključ (*engl. clientSecret*) – tajni ključ koji se koristi da potvrди identifikaciju klijenta.
- URI za preusmerenje (*engl. redirectionURI*) – URI aplikacije

## 2.2.2 Proces autentikacije



Slika 2.3 Uprošćen prikaz koraka autentifikacije

1. Aplikacija (klijent) zahteva da pristupi korisnikovom resursu.
2. Ukoliko je korisnik odobrio pristup, aplikacija prima kod ovlašćenja (*engl. authorization code*).
3. Aplikacija zahteva pristupni kod, koji predstavlja odobravanje pristupu resursima, od autorizacionog poslužioca, ali se pri tome predstavlja koristeći svoj jedinstveni identifikator i kod ovlašćenja koji je prethodno primila.
4. Ukoliko je identitet aplikacije potvrđen i kod ovlašćenja je validan, autorizacioni poslužilac izdaje pristupni kod aplikaciji. Provera ovlašćenja je ovim završena.
5. Aplikacija sada može da pristupi potrebnim resursima tako što, svaki put kada zatraži resurs, pošalje i pristupni kod kao vid autentifikacije.
6. Ukoliko je pristupni kod validan, poslužilac na kome se resurs nalazi šalje traženi resurs aplikaciji.

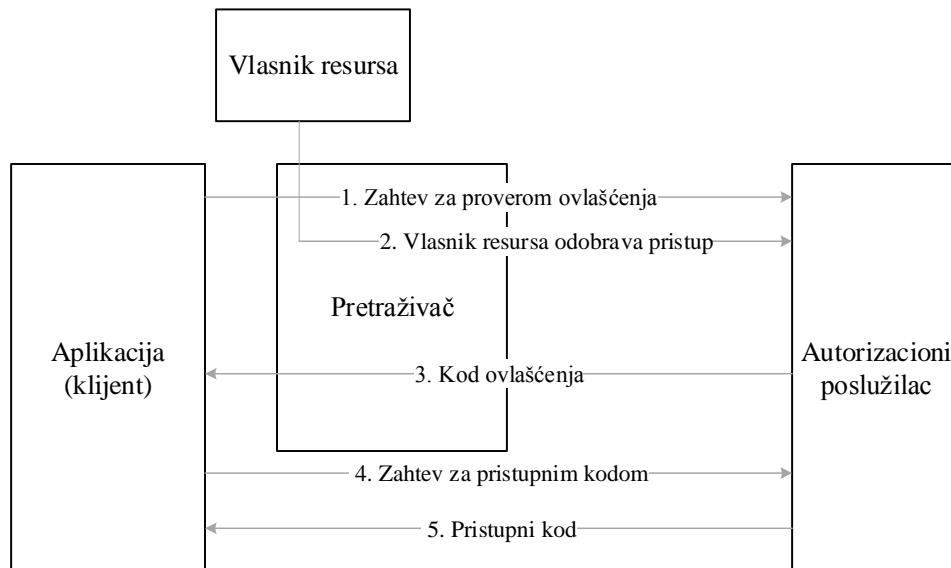
Prethodno navedeni koraci se menjaju u zavisnosti od toga koji tip provere ovlašćenja se koristi.

## 2.2.3 Provera ovlašćenja

U prva četiri koraka *Procesa autentikacije* u Poglavlju 2.2.2 opisan je tok provere ovlašćenja nakon čega aplikacija može da zahteva pristupni kod. U zavisnosti od toga na koji način aplikacija zahteva resurs, kao i od toga u kom scenariju se autentikacija obavlja i nivoa potrebne zaštite, OAuth 2.0 definiše četiri načina provere ovlašćenja aplikacije.

### 2.2.3.1 Korišćenjem koda ovlašćenja

Provera ovlašćenja korišćenjem koda ovlašćenja (*engl. Authorization Code*) predstavlja najpopularniji tip provere ovlašćenja jer nudi najveću bezbednost. Takođe, pored pristupnog koda, koristi i kod za osveženje pristupnog koda. Pošto se tok provere zasniva na preusmerenju, aplikacija mora biti sposobna da komunicira sa poslužiocem na kome se resurs nalazi i da prima zahteve, preko preusmeravanja, od autorizacionog poslužioca.



Slika 2.4 Tok provere ovlašćenja korišćenjem koda ovlašćenja

1. Klijent započinje komunikaciju tako što šalje korisnika na stranicu za autorizaciju gde je uglavnom potrebno da korisnik unese svoje podatke kako bi se ulogovao. (Npr. ukoliko se radi o autentikaciji sa Facebook-om, klijent bi bio preusmeren na početnu stranicu Facebook-a koja bi od njega zahtevala da unese korisničko ime i lozinku). Na URL stranice na koju šalje korisnika, klijent dodaje svoj identifikator kao i URL za preusmeravanje na koji će autorizacioni poslužilac vratiti korisnika kada se provera ovlašćenja završi uspešno.

Primer:

[https://mydomain.com/oauth/authorize?response\\_type=code&client\\_id=ID&redirect\\_uri=URL&scope=read](https://mydomain.com/oauth/authorize?response_type=code&client_id=ID&redirect_uri=URL&scope=read)

- <https://mydomain.com/oauth/authorize> - stranica za autorizaciju
- *response\_type* – specificira koji tip provere ovlašćenja se koristi, *code* označava da je reč o proveri korišćenjem koda ovlašćenja
- *client\_id* – identifikator klijenta
- *scope* – nivo pristupa koji aplikacija zahteva, *read* označava da aplikacija može samo da čita podatke

2. Autorizacioni poslužilac proverava klijentove podatke i utvrđuje da li vlasnik resursa dozvoljava ili odbija klijentov zahtev za pristup.
3. Ukoliko korisnik odbije pristup resursima, autentifikacija se zaustavlja. Prepostavljući da je korisnik odobrio pristup, autorizacioni poslužilac preusmerava pretraživač na klijenta koristeći URI za preusmerenje koji je klijent dostavio tokom svoje registracije. URI za preusmerenje će takođe da sadrži kod ovlašćenja.
4. Nakon preusmerenja, klijent traži pristupni kod od autorizacionog poslužioca tako što mu šalje kod ovlašćenja koji je prethodno primio zajedno sa svojom identifikacijom koja ovaj put uključuje i klijentov tajni ključ.
5. Ukoliko je vraćeni kod odgovarajući, autorizacioni poslužilac odgovara šaljući pristupni kod i, opcionalno, kod za osveženje pristupnog koda.

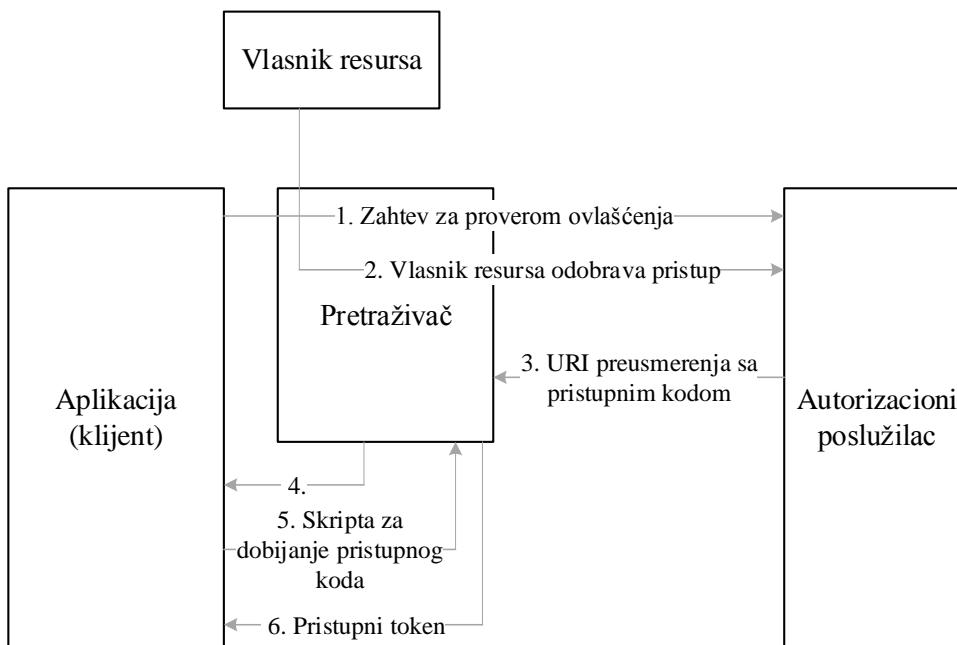
Primer odgovora:

```
{
  "access_token": "123456789132456789123456789"
  "token_type": "bearer" [9]
  "expires_in": 36000
  "refresh_token": "987654321"
}
```

