Универзитет у Београду
Математички факултет

Групни студентски пројекат

Менаџер управљања односа са клијентима студентске организације AIESEC

Студенти:	Обренић Стана, 1120/2012 Ракоњац Ђорђе, 1135/2013 Миљковић Андрија, 1025/2012 Атанасовски Петар, 1032/2012 Вељковић Немања, 1007/2012 Петковић Стефан, 1035/2012 Терзић Срђан, 1054/2012
Предмет:	Р338 Програмирање за Веб
Наставник:	др Саша Малков
Школска година:	2013-2014
Датум:	29.01.2014.

Садржај

1 Речник појмова

- *Корисник* – припадник организације, врши одређене процесе унутар система. Постоји три типа корисника који у зависности од своје позиције у систему имају одређене привилегије за вршење процеса.
- *Члан* – припадник организације, врши одређене процесе унутар система. Постоји три типа чланова који у зависности од своје позиције у систему имају одређене привилегије за вршење процеса.
- *Локални администратор* – корисник са средњим нивоом привилегија. Може да чита све податке, уноси институције и мења податке о члановима за свој ентитет.
- *Глобални администратор* – корисник са највишим привилегијама. Може да чита, мења и уноси све податке доступне кроз систем.
- *Институција* – компаније, фирма или јавна установа са којом организација обавља сарадњу.
- *Апликација* – представља однос између члана и институције.
- *Статус апликације* – ниво преговора до ког је стигла одређена апликација.
- *Извештај* – детаљан опис апликације и статуса у ком се та апликација налази.
- *Ентитет* – канцеларија којој члан може припадати.

2 Увод

Рад представља претварање идејног пројекта у функционалну Веб апликацију. Апликација треба да омогући унос, читање, брисање, мењање и адекватно приказивање свих информација које се налазе у бази података, а које представљају процес сарадње чланова организације са једне стране и одређених институција са друге. Апликација треба да помогне у сагледавању целокупне сарадње између чланова организације и њених партнера, као и да памти све завршене сарадње и статистички их обрађује.

3 Идеја

У систему постоји више типова корисника, који имају одређене привилегије за обављање текућих послова. Сви корисници успостављају сарадњу са институцијама, та сарадња пролази кроз неколико фаза и за сваку од фаза може, не нужно, бити везан неки извештај. Систем је потпуно транспарентан за све чланове организације и свако може имати увид у све расположиве информације. Глобални администратор има право да чита, мења, брише и модификује све информације у бази података, локални администратор има право мењања појединих података везаних за чланове из свог ентитета и уношења нових институција, док члан има право да уноси нове институције.

3.1 Захтеви

Потребно је направити апликацију која би чувала податке о потенцијалним партнерима организације, односима и контактима које би требало успоставити или одржавати, особама које су задужене за сарадњу и које раде на контакту. У склопу система налази се и анализа успешности сваког корака у продајном сектору. Постоји неколико циклуса кроз које пролази партнерска институција (послата на одобрење, одобрена, контактирана, заказан састанак, одржан састанак, потписан уговор). Пошто организација има више канцеларија, јако је битно да се зна која канцеларија контактира коју институцију, тако да је из тог разлога битно имати систем одобравања, и јасне извештаје сваког контактирања. Партнерска институција би требало по правилу да буде јединствено одређена матичним бројем из АПР-а. Чланови имају право уношења нових институција у базу, али свака од тих институција мора бити одобрена од стране неког од глобалних администратора. Од тренутка када је одобрена, члан који је пријавио има 2 месеца да прође кроз све процесе и потпише уговор. У супротном, она прелази поново на статус "Слободна", када је у сарадњу може узети други члан уз задржавање свих извештаја контактирања. Анализа успешности продајног сектора огледа се кроз следеће односе:

