TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut
Võrgutarkvara õppetool

Õppejõudude hindamise rakenduse REST API ja kasutajaliides kasutades Spring ja AngularJS raamistikke

Bakalaureusetöö

Üliõpilane: Marko Mets
Üliõpilaskood: 120925IAPB
Juhendaja: Jaagup Irve

Tallinn
2015

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

(*kuupäev*) (*allkiri*)

Annotatsioon

Käesoleva töö põhieesmärke oli kaks. Esiteks, teha valmis veebirakendus, mis võimaldaks tudengitel hinnata ning arvustada õppejõude ning lugeda teiste poolt jäetud arvustusi ja hinnanguid. Veebirakendus peab olema mugavalt kasutatav nii arvutite kui ka nutiseadmetega. Teiseks eesmärgiks on õppida kasutama Spring, AngularJS ja Bootstrap raamistikke.

Töö probleemideks oli läbi mõelda kasutajaliidese ja serveripoolse rakenduse struktuurid ning kuidas need kaks osa omavahel suhtlema panna. Lisaks sellele pidi mõtlema, milliseid raamistikke ja tehnoloogiaid oleks kõige otstarbekam vaadeldava rakenduse tegemise juures kasutada.

Töö tulemusena valmis plaanitud veebirakendus. Loodud rakendus on mugavalt kasutatav nii nutiseadmetega kui ka suure ekraaniga. On tehtud kasutades tänapäevaseid ja hinnatud raamistikke. Rakendus võimaldab jätta tagasisidet õppejõudude kohta ning teiste tagasisidet ka lugeda.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 5 peatükki, 5 joonist, 2 tabelit.

Abstract

This work had two main aims. Firstly to create an application for the purpose of rating and reviewing teachers by students and also reading ratings and reviews left by others. Application has to be modern and easy to use on both smartphones/tablets and computers. Secondly to learn more about different software frameworks (Spring and AngularJS) and how to implement them.

The main problems were to design structure of the front-end and back-end parts of the application, how to combine those two, which software frameworks to choose for development to make the application as attractive as possible.

As a result of the work an application to rate and review teachers was created. Application is easy to use on devices with different screen sizes. Application was developed with help of different modern and highly valued frameworks. Author gained much needed expirence with different technologies.

The thesis is in Estonian and contains 33 pages of text, 5 chapters, 5 figures, 2 tables.

Lühendite ja mõistete sõnastik

API	***Application Programming Interface*** reeglistik olemasoleva valmisprogrammiga suhtlemiseks
JAR	***Java Archive*** Java Arhiivi fail
WAR	***Web application Archive*** Veebirakenduse arhiivi fail

Jooniste nimekiri

Joonis 1: Õppejõudude loend mobiilivaates 23
Joonis 2: Õppejõudude loend tavavaates 24
Joonis 3: Superkasutaja paneel tavavaates 24
Joonis 4: Superkasutaja paneel mobiilivaates 25
Joonis 5: Andmebaasi diagramm 28

Tabelite nimekiri

Tabel 1: Rakenduse vaated 19
Tabel 2: REST otspunktid 27

Sisukord

1. Sissejuhatus 10
2. Raamistikud ja tehnoloogiad 11
2.1 REST 11
2.2 Spring 11
2.2.1 Spring Web MVC 11
2.2.2 Spring Security 12
2.2.3 Dependency injection 13
2.3 JSON 13
2.4 Jackson 13
2.5 AngularJS 14
2.6 Bootstrap 15
2.7 BCrypt 15
2.8 PostgreSQL 16
2.9 Maven 17
3. Rakenduse struktuur 18
3.1 Rakendus 18
3.1.1 Rakenduse võimalused 18
3.1.2 Rollid 19
3.2 Kasutajaliides 19
3.2.1 Vaated 19
3.2.2 Andmete saamine 20
3.2.3 Autentimine 20
3.2.4 Tõlkimine 21
3.2.5 Local storage 21
3.2.6 Responsive disain 22
3.3 Serverirakendus 25
3.3.1 Autentimine 25
3.3.2 REST otspunktid 26
3.3.3 Rakenduse kihid 27
3.4 Andmebaas 28