Kada je prošao proveru ovlašćenja, klijent može da zahteva resurse koristeći pristupni kod dok god je on validan. Kada pristupni kod istekne, aplikacija šalje kod za osveženje, kako bi dobila novi pristupni kod, naravno, u slučaju da je kod za osveženje dostavljen u prethodnom koraku.

### **2.2.3.2 Implicitni tok**

Implicitni tok provere ovlašćenja (*engl. Implicit flow*) služi za dobijanje pristupnog koda, ali ne i koda za osveženje. Ovaj tok je takođe zasnovan na preusmerenju, ali za razliku od prethodnog, gde klijent šalje odvojene zahteve za identifikaciju i za pristupni kod, u implicitnom toku, klijent dobija pristupni kod kao odgovor na uspešnu identifikaciju. Samim tim, ovaj tok provere ovlašćenja ima manju sigurnost.



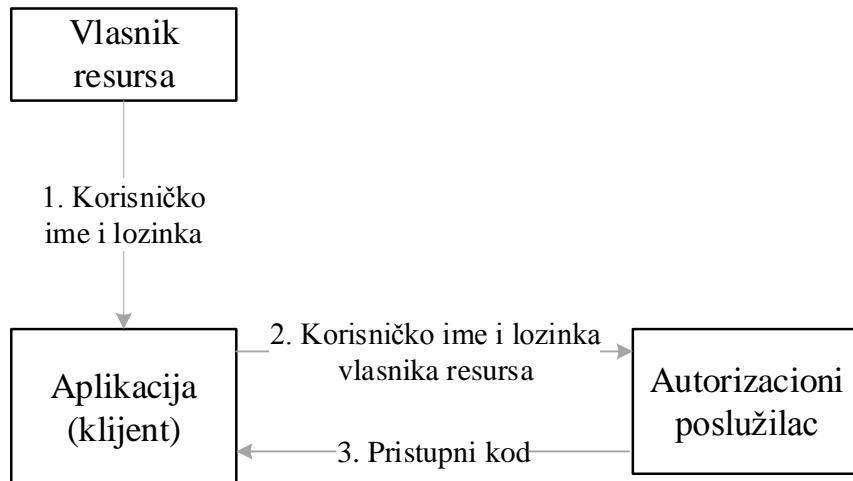
Slika 2.5 Implicitni tok provere ovlašćenja

1. Klijent šalje korisnika na stranicu za autorizaciju. Na URL te stranice dodaje svoj identifikator i URI za preusmerenje. U ovom slučaju *response\_type* će biti postavljeno na „*token*“.
2. Autorizacioni poslužilac proverava klijentove podatke i utvrđuje da li vlasnik resursa dozvoljava ili odbija klijentov zahtev za pristup.
3. Ukoliko korisnik odbije pristup resursima, autentikacija se zaustavlja. Pretpostavljajući da je korisnik odobrio pristup, autorizacioni poslužilac preusmerava pretraživač na klijenta koristeći URI za preusmerenje koji je klijent dostavio tokom svoje registracije. URI za preusmerenje sadrži i pristupni kod u fragmentu.
4. Pretraživač usmerava vlasnika resursa nazad do klijenta kroz URI za preusmerenje i šalje zahtev poslužiocu na kome se nalazi resurs. Ovaj zahtev ne uključuje fragment sa pristupnim kodom jer ga je pretraživač zadržao kod sebe [10].
5. Aplikacija šalje skriptu koja je potrebna za pristup adresi resursa i dobijanje pristupnog koda i ostalih parametara koji se nalaze u fragmentu.
6. Pretraživač izvršava skriptu i dobija pristupni kod.
7. Pretraživač prosleđuje pristupni kod klijentu.

### 2.2.3.3 Provera ovlašćenja putem lozinke vlasnika resursa

Provera ovlašćenja putem lozinke vlasnika resursa (*engl. Resource Owner Password Credentials Grant*) je pogodna u slučaju da vlasnik resursa ima potpuno poverenje u klijenta

(npr. reč je o aplikaciji sa visokim privilegijama u sistemu). Autorizacioni poslužilac treba da obrati pažnju kada odobrava ovaj tip provere ovlašćenja i treba da ga dozvoli samo ukoliko nije moguće implementirati prethodna dva toka provere.



Slika 2.6 Tok provera ovlašćenja putem lozinke vlasnika resursa

1. Vlasnik šalje svoje podatke (korisničko ime i lozinku) klijentu.
2. Klijent zahteva pristupni kod od autorizacionog poslužioca šaljući podatke vlasnika resursa koje je prethodno primio.

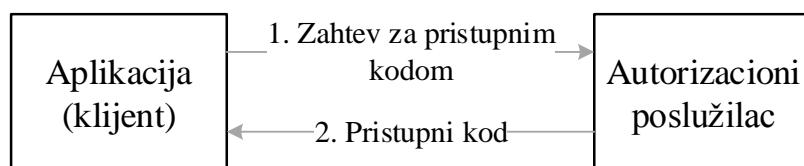
Primer zahteva:

```
{
  "grant_type": "password"
  "username": "username"
  "password": "password"
}
```

3. Autorizacioni poslužilac proverava ovlašćenja klijenta i podatke vlasnika resursa i, ukoliko je sve validno, šalje pristupni kod klijentu.

#### 2.2.3.4 Provera ovlašćenja putem klijentovih podataka

Kod provere ovlašćena putem klijentovih podataka (*engl. Client Credentials Grant*) klijent može da dobije pristupni kod šaljući samo svoje podatke. Ovo se radi kada klijent želi da pristupi podacima koji su zaštićeni i pod njegovom kontrolom. Ovaj tip provere ovlašćenja se koristi ako i samo ako se klijentskoj aplikaciji u potpunosti „veruje“.



Slika 2.7 Tok provere ovlašćenja putem klijentovih podataka

1. Klijent zahteva pristupni kod od autorizacionog poslužioca šaljući svoje podatke.

Primer zahteva:

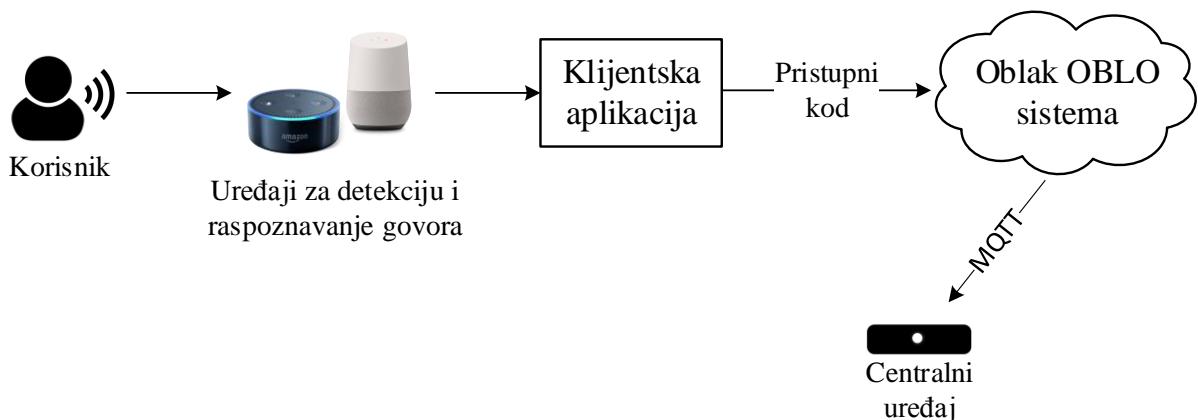
```
{  
    "grant_type": "client_credentials"  
}
```

2. Autorizacioni poslužilac proverava primljene podatke i, ukoliko su ispravni, šalje pristupni kod klijentu.

### 3. Koncept rešenja

U nastavku data je motivacija za implementiranjem OAuth 2.0 modula u OBLO sistem, kao i kratak opis realizacije ovog modula.