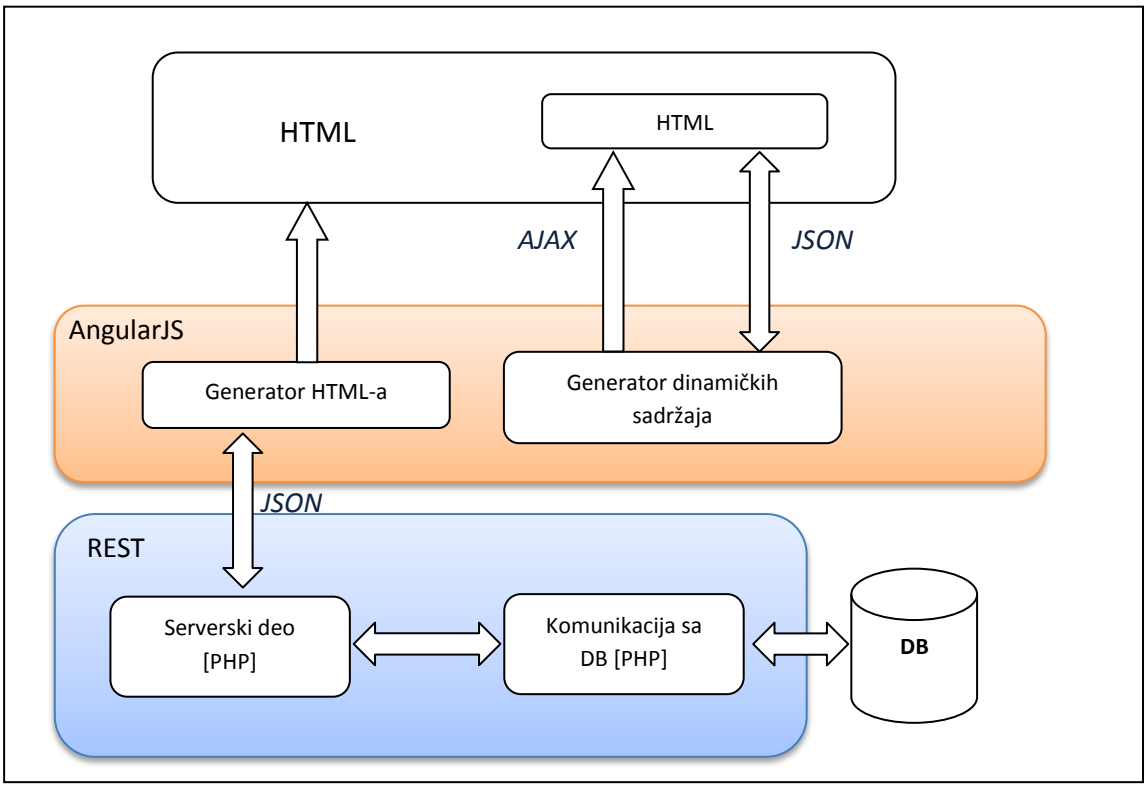
- Број одобрених институција / Број пријављених институција
- Број контактираних институција / Број одобрених институција
- Број заказаних састанака / Број контактираних институција
- Број уговора/Број састанака

3.2 Циљеви

Испуњење свих дефинисаних захтева и долазак до функционалне Веб апликације која ће моћи да ради у реалним условима.

4 Архитектура система

Одабрана је архитектура тешких клијената са коришћењем **REST** архитектуре за испоручивање података. Коришћење **REST** архитектуре се може показати као добро, нарочито ако се у будућности буде правила андроид (или нека друга) апликација, као што је планирано. Архитектура тешких клијената подразумева испоручивање необрађених података клијенту од стране сервера, где се клијент брине о њиховој обради и адекватном приказивању. Овакав приступ смањује оптерећење сервера. Са све јачим клијентским рачунарима ова архитектура постаје јако популарна.



Слика 4.1: Архитектура апликације

Архитектура апликације је вишеслојна, а састоји се из:

1) ***Базе података*** која је детаљно објашњена у делу 4.1

2) ***REST слоја*** - састоји се из серверског дела и дела комуникације са базом података. Серверски део прима захтеве и врши њихову обраду, а затим захтеве за подацима шаље делу који комуницира са базом података који потом податке враћа серверском делу који исте припрема у JSON формату и као такве прослеђује Angular слоју. Захтеви у серверском делу се успостављају са методама (GET, POST, PUT, DELETE) тј. методама ПЧМБ-а потом захтевом (нпр. GET institutions/school - који ће вратити све школске институције), што и дефинише REST.