4. Hinnang rakendusele 29
4.1 Tagasiside 29
5. Kokkuvõte 31
Summary 32
Kasutatud kirjandus 33

1. Sissejuhatus

Tihti soovivad tudengid enne aine deklareerimist teada saada, milline ainet antavatest õppejõududest on kõige parem. Erinevates foorumites võivad olla kirja pandud mõningad arvamused huvi pakkuva õppejõu kohta, kuid ei ole olemas ühte keskset andmebaasi, mis ongi pühendatud ainult õppejõudude hindamisele teiste kasutajate poolt.

Käesoleva töö eesmärgiks on teha valmis funktsioneeriv, moodne veebirakendus, mis võimaldaks tudengitel hinnata ning arvustada õppejõude ning lugeda teiste poolt jäetud arvustusi ja hinnanguid. Veebirakendus peab olema mugavalt kasutatav nii arvutite kui ka nutiseadmetega. Teiseks eesmärgiks on õppida kasutama Spring, AngularJS ning Bootstrap raamistikke.

Töö on jaotatud mitmeks osaks. Esmalt tutvustatakse kasutatud raamistikke ja tehnoloogiaid. Seejärel räägitakse, kuidas rakendus on üles ehitatud ning peale seda tutvustatakse testimise käigus leitud vigasid ning ettepanekuid rakenduse paremaks toimimiseks.

2. Raamistikud ja tehnoloogiad

Sellest peatükis tutvustatakse kõiki töös kasutatud raamistikke ning tehnoloogiaid.

2.1 REST

REST on olekuta (i.k stateless) klient-server kommunikatsiooni protokoll, mis kasutab HTTP-d. REST-i idee seisneb selles, et keerukate protokollide (SOAP) asemel kasutatakse lihtsat HTTP-d [1]. Kõigi CRUD (Create, Read, Update, Delete) operatsioonide puhul kasutatakse REST-i puhul HTTP päringuid (POST, GET, PUT, DELETE). REST protokolli kasutav rakendus võib päringule vastata XML, JSON võis CSV formaadis.

Vaadeldavas töös kasutatakse JSON formaati, sest serverist saadud andmete kuvamine kasutajaliideses toimub JavaScripti abil.

2.2 Spring

Spring on avatud lähtekoodiga raamistik Java platvormi jaoks [2]. Spring raamistik lihtsustab veebirakenduste kirjutamist, võimaldab teostada "sõltuvuste süstimist" (i.k dependency injection), lihtsustab autentimist ja palju muud [3]. Järgnevalt kirjeldatakse, mis funktsionaalsust täpsemalt kasutati.

2.2.1 Spring Web MVC

Spring Web MVC lihtsustab veebirakenduste tegemist. Raamistik võimaldab defineerida kontrollereid, mis teenindavad sisse tulevaid päringuid. Kontrolleri rolli täidab Java klass. Et Spring teaks, et tegemist on kontrolleriga, tuleb klassile kirjutada annotatsioon *@Controller*. Kirjutades kontrollerile annotatsioon *@RequestMapping("/url")*, hakkab kontroller teenindama päringuid, mis tulevad aadressile */url*. Näiteks:

```
@RequestMapping("/url")
@Controller
public class DataController{}
```

Klassile tuleb kirjutada meetodid, mis hakkavad sisse tulevate päringutega tegelema. Meetod peab olema *public*. Ka meetodile tuleb kirjutada annotatsioon, näiteks *@RequestMapping("/data")*, mis tagab, et antud meetod tegeleb GET päringutega URL-le */url/data*. *@RequestMapping("/data", method=RequestMethod.POST)* võimaldab pöörduda sama URL-i poole POST päringuga. *Method* võib olla üke järgnevatest: *RequestMethod.GET, RequestMethod.POST, RequestMethod.DELETE, RequestMethod.PUT, RequestMethod.HEAD, RequestMethod.OPTIONS, RequestMethod.TRACE* [4].