OBLO sistem je proširen aplikacijom za zadavanje komandi glasom. Prikaz arhitekture je dat na sledećoj slici.



Slika 3.1 Povezivanje OBLO sistema sa servisima za prepoznavanje govora

Klijenti, čije povezivanje je implementirano u OBLO sistem su Amazon i Google, sa svojim servisima za prepoznavanje govora – Amazon Alexa [11] i Google Assistant [12], respektivno. Međutim, sistem je prilagodljiv za autentikaciju i sa drugim servisima ukoliko je to potrebno. Svaki od servisa za prepoznavanje govora ima klijentsku aplikaciju koja obrađuje izgovorene naredbe i šalje ih OBLO sistemu kako bi ih fizički realizovao. Pored obrade naredbe, aplikacije se brinu o tome da se u sistem ne šalju naredbe koje nisu definisane. Na primer, aplikacija proverava da li postoji uređaj koji korisnik želi da kontroliše, odnosno čije stanje želi da očita.

Kako bi svaki klijent OBLO sistema mogao da upravlja svojim uređajima uz pomoć govora, potrebno je povezati klijentsku aplikaciju, koja se bavi obradom izgovorene naredbe,

sa korisnikovim OBLO nalogom. U suprotnom, OBLO sistem ne bi imao informaciju koji korisnik želi da pristupi kojem centralnom uređaju pa na kraju ni kojem krajnjem uređaju korisnik želi da pristupi, jer je moguće da na dva centralna uređaja postoje dva uređaja sa istim imenom. Zbog svih prednosti koje su navedene u Poglavlju 2.2 za povezivanje ova dva sistema implementiran je OAuth 2.0 modul u okviru OBLO sistema. Jedan od najbitnijih zahteva sistema je svakako bezbednost korisnikovih informacija, te je implementirana provera ovlašćenja korišćenjem koda ovlašćenja i implicitni tok provere ovlašćenja.

Koristeći terminologiju OAuth 2.0 okruženja za autentikaciju, sistem je podeljen na:

1. Vlasnika resursa kojeg predstavlja korisnik OBLO sistema
2. Poslužioca na kome se resurs nalazi kog prestavlja OBLO sistem
3. Autorizacioni poslužilac koji je takođe implementiran u okviru OBLO sistema
4. Klijent koji je, u stvari, aplikacija koja se bavi obradom glasovne komande.

### 3.1 Ideja realizacije

Cela realizacija autentikacije smeštena je na oblak OBLO sistema. Ideja je da se na nivou celog sistema definišu određeni OAuth 2.0 klijenti sa kojima sistem može da se povezuje. Te klijente može da doda samo korisnik koji ima administratorska prava. Međutim, ukoliko nekoliko klijenata postoji u okviru celog sistema, to ne znači da je svaki korisnik povezan sa svakim klijentom. Naprotiv, korisnik sam bira koji servis želi da koristi i, samim tim, koje klijente želi da poveže sa svojim nalogom. Na primer, ukoliko jedan korisnik želi da koristi Amazon servis za prepoznavanje govora, on će se povezivati samo sa Amazon-om, dok neki drugi korisnik može da se povezuje samo sa Google servisom, neki treći može da ima oba, i Google i Amazon, a četvrti, pored ova dva može da ima još neki treći servis za prepoznavanje govora. Nakon što korisnik odabere servis sa kojim želi da se poveže i započne autentikaciju, potrebno je da dopusti aplikaciji pristup resursima. Međutim, klijent mora da izabere jedan centralni uređaj čijim uređajima će moći upravljati.

Ukoliko korisnik odluči da ne želi više da koristi usluge jednog servisa za prepoznavanje govora ili želi da kontroliše drugi centralni uređaj, omogućeno mu je da obriše klijenta sa svog naloga.

## 4. Programsко rešenje

Kao što je već rečeno, OAuth 2.0 modul za autentikaciju OBLO sistema sa sistemima za prepoznavanje govora implementiran je u okviru oblaka OBLO sistema.

### 4.1 Delovi oblaka OBLO sistema

Oblak OBLO sistema je implementiran po principu mikro servisa, gde su različiti moduli odvojeni po principu logike koju obrađuju i međusobno komuniciraju putem HTTP protokola.

Osnovne funkcije oblaka razdvojene su po sledećim mikro servisima:

- Selfcare – deo namenjen korisnicima
- Backoffice – deo namenjen administratorima OBLO sistema
- Webshop – prodavnica proizvoda dostupnih korisnicima sistema kao i potencijalnim kupcima
- CRM (*engl. Customer Relationship Management*) – upravljanje korisničkim nalozima, svaki korisnik sadrži jedinstveni identifikator koji ga jednoznačno određuje u okviru celog sistema
- Auth (*engl. authorization*) – servis zadužen za proveravanje prava pristupa korisnika
- Notifier – dostavljanje obaveštenja korisnicima
- CDS (*engl. Content Delivery Service*) - upravljanje digitalnim sadržajima namenjenim korisnicima sistema
- CDN (*engl. Content Delivery Node*) – usko povezan sa CDS, služi za skladištenje i dostavljanje sadržaja korisnicima

- UDM (*engl. User Device Management*) – komunikacija sa centralnim uređajem korisnika

## 4.2 OAuth 2.0 u OBLO sistemu

OAuth 2.0 modul dodat je kao novi servis u oblaku. Servis je implementiran na Node.js platformi koristeći biblioteku otvorenog koda *oauth2orize* [13].

Kako bi se omogućila celokupna implementacija autentikacije, izmene su napravljene u nekoliko modula.

### 4.2.1 Administratorski deo OBLO sistema

Na administratorkoj strani oblaka uvedena je mogućnost dodavanja novog klijenta (aplikacija koja se autentikuje sa OBLO sistemom) u okviru celog sistema. Ukoliko klijent nije dodat u sistem od strane administratora, neće moći ni započeti autentikaciju. Manipulisanje globalnim klijentom implementirano je u okviru *client* servisa na oblaku. Omogućene su sledeće funkcije nad klijentom:

- Administrator želi da doda novog klijenta u sistem.

Šalje se HTTP zahtev:

POST /clients/client HTTP/1.1

Nakon pristizanja zahteva poziva se metoda *createClient* koja se nalazi u servisu *client* za dodavanje klijenta.

Metoda *createClient* pokupi podatke o novom klijentu koje je administrator uneo – identifikator klijenta i tajni ključ i unosi ih u MongoDB [14] kao novi dokument u kolekciji *thirdParties*.

- Administrator želi da izmeni podatke postojećem klijentu.

Šalje se HTTP zahtev:

PATCH /clients/client/:id HTTP/1.1

Nakon pristizanja zahteva poziva se metoda *updateClient* koja se nalazi u servisu *client* kojoj se prosleđuju novi klijentovi podaci kako bi ih zamenila sa starim u bazi podataka.