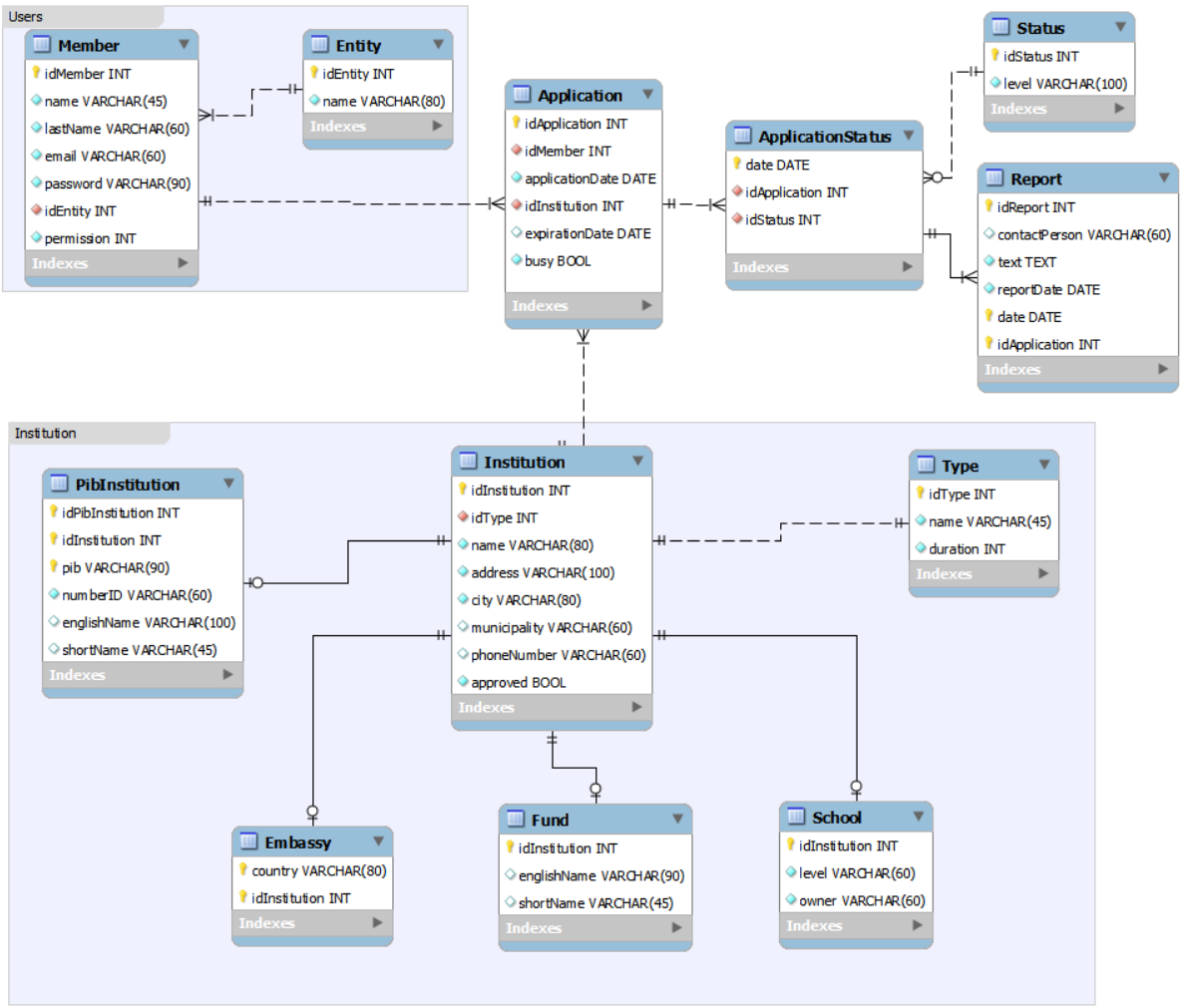
3) ***Angular слоја*** - овај слој омогућава комуникацију презентационог слоја и REST слоја, врши припрему и приказивање затраженог и добијеног садржаја. Комуникацију са REST слојем врши JSON-ом и AJAX-ом што исто користи и за комуникацију са презентационим слојем.