Olgu meil URL */data/123*. Et saada kätte väärtus *123* tuleb kasutada kahte annotatsiooni: meetodile tuleb kirjutada *@RequestMapping("/data/{id}")* ning meetodi parameetriks *@PathVariable String id*. Näiteks:

```
@RequestMapping("/data/{id}")
public void getDataById(@PathVariable String id){...}
```

Spring automaatselt saab kätte soovitud väärtuse ning salvestab selle muutujasse id tüübiga *String*. Peab silmas pidama, et muutuja nimi ning loogeliste sulgude vahel olev väärtus oleksid sama nimega.

Kontrolleri meetodid võivad olla *void* tüüpi, kuid võivad ka tagastada *String* ja *int* tüüpi anmdeid. Lisaks sellele võivad nad tagastada ka jsp ja html lehti.

Näiteks:

mvc-dispatcher-servlet.xml

```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/pages/"/>
    <property name="suffix" value=".jsp"/>
</bean>
```

```
public String getDataById(){return "data"}
```

Antud näide tagastab lehe *data.jsp*, mis asub kaustas /WEB-INF/pages.

2.2.2 Spring Security

Spring Security on Springi moodul, mis keskendub Java rakendustes autentimisele ja autoriseerimisele [5]. Spring Security abil ei pea ise kirjutama palju koodi, mis tegeleb autentimisega. Spring Security on kergesti laiendatav, et täita arendaja erivajadusi. Näiteks eduka autentimise puhul on rakendusel nõue saata päringu algatajale kasutaja unikaalne identifikaator ja tema rollid. Selle saavutamiseks tuleb kirjutada enda klass, mis on Spring Security raamistiku ühe klassi alamklass ning kirjutada üle meetod, mis kutsutakse välja

eduka autentimise puhul. Seejärel tuleb öelda raamistikule, et seda klassi on vaja kasutada eduka autentimise puhul. Kui autentimine on õnnestunud, siis tehtud klassi üle kirjutatud meetod kutsutakse välja. Samamoodi saab käituda ka teiste olukordade puhul: ebaõnnestunud autentimine, välja logimine, jne.

2.2.3 Dependency injection

Dependecy Injection (DI) ehk sõltuvuste süstimine on muster, kus objekt, soovides kasutada teist objekti, ei pea seda ise algväärtustama, vaid talle on juba antud viit soovitavale objektile. DI ei ole omane ainult Springile, kuid Springis on tehtud selle kasutamine väga mugavaks.

DI kasutamiseks tuleb kirjutada objektile, mida soovitakse kasutada, annotatsioon *@Service, @Component*, *@Repository* või *@Controller*. Kui soovitakse saada viit objektile, siis kasutatakse annotatsiooni *@Autowired*. Annotatsioonid *@Service, @Component* ja *@Repository* on *@Component*-i erijuhud, mida kasutatakse vastavalt teenuskihis, esituskihis ja andmebaasiga suhtluse kihis.

Näiteks:

```
@Component
public class User{}

public class UserController{
        @Autowired
        private User user;
}
```

2.3 JSON

JSON (JavaScript Object Notation) on lihtne andmevahetusformaat, mida on nii inimeste kui ka masinate jaoks kerge lugeda ja kirjutada. JSON põhineb kahel struktuuril: nime ja väärtuse paaridel ning jadadel. [6] Vaadeldavas rakenduses toimub kasutajaliidese ja serveri vaheline suhtlus kasutades JSON formaati.

2.4 Jackson

Jackson on avatud lähtekoodiga JSON-i parser ja generaator [7]. Vaadeldavas töös kasutatakse seda Java objektide ja kollektsioonide teisendamiseks JSON formaadi ja JSON formaadis olevate andmete teisendamiseks Java objektiks.

Eelnevalt sai mainitud, et kontrolleri meetodid võivad tagastada erinevat tüüpi andmeid. Selleks, et tagastada Java objektis sisalduvaid andmeid JSON formaadis, saab väga mugavalt kasutada Jacksonit. Selleks tuleb meetodile kirjutada annotatsioon *@ResponseBody*(kasutatakse selles töös) või klassile kirjutada *@Controller* asemel *@RestController* (sel juhul ei pea kirjutama *@ResponseBody*, sest see lisatakse automaatselt).