- Administrator želi da obriše postojećeg klijenta.

Šalje se HTTP zahtev

DELETE /clients/client/:id HTTP/1.1

Nakon pristizanja zahteva poziva se metoda *deleteClient* kojoj se prosleđuje identifikator klijenta kojeg treba obrisati iz sistema, odnosno baze podataka.

- Pregled postojećih klijenata

Šalje se HTTP zahtev

```
GET clients/client?user=admin HTTP/1.1
```

Nakon pristizanja zahteva poziva se metoda *getClients* kojoj se prosleđuje grupa korisnika (*admin*) na osnovu koje se dobavlja spisak svih klijenata. Ukoliko se ovoj metodi prosledi grupa *user* prikazaće se spisak klijenata autentikovanih sa konkretnim korisnikom.

#### 4.2.2 Implementacija OAuth servisa

Autentikacija klijenta je implementirana u okviru *oauth* servisa na oblaku.

Pre započinjanja procesa autentikacije, klijent mora da podesi svoje podatke – identifikator, tajni ključ, URI za preusmerenje, kao i putanje na OBLO sistemu koje treba da „pogodi“ kako bi dobio potrebne kodove.

The screenshot shows the configuration interface for a Google Cloud Platform OAuth client. The 'Grant type' is set to 'Authorization code'. The 'Client information' section contains the 'Client ID' (google-oblo) and 'Client secret' (redacted). The 'Authorization URL' is set to <https://aisha-test.rt-rk.com/login>, and the 'Token URL' is set to <https://aisha-test.rt-rk.com/oauth2/token>.

Slika 4.1 Primer podešavanja klijentovih podataka za Google Assistant aplikaciju

Kada korisnik započne povezivanje naloga, klijentska aplikacija šalje HTTP zahtev

POST /login HTTP/1.1

uz koji šalje svoje parametre – identifikator, tip provere ovlašćenja i URI za preusmerenje. Nakon pristizanja zateva na oblak pretraživač otvara početnu stranicu OBLO sistema na kojoj je potrebno da korisnik unese svoje podatke za pristup sistemu. Kada korisnik unese podatke, *auth* servis oblaka je zaslužan za proveru validnosti podataka i za kreiranje sesije za konkretnog korisnika. Takođe, uz proveru korisnikovih podataka, proveravaju se i podaci klijenta, odnosno da li je admin prethodno odobrio autentikaciju konkretnom klijentu – da li on postoji u sistemu. Ukoliko su klijentovi podaci validni, potrebno je generisati transakciju za proveru ovlašćenja. Paket *oauth2orize* zahteva sesiju, kako bi se propisno obavila provera ovlašćenja, međutim u okviru OBLO sistema, sesija se odnosi na potvrdu da je korisnik prijavljen na sistem. Ovaj problem je rešen tako što se transakcija potrebna *oauth* modulu čuva u bazi podataka, iščitava iz nje kad god je potrebna i briše kada se komunikacija završi. Proveru postojanja klijenta u sistemu i pamćenja oauth sesije u bazi obavlja funkcija *authorization*.

```
exports.authorization = [
  server.authorize(function(clientID, redirectURI, done) {
    ThirdParty.findOne({'id': clientID}, function (err, thirdParty) {
      if (err)
        return done(err);

      return done(null, thirdParty, redirectURI);
    });
  }),

  function(req, res, next) {
    oauth2transaction.saveTransaction(req, res, next);
  }
];
```

Slika 4.2 Funkcija provere postojanja klijenta i čuvanja transakcije

Ukoliko se korisnik uspešno prijavi na sistem, pojavljuje mu se stranica na kojoj treba da izabere za koji centralni uređaj odobrava glasovnu kontrolu.

Funkcija *decision* koja poziva biblioteku *oauth2orize* se brine o tome da li je korisnik dozvolio dalju autentikaciju.

```
exports.decision = [
  server.decision()
];
```

Slika 4.3 Funkcija utvrđivanja korisnikove odluke

Klikom na dugme *CANCEL* korisnik zaustavlja proces autentikacije, odnosno ne dozvoljava povezivanje sa klijenom. Sa druge strane, ukoliko izabere jedan od ponuđenih centralnih

uređaja, korisnik dozvoljava autentikaciju i, samim tim, dozvoljava klijentu da koristi podatke kao što je jedinstveni identifikator centralnog uređaja koji je odabrao. Nakon odabiranja uređaja *oauth* servis započinje proces autentikacije. Kao što je objašnjeno u Poglavlju 2.2.3, postoji više vrsta provere ovlašćenja klijenta. U OBLO sistemu su implementirane dve:

- korišćenjem koda ovlašćenja
- implicitni tok

Sam klijent odlučuje koju proveru ovlašćenja će koristiti.

#### **4.2.2.1 Provera ovlašćenja korišćenjem koda ovlašćenja**

Kada klijent odluči da koristi proveru ovlašćenja putem koda ovlašćenja, *oauth* servis mora da ga prijavi za taj tip provere. To obavlja srednji sloj *oauth2orize* biblioteke. Nakon prijave, *oauth* servis generiše kod ovlašćenja koji će se kasnije koristiti za kreiranje pristupnog koga.

```
server.grant(oauth2orize.grant.code(function(client, redirectURI, user, ares, done) {
    var value = crypto.randomBytes(16).toString("hex");

    var ac = new Code({
        value: value,
        clientId: client.id,
        redirectUri: redirectURI,
        userId: user._id
    });

    ac.save(function(err) {
        if (err) {
            return done(err);
        }
        return done(null, value);
    });
}));
```

Slika 4.4 Funkcija prijave tipa provere ovlašćenja

Nakon prijave tipa provere ovlašćenja i kreiranja koda ovlašćenja u bazi podataka, potrebno je poslati kod klijentu. Razmenu koda ovlašćenja izmedju *oauth* servisa i klijenta obavlja funkcija *authorization\_code* srednjeg sloja *oauth2orize* biblioteke.

```

server.exchange(oauth2orize.exchange.code(function(client, code, redirectURI, done) {
    Code.findOne({
        value: code
    }, function(err, code) {
        if (err) return done(err);

        if (redirectURI != code.redirectUri) return done(null, false);

        var refreshToken = crypto.randomBytes(256).toString("hex");
        var accessToken = crypto.randomBytes(256).toString("hex");

        var at = new Token({
            accessToken: accessToken,
            owner: code.clientId,
            gtwSerial: oauth2transaction.serialNumber
        });

        var rt = new RefreshToken({
            refreshToken: refreshToken,
            userId: code.userId,
            clientId: code.clientId
        });

        ...
        return done(null, at.accessToken, rt.refreshToken, {expires_in: expiration_time});
    });
}));

```

Slika 4.5 Deo funkcije koja poziva funkciju *authorization\_code*

Kada se razmeni kod ovlašćenja sa klijentom, potrebno je proveriti da li je klijent vratio kod koji odgovara poslatom kodu pri zahtevu da mu se dodeli pristupni kod (Poglavlje 2.2.3.1 tačka 4). Ukoliko je kod validan generišu se novi pristupni kod i kod za osveženje pristupnog koda koji su vezani za konkretnog korisnika i klijenta. Pored ova dva koda šalje se i vremenski period validnosti poslogog pristupnog koda (*expiration\_time*). U modelu pristupnog koda se čuva i identifikator centralnog uređaja sa kojim je korisnik odobrio povezivanje. Kod ovlašćenja koji je iskorišćen se briše iz baze podataka kako ne bi mogao biti nanovo iskorišćen za generisanje pristupnog koda.