4) ***Презентационог слоја*** - врши приказ података и остварује директну комуникацију са корисником.

4.1 База података

ERR (Entity-relationship model)

Апстрактан начин за описивање база података. Описује податке које обухватају објекте и везе између њих у реалном свету, користи се за развој иницијалне базе података. На слици 4.2 приказан је EPP дијаграм који описује базу података, ентитете и њихове међусобне односе који су коришћени у пројекту.



Слика 4.2: EPP дијаграм базе података

Корисници

Ентитет (*Entity*) - представља канцеларију за коју је члан везан.

Члан (*Member*) - садржи податке о члану, личне податке, пермисије и информацију којој канцеларији члан припада.

Институције

Тип (*Type*) **-** садржи податке о типовима институција њиховим називима и дужином трајања сарадње за тај одређени тип институције. (Типови институције су: компаније, хотели, хостели, амбасаде, школе, фондови, предшколске установе, невладине организације и канцеларије за младе).

Институција (*Institution*) - садржи све релевантне податке о институцији, типу институције и да ли је она одобрена или не (само уколико је институција одобрена, члан може успоставити сарадњу са њом).

Пиб институција (*PibInstitution*) **-** садржи конкретне податке о институцијама које имају свој Пиб и матични број. Институције овог типа су: компаније, хотели, хостели, предшколске установе и невладине организације.

Амбасада (*Embassy*) **-** садржи додатни податак коју земљу представља амбасада, а пошто само једна амбасада може постојати за једну земљу, поље *country* је проглашено за примарни кључ.

Фонд (Fund) - садржи додатне податке о фонду њиховом кратком и дугом називу.

Школа (*School*) - садржи податке о школи, ког је нивоа (основна, средња, виша, факултет...), које је власничке структуре (државна или приватна).

Апликације

Апликација (*Application*) - представља везу (сарадњу) између чланова и институција. Садржи информације о датуму пријаве и датуму истека апликације, као и поље *busy* које одређује да ли је нека институција заузета или не, а уједно служи и за архивирање свих сарадњи чланова и институција.

Статус апликације (*ApplicationStatus*) - одређује на ком нивоу (статусу) се налази тренутна апликација односно сарадња, као и извештаји које чланови пишу како би се имало у виду докле се стигло и како тече сарадња.

Статус (*Status*) - представља ниво сарадње. Он се одвија у неколико нивоа, а то су: слободан, аплициран, одобрен, контактиран, дефинисан датум састанка, одржан састанак и потписан уговор.

Извештај (*Report*) - додатне информације о сарадњи, информације о особама за контакт и адекватан текст.

5 Изворни код апликације

Целокупан код апликације организован је у две велике целине, серверски и клијентски део. Пошто су коришћене архитектура тешких клијената и **REST** сервис, оваква подела се наметнула као природна.