Näiteks:

```
public class User{
	private String firstName = "Peeter";
	private String lastName = "Mets";
	private int age = 29;
	// getters, setters
}
```

```
public @ResponseBody User getUser(){
	return new User();
}
```

Eelnev näide tagastab objekti andmed JSON kujul

```
{
	"firstName": "Peeter",
	"lastName": "Mets",
	"age": 29
}
```

Jackson on võimeline teisendama ka keerukamaid struktuure, näiteks objektide kollektsioone ning objekte, mis sisaldavad viitasid teistele objektidele.

Jackson suudab teisendada ka JSON-is olevaid andmeid Java objektiks.

Näiteks saates päringuga kaasa eelmainitud JSON-i:

```
public void addUser(@RequestBody User user){
	//muutujas User on salvestatud saadetud JSON-i väärtused.
}
```

Peab silmas pidama, et JSON-is olevate andmete võtmete nimed oleksid samad, mis Java objekti muutujate nimed.

2.5 AngularJS

AngularJS on avatud lähtekoodiga JavaScripti raamistik veebirakenduste loomiseks [8]. Angular võimaldab kasutada kasutajaliideste tegemisel MVC mustrit. Angulariga saab laiendada olemasoelvat HTML-i funktsionaalsust.

Angularis on kontroller JavaScripti funktsioon, mida saab siduda HTML vaatega ning kuvada seal andmeid [9]. Angular sünkroniseerib automaatselt andmed vaate ja kontrolleri vahel, ilma et peaks ise selle jaoks koodi juurde kirjutama.

Angulariga saab vaadetes andmeid kuvada dünaamiliselt. Selle jaoks saab kasutada Angulari omadust nimega *Expressions*, mis kujutab endast koodijuppe, mis on pandud HTML dokumendis loogsulgude vahele [10]. Näiteks {{1+1}} tulemuseks oleks 2 või kui kontrolleris on olemas muutuja *name*, siis {{name}} tulemuseks oleks muutuja väärtus.

Angular võimaldab luua direktiive, mis on enda tehtud HTML-i argumendid ja elemendid [11]. Enim vajalikud direktiivid on raamistikus juba olemas. Näiteks *ng-repeat* võimaldab HTML-is itereerida üle kollektsiooni: `<div ng-repeat="obj in objects">{{obj.name}}</div>`, kus *objects* on objektide massiiv ning igal objektil on väärtus *name*. Eelnev näide loob div elemendi iga objekti kohta massiivis ning trükib tema nime. Samuti võib luua direktiiviga uusi HTML elemente, näiteks `<pagination></pagination>`, mis lubab mugavalt teostada andmete lehekülgedeks jagamist.

Filter on mõeldud etteantud andmete vormistamiseks [12]. Näiteks saab teha filtri `<div ng-repeat={{person in persons | orderBy: '-age'}}>{{person.age}}</div>`, mis sorteerib kollektsiooni *persons* vanuse järgi kahanevas järjekorras ning trükib välja iga objekti vanuse välja väärtuse. Antud töös kasutatakse filtreid kollektsioonide sorteerimiseks, lehekülgedeks jagamiseks ning tõlkimiseks.

2.6 Bootstrap

Bootstrap on raamistik kasutajaliideste kvaliteetseks ja kiireks arendamiseks. Ta pakub palju eeldefineeritud stiile erinevate kasutajaliideste elementide jaoks. Lisaks võimaldab lihtsalt ja kiirelt luua soovitud lehe struktuur. Bootstrap teeb väga lihtsaks kasutajaliideste loomise erineva ekraanisuurusega seadmete jaoks. Veel üks positiivne omadus selle raamistiku puhul on selle ühilduvus kõigi moodsate brauseritega (Chrome, Safari, Firefox, Internet Explorer, Opera) [13].