Kao što je objašnjeno u Poglavlju 2.2.3.1 da bi dobio pristupni kod, klijent mora da se preusmeri na putanju za dobijanje pristupnog koda. U ovom slučaju, klijent mora da pošalje HTTP zahtev

POST /oauth2/token HTTP/1.1

Kada zahtev pristigne na oblak, funkcija *token* u okviru *oauth* servisa je zaslužna za dalje slanje pristupnog koda klijentu. Funkcija *token* poziva srednji sloj biblioteke *oauth2orize* koji obavlja slanje pristupnog koda.

#### 4.2.2.2 Kod za osveženje pristupnog koda

Nakon isteka vremenskog perioda u kom je pristupni kod validan on se briše iz baze podataka. Kako bi klijent koji je već obavio autentikaciju mogao i dalje da pristupa

dozvoljenim resursima, potrebno mu je dostaviti kod za osveženje pristupnog koda. Klijent šalje kod za osveženje, koji je dobio kada i pristupni kod, i ukoliko taj kod zaista postoji u bazi podataka, *oauth* servis generiše novi pristupni kod. Ova razmena se obavlja svaki put kada novom pristupnom kodu istekne vremenski period validnosti.

```
server.exchange(oauth2orize.exchange.refreshToken(function (client, refreshToken, scope, done) {
    RefreshToken.findOne({refreshToken: refreshToken}, function (err, token) {
        if (err) return done(err);
        if (!token) return done(null, false);

        var newAccessToken = crypto.randomBytes(256).toString("hex");

        ...

        done(null, newAccessToken, refreshToken, {expires_in: expiration_time});
    })
}));
```

Slika 4.6 Funkcija slanja koda za osveženje pristupnog koda

#### 4.2.2.3 Implicitni tok provere ovlašćenja

U Poglavlju 2.2.3.2 je objašnjeno da implicitni tok ne generiše kod ovlašćenja već, nakon potvrđene validacije korisnika i klijenta, generiše pristupni kod. Takođe, kod implicitnog toka, klijent ne mora da se prijavljuje za korišćenje tog tipa provere ovlašćenja.

```
server.grant(oauth2orize.grant.token(function (client, user, ares, done) {
    var accessToken = crypto.randomBytes(256).toString("hex");

    var at = new Token({
        accessToken: accessToken,
        owner: client.id,
        gtwSerial: oauth2transaction.serialNumber
    });

    ...

    return done(null, accessToken);
}));
```

Slika 4.7 Funkcija koja obavlja implicitni tok provere ovlašćenja

Metoda *grant* obavlja čitav proces slanja pristupnog koda klijentu, jer se klijent ne preusmerava na putanju na kojoj dobija pristupni kod, kao u prethodnom slučaju (nema koda ovlašćenja, pa nema razloga za preusmerenjem).

#### 4.2.3 Interakcija sa korisnikom

Kako bi korisnik imao grafičku prezentaciju koja mu govori da li je na njegov nalog povezan neki klijent, u okviru web aplikacije OBLO sistema dodat je prikaz liste povezanih klijenata.

The screenshot shows a user profile editing interface. At the top, the user's name 'Milica Matic' is displayed. Below it, there are fields for 'First name' (Milica) and 'Last name' (Matic). A language selection dropdown shows 'English'. A large blue user icon is on the left.

The interface has three tabs: 'INFO' (selected), 'DATA', and 'ADVANCED'. Under 'INFO', there is a 'Address' section with fields for 'Address\*', 'City\*', 'Zip\*', and 'Country\*'. There are input fields with placeholder text '0 / 40' and '0 / 40'. Below this are sections for 'Clients' (listing 'AMAZON-OBLO', 'GOOGLE-OBLO', and 'AMAZON') and 'Contacts' (listing an email 'milica.matic@rt-rk.com' and a phone number 'New phone'). At the bottom, there is a link 'ADD NEW EMAIL'.

Slika 4.8 Informacije o korisniku

Korisnik može da obriše postojećeg klijenta ukoliko ne želi više da mu dopusti pristup resursima. Klikom na dugme za brisanje klijenta šalje se HTTP zahtev:

**DELETE /oauth2/token/:id HTTP/1.1**

Nakon što primi zahtev, *oauth* servis poziva metodu *deleteToken* koja briše iz baze podataka pristupni kod i, ukoliko postoji, kod za osveženje pristupnog koda.

### 4.3 Čuvanje podataka u bazi podataka

Svi bitni podaci sistema čuvaju se u MongoDB [14]. MongoDB predstavlja vodeću NoSQL bazu podataka. Podaci se čuvaju kao JSON dokumenti sa dinamičkim šemama. Ova baza podataka čini pohranjivanje podataka u aplikacijama jednostavnijom i bržom. Podaci u MongoDB su organizovani po kolekcijama. Jedna kolekcija može da se sastoji od jednog ili više dokumenata koji se sastoje od skupa parova ključ – vrednost.

Kako bi se olakšao rad sa MongoDB, koristi se *mongoose* [15]. *Mongoose* predstavlja Node.js paket koji dozvoljava pristup bazi podataka radi obavljanja CRUD operacija (operacije dodavanja, čitanja, izmene i brisanja podataka iz baze podataka). Rešenje je predstavljeno u obliku šema, koje opisuju stvarne podatke u bazi, i modela, koji predstavljaju operacije koje mogu da se obavljaju nad tim podacima. Šeme odgovaraju kolekcijama u MongoDB, a izvršavanjem akcije čuvanja podataka (*ime\_šeme.save()*) u bazu se dodaje novi dokument.

Kako bi implementacija OAuth 2.0 modula u OBLO sistemu bila uspešna, dodato je nekoliko *mongoose* šema:

1. *thirdParties* – Ova šema služi za čuvanje podataka globalnog klijenta. Sadrži sledeća polja:
  - *id* – jedinstveni identifikator klijenta
  - *secret* – tajni ključ klijenta
2. *grantCodes* – Ova šema služi za privremeno čuvanje koda ovlašćenja. Sadrži sledeća polja:
  - *value* – kod ovlašćenja
  - *redirectUri* – URI za preusmerenje koji se dobije od klijenta koji pokušava da se autentikuje
  - *userId* – identifikator korisnika sa čijim nalogom klijent pokušava da se autentikuje
  - *clientId* – identifikator klijenta koji pokušava da se autentikuje
3. *oauth2transactions* – Ova šema služi za čuvanje sesija provere ovlašćenja (pogledati Poglavlje 4.2.2). Sadrži sledeća polja:
  - *user* – identifikator korisnika sa čijim nalogom klijent pokušava da se autentikuje
  - *tid* – identifikator konkretnе transakcije
  - *transaction* – objekat koji opisuje transakciju

Pored novih šema koje su dodata, izmenje su i neke postojeće:

4. *Profiles* – Ova šema sadrži podatke o korisnicima. Proširena je dodavanjem sledećeg polja:
  - *clientName* – niz koji sadrži imena klijenata sa kojima je korisnik trenutno autentikovan
5. *Auths* – Ova šema sadrži poverljive podatke o korisnicima (kao što je, na primer, lozinka naloga). Proširena je dodavanjem sledećeg polja:
  - *accessToken* – niz dokumenata koji skladište vrednosti pristupnih kodova koji se šalju klijentu nakon uspešne autentikacije. Kako bi se u svakom trenutku znalo koji pristupni kod je vezan za kojeg klijenta i centralni uređaj, svaki dokument u ovom nizu, pored vrednosti pristupnog ključa, sadrži i polja *owner*, odnosno identifikator povezanog klijenta, i *gtwSerial*, odnosno broj centralnog uređaja kojem je dozvoljen pristup.

## 5. Verifikacija

Da bi se utvrdila ispravnost i stabilnost sistema, on mora biti podvrgnut različitim testovima. U ovom poglavlju opisani su testovi sprovedeni nad sistemom, kao i rezultati tih testova. Testiranje je obavljenom onim redom kojim se projekat razvijao. Prva faza obuhvata manipulisanje klijentima na nivou celog sistema od strane administratora. Druga i treća faza testiraju autentikaciju klijenta sa OBLO sistemom korišćenjem dva tipa provere ovlašćenja klijenta. Poslednja, četvrta faza obuhvata testiranje grafičke korisničke sprege.