5.1 Серверски део

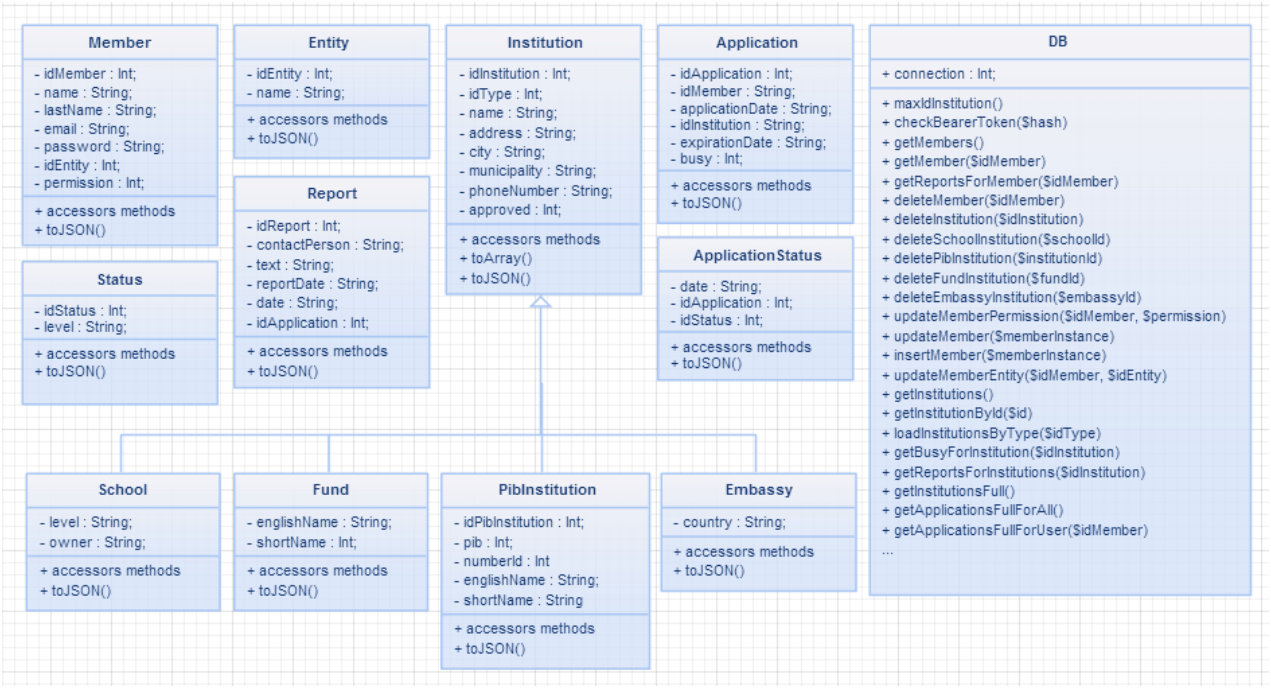
Изворни код серверског дела организован је у неколико **PHP** датотека, међу којима су:

server.php
db.php
func.php

као и појединачне датотеке за сваку од класа које су коришћене:

Application.php
ApplicationStatus.php
Report.php
Status.php
Institution.php
Embassy.php
Fund.php
PibInstitution.php
School.php

Организација ових класа се може видети на дијаграму класа на слици 5.1.



Слика 5.1: Дијаграм класа

Свака од ових класа поседује поља која одговарају атрибутима одговарајућих табела базе података. Такође, свака класа поседује и *toJSON()* метод који враћа JSON репрезентацију конкретне инстанце класе. Ради лакшег ковертовања низова објеката ових класа у **JSON** формат, сва поља су проглашена као јавна и на тај начин постала доступна методу *json_encode()*. Због недостатка времена били смо приморани на овакав потез који нарушава почетну замисао о енкапсулацији.

Датотека *server.php* је у начелу прималац **HTTP** захтева. Она прихвата **URI** и тип захтева, затим парсира адресу и у зависности од типа захтева (**GET**, **POST**, **PUT** и **DELETE**) и елемената адресе позива одговарајући метод објекта *$db*. Ови методи враћају **JSON** репрезентацију траженог ресурса или обавештења. Тако добијени подаци се затим шаљу клијенту као тело одговора на послати захтев, уз одговарајући статусни код.

$db је инстанца класе *DB* која се налази у засебној датотеци *db.php.* Ова класа је задужена за комуникацију са базом података и извршавање конкретних упита које захтева клијент. Њен код је организован по методама које се позивају у *server.php* датотеци и које извршавају различите упите.

Датотеке *server.php* и *db.php* представљају основу **REST** сервиса који представља срж целокупне апликације.