### 5.1 Testiranje manipulisanja klijentima od strane administratora

Pod manipulacijom klijentima od strane administratora podrazumeva se dodavanje klijenta u okviru celog OBLO sistema, izmena postojećeg klijenta i brisanje klijenta.

Indikacija uspešnog dodavanja klijenta je njegov upis u bazu podataka u kolekciju *thirdParties*.

thirdparties		
0.010 s	5 Docs	
Key	Value	Type
▶  (1) ObjectId("591d78a4acf657a825c2732d")	{ 4 fields }	Document
▶  (2) ObjectId("5926ad01583abcf1159c68f9")	{ 4 fields }	Document
▶  (3) ObjectId("594a5b31c2287fb3a5b1466")	{ 4 fields }	Document
▶  (4) ObjectId("594b9b96b66e95687d28a1cb")	{ 4 fields }	Document
◀  (5) ObjectId("59524a6e0f1cf2423db9c454")	{ 4 fields }	Document
<code>_id</code>	ObjectId("59524a6e0f1cf2423db9c454")	ObjectId
<code>id</code>	amazon	String
<code>secret</code>	123456789	String
<code>__v</code>	0	Int32

Slika 5.1 *thirdParties* kolekcija

Dokumenti kolekcije *thirdParties* sadrže podatke za svakog globalnog klijenta. Kada administrator doda klijenta u sistem, u kolekciju *thirdParties* se dodaje novi dokument koji

sadrži identifikaciju (*id*) i tajni ključ klijenta (*secret*), kao i identifikator dodatog dokumenta (*\_id*). kao što je prikazano na slici 5.1.

Nakon izmene klijentovih podataka, baza podataka se osvežava, tako da se u njoj uvek nalaze najazurniji podaci – Slika 5.2.

thirdparties 0.012 s   5 Docs		
Key	Value	Type
▶ (1) ObjectId("591d78a4acf657a825c2732d")	{ 4 fields }	Document
▶ (2) ObjectId("5926ad01583abcf1159c68f9")	{ 4 fields }	Document
▶ (3) ObjectId("594a5b31c2287fb3a5b1466")	{ 4 fields }	Document
▶ (4) ObjectId("594b9b96b66e95687d28a1cb")	{ 4 fields }	Document
◀ (5) ObjectId("59524a6e01cf2423db9c454")	{ 4 fields }	Document
✍ _id	ObjectId("59524a6e01cf2423db9c454")	ObjectId
🕒 id	amazonEdited	String
🕒 secret	123456789	String
🕒 __v	0	Int32

Slika 5.2 *thirdParties* kolekcija

Kada administrator izmeni podatke određenom klijentu, u dokumentu sa identifikatorom tog klijenta (*\_id*) se menja polje koje je administrator izmenio. U primeru sa slike 5.2, administrator je klijentu promenio identifikaciju – sa *amazon* na *amazonEdited*. Tajni kod, kao i identifikator dokumenta ostaju nepromenjeni.

Nakon brisanja klijenta iz sistema, njegovi podaci se brišu iz kolekcije *thirdParties*, međutim brišu se i pristupni kodovi i kodovi za osveženje pristupnog koda za sve korisnike koji su eventualno povezani sa konkretnim klijentom. Prikaz kolekcije nakon brisanja klijenta *amazonEdited* dat je na slici 5.3.

Key	Value	Type
▶ (1) ObjectId("591d78a4acf657a825c2732d")	{ 4 fields }	Document
▶ (2) ObjectId("5926ad01583abcf1159c68f9")	{ 4 fields }	Document
▶ (3) ObjectId("594a5b31c2287fb3a5b1466")	{ 4 fields }	Document
▶ (4) ObjectId("594b9b96b66e95687d28a1cb")	{ 4 fields }	Document

Slika 5.3 *thirdParties* kolekcija

## 5.2 Povezivanje naloga

Testiranje autentikacije obavljeno je ručno. Klijenti korišćeni za autentikaciju su Amazon i Google, međutim u radu su prikazane slike povezivanja OBLO naloga sa korisnikovim nalogom na Amazonovom servisu. Test se smatra uspešnim ukoliko je autentikacija obavljena uspešno, odnosno, ukoliko pretraživač preusmeri korisnika na Amazonovu stranicu za indikaciju uspešnog povezivanja naloga.

Na početku je potrebno podesiti klijentove podatke. Amazon nudi opciju podešavanja potrebnih podataka u okviru podešavanja Alexa aplikacije [16]. Alexa aplikacija predstavlja

API za kreiranje novih mogućnosti za glasovnu kontrolu (na primer puštanje muzike, čitanje vesti...). Ukoliko je za rad Alexa aplikacije neophodno pristupati korisničkim podacima sa nekog zaštićenog servisa, Amazon dopušta povezivanje Amazon korisničkog naloga sa nalogom na tom servisu upotrebom koda ovlašćenja ili implicitnog toka. U skladu sa tim, implementirani OAuth 2.0 modul testiran je povezivanjem naloga na oba načina.

Za potrebe testiranja, kreiran je globalni klijent sa identifikatorom *amazon*, kao što je objašnjeno u Poglavlju 5.1. Kako bi dalje povezivanje naloga bilo moguće, Alexa aplikacija podešena je tako da mu je identifikator klijenta *amazon*, a tajni ključ je identičan ključu globalnog klijenta *amazon*. Takođe, podešena je putanja za preusmerenje (Redirect URLs), kao i putanje do autorizacionog servera za prikazivanje početne strane (Authorization URL) kao i za potraživanje pristupnog koda (Access Token URI) – Slika 5.4. Ukoliko se koristi implicitni tok provere ovlašćenja nije potrebno obezbediti putanju za potraživanje pristupnog koda, jer nema preusmerenja. Pogledati Poglavlja 2.2.1.4, 2.2.3.1 i 2.2.3.2.

**Account Linking**

Do you allow users to create an account or link to an existing account with you?  Yes  No  
[Learn more](#)

**Authorization URL**  
The url where customers will be redirected in the companion app to enter login credentials.

**Client Id**  
Unique public string used to identify the client requesting for authentication.

**Domain List (Optional)**  
The list of domains that the authorization URL will fetch content from. You can provide up to 30 domains.  
[Add domain](#)

**Scope (Optional)**  
List of permissions to request from the skill user. You can provide up to 15 scopes.  
[Add scope](#)

**Redirect URLs (Optional)**  
The list of valid HTTPS redirection endpoints that could be requested during authorization to redirect the user back to after the authorization process.  
[Learn more](#)

**Authorization Grant Type (Optional)**  
Specifies the OAuth authorization grant that Alexa uses to obtain an access token from your provider.  
[Learn more](#)  
 Implicit Grant  Auth Code Grant

**Access Token URI**  
This URI will be used for both access token and token refresh requests.

**Client Secret**  
\*\*\*\*\*

Slika 5.4 Podešavanje Amazon Alexa aplikacije za povezivanje naloga uz proveru ovlašćenja korišćenjem koda ovlašćenja

## Account Linking

Do you allow users to create an account or link to an existing account with you?  Yes  No  
[Learn more](#)

**Authorization URL**  
The url where customers will be redirected in the companion app to enter login credentials.

**Redirect URLs (Optional)**  
The list of valid HTTPS redirection endpoints that could be requested during authorization to redirect the user back to after the authorization process. <a class="tocLinks" target="\_blank" href="https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/linking-an-alexa-user-with-a-user-in-your-system#redirect-url-values">Learn more</a>.

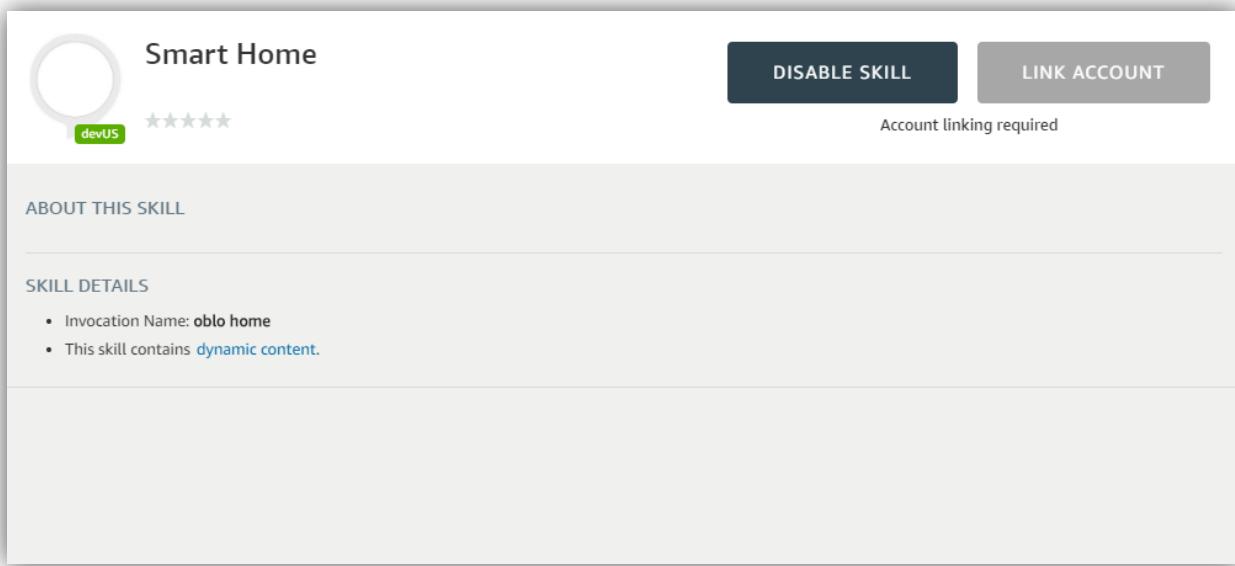
**Client Id**  
Unique public string used to identify the client requesting for authentication.

**Domain List (Optional)**  
The list of domains that the authorization URL will fetch content from. You can provide up to 30 domains.  
[Add domain](#)

**Scope (Optional)**  
List of permissions to request from the skill user. You can provide up to 15 scopes.  
[Add scope](#)

**Authorization Grant Type (Optional)**  
Specifies the OAuth authorization grant that Alexa uses to obtain an access token from your provider.  
 Implicit Grant  Auth Code Grant  
[Learn more](#)

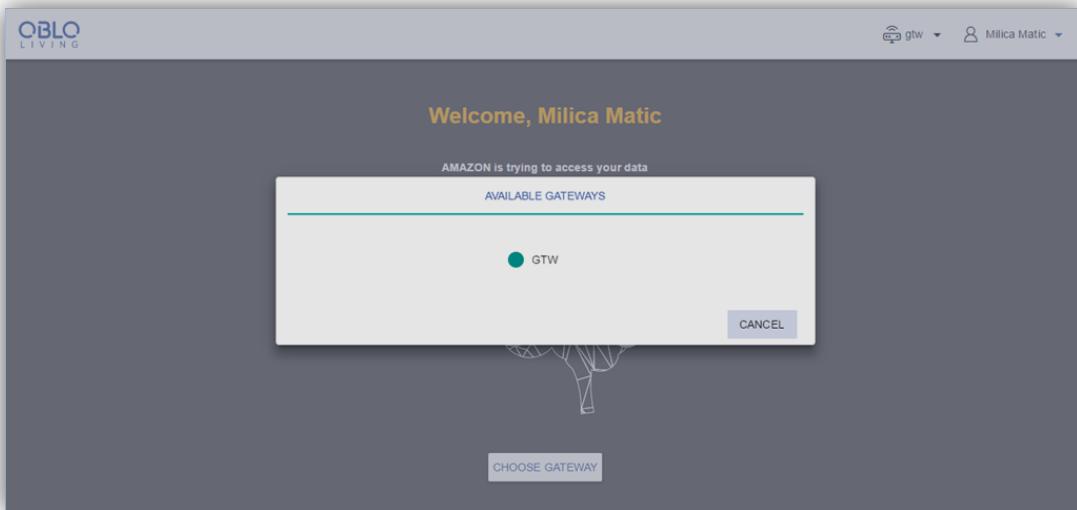
Slika 5.5 Podešavanje Amazon Alexa aplikacije za povezivanje naloga uz proveru ovlašćenja korišćenjem implicitnog toka



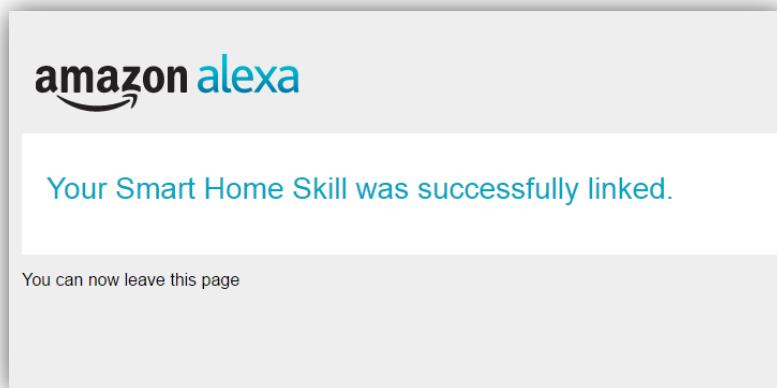
Slika 5.6 Amazon web aplikacija

Na slici 5.6 prikazana je Amazon web aplikacija za povezivanje Amazon naloga sa OBLO nalogom. Klikom na dugme *LINK ACCOUNT* započinje proces autentikacije. Nakon što se korisnik prijavi na OBLO sistem i odabere centralni uređaj sa kojim se povezuje,

pretraživač ga preusmerava na Amazon web stranicu uz indikaciju uspešnog povezivanja naloga.



Slika 5.7 Stranica za odabir centralnog uređaja



Slika 5.8 Indikacija uspešnog povezivanja naloga

Kao dokaz da je i sa OBLO strane sve regularno u bazi podataka su upisani svi podaci neophodni za dalju komunikaciju dva sistema. U kolekciju *profiles* u dokument *clientName* dodaje se novo polje koje predstavlja identifikator klijenta sa kojim je korisnik povezan.

Key	Value	Type
▷ (1) ObjectId("5900ba59bbfd7530431ae74a")	{ 11 fields }	Document
◀ (2) ObjectId("59196555e10297bb20f35d81")	{ 13 fields }	Document
🔗 _id	ObjectId("59196555e10297bb20f35d81")	ObjectId
🕒 firstName	Milica	String
🕒 lastName	Matic	String
🕒 gender	female	String
🕒 username	milica.matic@rt-rk.com	String
🕒 created	5/12/2017, 3:33:33 PM	Date
🕒 preferredLanguage	en_EN	String
▷ (3) groups	Array[1]	Array
▷ (3) contact	Array[1]	Array
▷ (3) address	Array[0]	Array
🕒 __v	25	Int32
🕒 birthDate	null	Null
◀ (3) clientName	Array[4]	Array
🕒 0	google-oblo	String
🕒 1	amazon	String
🕒 2	amazon-oblo	String
🕒 3	amazon-reliance	String
▷ (3) ObjectId("59196ad7e10297bb20f35d87")	{ 13 fields }	Document