Датотека *func.php* садржи функцију isAutorized() која проверава да ли је корисник ауторизован да приступи апликацији. Она позива функцију *checkBearerToken()* из објекта *$db* која проверава корисничко име и шифру у бази и у зависности од успешности операције враћа ауторизациони токен или нулу.

5.2 Клијентски део

Клијентски део апликације подељен је у више директоријума који групишу одговарајуће датотеке:

css - датотека са свим .css фајловима

img - слике које се користе у апликацији

js - датотека са нашим **JavaScript** кодом, најбитнији фајлови су:

services.js - фабрике за повезивање са **REST** сервисом

controllers.js - контролери страница, учитавају и обрађују садржај

app.js - датотека у коме је одређено рутирање, повезани контролери са страницама

lib - спољашње библиотеке које смо користили за рад

templates - .html фајлови, који се учитавају у Мастер страницу

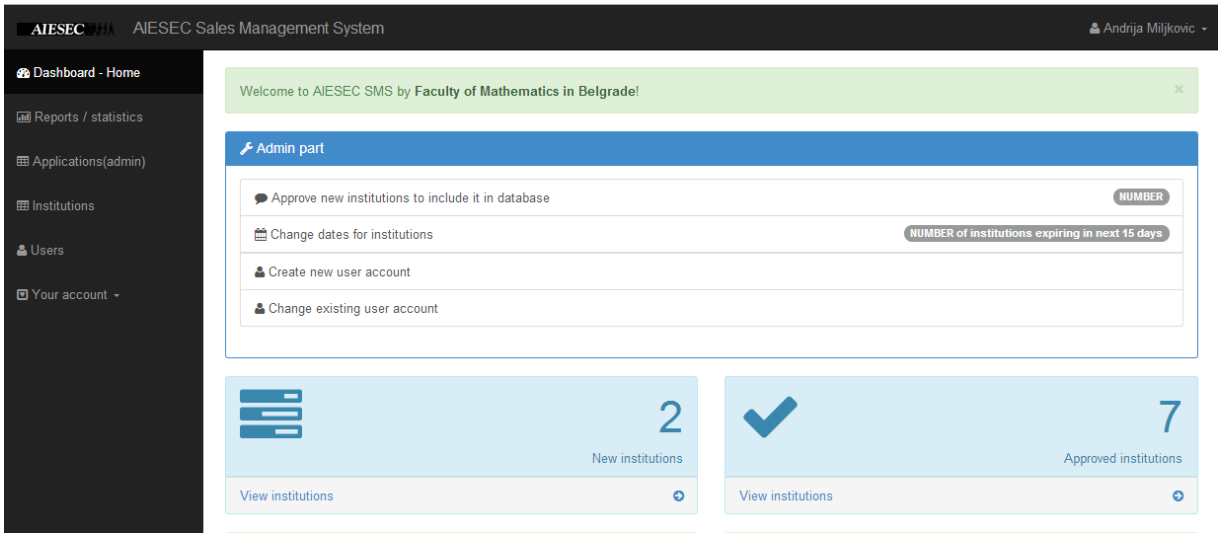
index.html - Мастер страница у коју се Ајах-ом учитавају потребни подаци

login.html - страница за логовање која је издвојена из система

Сам *AngularJS* је организован по МПК принципу, кога смо се и ми придржавали. Поглед је рађен помоћу *BootStrap* библиотеке, која омогућава удобан рад и што је још битније приказ који се разликује у зависности од уређаја на коме се приказује, било да је то рачунар, таблет или мобилни телефон. Помоћу *AngularJS* и *BootStrap* библиотека, на врло ефикасан начин се врши приказ, филтрирање и сортирање потребних података.

Модел део овде представљају фабрике које учитавају податке из **REST** сервиса и омогућавају све ПЧМБ операције. За учитавање података коришћен је *$resource* модул за рад са **RESTFull** сервисима. Ту су дефинисане методе *query*, *get*, *create*, *update*, *delete* које се касније само позову из одговарајућег контролера.