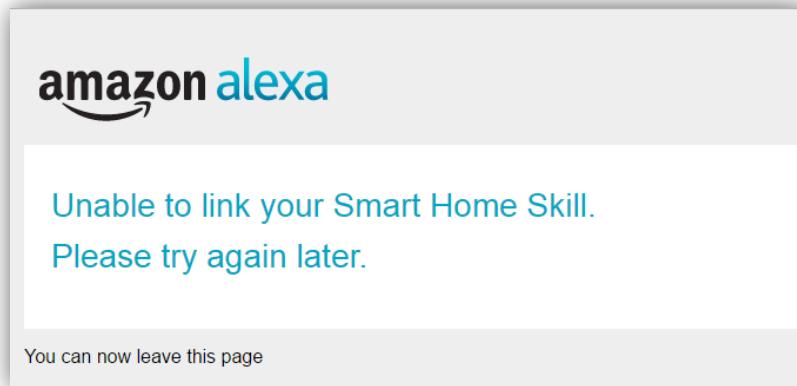
Slika 5.9 *profiles* kolekcija

Kako bi klijent mogao da pristupa resursima korisnika nakon uspešnog povezivanja, u sistemu mora biti zapamćen pristupni kod koji je korišćen za povezivanje naloga. Pristupni kod se čuva u kolekciji *auths* u kojoj se čuvaju poverljivi podaci korisnika (Poglavlje 4.3). Kolekcija *auths* u kojoj je, nakon uspešnog povezivanja naloga, dodat dokument *accessToken* prikazana je na slici 5.10.

Key	Value	Type
▷ (1) ObjectId("5900ba59bbfd7530431ae74c")	{ 7 fields }	Document
◀ (2) ObjectId("59196555e10297bb20f35d83")	{ 8 fields }	Document
🔗 _id	ObjectId("59196555e10297bb20f35d83")	ObjectId
🕒 id	59196555e10297bb20f35d81	String
🕒 identity	milica.matic@rt-rk.com	String
🕒 secret	\$2a\$08\$O5NNwEFFr23Se0cVH8njdeS2KSdzuOQG2EvufZyHKncqfsrqFS4m2	String
🕒 kind	username	String
▷ (3) accessToken	Array[3]	Array
▷ (3) 0	{ 4 fields }	Object
▷ (3) 1	{ 4 fields }	Object
◀ (3) 2	{ 4 fields }	Object
🔗 _id	ObjectId("59524ef40a9b665841f48353")	ObjectId
🕒 accessToken	7cd7d75d452afeba354059de568bf7eee08163158fda8d955fb89ea01d0ae26c5967af6ad99302abb8be	String
🕒 owner	amazon	String
🕒 gtwSerial	784561B7F969	String
🕒 __v	22	Int32
🕒 activated	5/15/2017, 10:23:32 AM	Date
▷ (3) ObjectId("59196ad7e10297bb20f35d87")	{ 8 fields }	Document

Slika 5.10 *auths* kolekcija

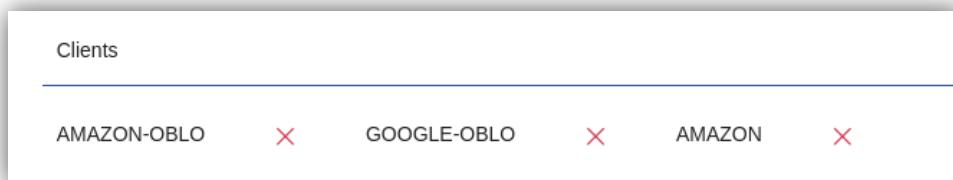
Ukoliko je došlo do neke greške tokom povezivanja naloga, pretraživač preusmerava korisnika na Amazon web stranicu uz indikaciju neuspešnog povezivanja naloga.



Slika 5.11 Indikacija neuspešnog povezivanja naloga

### 5.3 Testiranje grafičke korisničke sprege

U ovom poglavlju testirana je grafička korisnička sprega. Korisnik treba da se prijavi na OBLO web aplikaciju i pristupi stranici sa podešavanjima naloga. Na stranici sa podešavanjima su prikazani svi klijenti sa kojim je njegov nalog povezan.



Slika 5.12 Podešavanja korisničkog naloga

Pritiskom na dugme za brisanje klijenta isti nestaje sa liste, kao i iz baze podataka. Tačnije, briše se identifikator konkrentog klijenta iz kolekcije *profiles* u dokumentu *clientName*, kao i svi pristupni kodovi u kolekciji *auths* koji odgovaraju konkretnom korisniku i klijentu.

## 6. Zaključak

U ovom radu je opisana implementacija OAuth 2.0 modula u okviru OBLO sistema pametnih kuća u cilju povezivanja naloga klijentske aplikacije za obradu glasovne komande sa korisnikovim OBLO nalogom. Implementirana su dva načina povezivanja naloga – sa i bez korišćenja koda ovlašćenja. U sistem je dodata grafička korisnička sprega koja olakšava korisniku manipulaciju klijentima sa kojima je njegov nalog povezan.

Prednost ovog rešenja ogleda se u bezbednosti koju pruža. Strana klijentska aplikacija ne može da zahteva povezivanje sa korisnikom OBLO sistema ukoliko nije odobrena od strane administratora sistema. Međutim i sa odobrenjem za povezivanje naloga klijentska aplikacija ne može da pristupi ni jednom delu OBLO sistema za koji nema ovlašćen pristup. Iako je OAuth 2.0 modul iskorišćen za povezivanje sistema za raspoznavanje glasovnih komandi i OBLO sistema, ova komponenta može da se koristi za bilo koju aplikaciju koja bi potencijalno zahtevala pristup resursima korisnika OBLO sistema.

Dalji pravci poboljšanja sastoje se od proširenja sistema za povezivanje više centralnih uređaja sa jednim klijentom. Takođe poželjno bi bilo povezivanje novih sistema za glasovnu komandu sa OBLO sistemom kako bi se ispitalo održavanje OAuth 2.0 modula u nepoznatom sistemu.

## 7. Literatura

- [1] OBLO official site: <http://www.obloliving.com/pocetna/>
- [2] ZIGBEE, Alliance. Zigbee specification. ZigBee document 053474r13, 2006.
- [3] Christian Paetz, “Z-Wave Basics: Remote Control in Smart Homes“, 2013
- [4] MQTT protokol, [Online] Dostupno na: <http://mqtt.org>
- [5] Amazon Echo uređaj za detekciju i raspoznavanje govora, [Online] Dostupno na: [https://en.wikipedia.org/wiki/Amazon\\_Echo](https://en.wikipedia.org/wiki/Amazon_Echo)
- [6] Google Home uređaj za detekciju i raspoznavanje govora, [Online] Dostupno na: <https://madeby.google.com/home/>
- [7] Milan Tucić: Protokol za razmenu poruka u sistemim pametnih kuća, Univerzet u Novom Sadu, Fakultet Tehničkih Nauka, 2016
- [8] [RFC6749] D.Hardt, “The OAuth 2.0 Authorization Framework”, oktobar 2012
- [9] M. Jones, D. Hardt, D. Recordon, “The OAuth 2.0 Protocol: Bearer Tokens”, maj 2011
- [10] [RFC2616], R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, “Hypertext Transfer Protocol - HTTP/1.1”, jun 1999
- [11] Alexa Voice Service, [Online] Dostupno na: <https://developer.amazon.com/alexavoice-service>
- [12] Google Assistant, [Online] Dostupno na: <https://assistant.google.com/>
- [13] Node.js paket *auth2orize*, [Online] Dostupno na: <https://github.com/jaredhanson/oauth2orize>
- [14] MongoDb – noSQL baza podataka, [Online] Dostupno na: <https://www.mongodb.com/>
- [15] Mongoose, [Online] Dostupno na: <http://mongoosejs.com/>

[16] Alexa Skills Kit: [Online] Dostupno na:

<https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/getting-started-guide>