Модел и поглед се повезују помоћу контролера. Он дефинише функције које се на одговарајући клик или у зависности од **URI** адресе приказују на екрану. Контролер је са фабрикама повезан помоћу *Dependency Injection* шаблона, који претпоставља где је потребан одређени ресурс из тог сервиса, а довољно је само навести назив сервиса при позиву, и он ће бити доступан контролеру. Иста ствар важи и за позив функција контролера из погледа.



Слика 5.2: Изглед апликације

6 Сигурност система

Много пажње је посвећено на пољу сигурности апликације. Генерално имплементиран је oauth протокол који иако није званичан стандард претендује да то буде што се тиче рест сервиса. У наставку је детаљније објашњен принцип аутентификације и ауторизације корисника.

6.1 Аутентификација

Како је по принципима **REST** сервиса сваки захтев комплетан, било је потребно осмислити имплементацију *oauth* протокола која одговара нашим потребама. Наиме, да би нека апликација (претраживач) односно у нашем слчају корисник добио могућност да приступа **REST** апију потребно је да сервер испреговара токен који ће се користити у свим наредним позивима **REST** апи-ја. Захтевање токена од стране корисника се врши на следећи начин:

Корисник (апликација која користи **REST** сервис) шаље: **POST** */oauth/token HTTP*/1.1 захтев при чему је обавезано ауторизационо заглавље у облику:

Authorization: Basic eHZ6MWV2RlM0d0VFUFRHRUZQSEJvZzpMOHFxOVBaeVJn
NmllS0dFS2hab2xHQzB2SldMdzhpRUo4OERSZHlPZw==

Ниска која се прослеђује представља *base64* енкодирану ниску која је састоји од комбинације корисниковог емејла и лозинке. Након прослеђеног захтева за токеном, сервер декодира ниску, хешује је при чему се иста салтује и онда се у бази проверава да ли постоји 1-1 кореспонденција са неким постојећим хешом. Уколико постоји такав хеш у бази, сервер враћа **JSON** одговор у коме постоји токен који ће корисник тј апликација користити у наредним позивима **REST** апија. Одговор је облика:

{"token_type":"bearer","access_token":"28c17e1db7bf041caab1caf53626c5df"}

Уколико је апликација успешно испреговарала токен који ће се користити у наредним позивима, *access_token* се снима код клијента као колачић, а заглавље сваког наредног позива према **REST** апију се врши прослеђивањем такође ауторизационог заглавља облика:

Authorization: **Bearer** YW5kcmlqYS5taWxqa292aWNAYWllc2VjLm5ldDoxMjM=

При чему ниска представља енкодиран облик хеша приступног токена. На овај начин смо сигурни да је корисник баш онај за кога се представља. Бар у оној мери у коме је тешко направити две различите ниске које пропуштене кроз мд5 алгоритам враћају исти резултат. У случају компромитоване базе, нападач неће моћи да користи *rainbow* табеле па се пробијање хешева своди на напад речником или *brute-force* што онда директно зависи од јачине саме лозинке. У наредној секцији ће кратко бити представљен проблем ауторизације.

6.2 Ауторизација

Сваки корисник у бази података садржи податак о свом рангу тј. о дозволама. Провера да ли је корисник ауторизован да изврши неки захтев се врши на серверу и то тако што се за одређени токен уколико је упарен проверава да ли власник токена има овлашћења да изврши одређени захтев, уколико нема враћа се 401 грешка тј. грешка о неауторизованом приступу. Такође битно је напоменути да се на клијентској страни тј. страни апликације користе *angular interceptori* који на овакве захтеве одговарају тако што корисника враћају на страницу за логовање.

7 Литература

За израду пројекта коришћени су следећи извори:

- Eric J. Naiburg, Robert A. Maksimchuk, *UML za projektovanje baza podataka,* CET Computer Equipment and Trade, 2002.
- Leonard Richardson and Sam Ruby, RestFULL Web Services, O'Reilly
- Lorna Mitchell, Davey Shafik, Matthew Turland, PHP master write cutting-edge code
- http://angularjs.org
- http://stackoverflow.com
- http://getbootstrap.com
- http://www.php.net