2.7 BCrypt

Selleks, et kasutajate paroolid ei oleks andmebaasis loetaval kujul, tuleb neid salvestada räsitud kujul. Antud töös kasutatakse selleks eesmärgiks bcrypt algoritmi.

Algoritmi põhimõte seisneb selles, et räsiväärtuse genereerimine oleks arvutuslikult keerukas [14]. See muudab *brute force* rünnakud mittetõhusaks, sest räsi genereerimiseks kulub aega. Üksiku parooli jaoks on see lubatav, kuid suure hulga jaoks ebaotstarbekas, mis tähendab, et algse teksti saamiseks kulub palju aega. Keerukuse määramiseks tuleb kasutada keerukuse astmeid, mis on arvväärtused vahemikus 4 kuni 31. Väärtuse suurendamine ühe võrra suurendab kulukust kaks korda (eksponentsiaalselt) [15].

Iga uue räsi genereerimisel tekitatakse erinev sool. See tagab asjaolu, et kahel ühesugusel sisendil ei oleks sama räsi.

Bcrypt genereerib 60 tähemärgi pikkuse räsi, mis koosneb kasutatava räsialgoritmi tähisest, keerukuse astmest, genereeritava parooli räsiväärtusest (31 tähemärki) ja soolast (22 tähemärki).

Näiteks:
$2a$10$/B4x3FWBZEKLp1g29g//6.q6r/0qb4qU19uDPVpiWZUrbej01M0sy

- $2a$ - räsialgoritmi tähis
- 10$ - kulukus
- /B4x3FWBZEKLp1g29g//6. – sool
- q6r/0qb4qU19uDPVpiWZUrbej01M0sy – parooli räsiväärtus

Kui esmakordselt soovitakse saada mõne sisendi räsi, siis genereeritakse sellele juhuslik sool ning sellega genereeritakse räsi. Kui soovitakse sisendit võrrelda räsitud väärtusega, siis võetakse räsist sool ning genereeritakse sellega sisendi räsi ja võrreldakse räsitud väärtusega..

2.8 PostgreSQL

PostgreSQL on avatud lähtekoodiga objekt relatsiooniline andmebaasisüsteem [16]. Kuigi PostgreSQL pakub palju erinevat funktsionaalsust, siis sellest töös kasutatakse sellest vaid osa: tabelid, vaated ja funktsioonid.

Töös kasutatakse vaateid, et peita mahukate SQL päringute loogikat. Andmebaasi kasutavas rakenduses mahuka päringu kirjutamise asemel saab sooritada SELECT päringu vaatele.

Funktsioone kasutatakse töös selleks, et tabeli reas olevat arvväärtust suurendada või vähendada ühe võrra.

Andmebaasiga ühenduse saamiseks kasutatakse töös JDBC-d (Java Database Connectivity). Tegemist on API-ga, mis võimaldab Javal suhelda erinevate andmebaasidega [17]. Iga andmebaasi jaoks on loodud eraldi draiver, mida JDBC kasutab. Antud töös kasutatakse loomulikult PostgreSQL draiverit.

2.9 Maven

Maven on tarkvaraprojekti haldamise tööriist. Selles projektis kasutatakse seda serverirakenduse sõltuvuste haldamiseks ning WAR faili genereerimiseks. Selle asemel, et erinevate raamistike ja teekide .jar faile käsitsi internetist tõmmata ning siis seejärel nad projekti paigutada, saab vajalikud sõltuvused defineerida Maveni abil, mis nad ise alla tõmbab ja projektile kättesaadavaks teeb. Mavenit kasutava projekti kataloogis on olemas fail pom.xml, mille sees defineeritaksegi kõik vajalikud sõltuvused. Antud projektis on Maveni abil defineeritud kõik sõltuvused: Spring, Spring Web MVC, Spring Security, PostgreSQL draiver ja Jackson.

3. Rakenduse struktuur

Käesolevas peatükis tutvustatakse rakenduse ülesehitust. Kirjeldatakse, mida täpselt võib rakenduses teha ning kuidas toimib kasutajaliides ning kuidas on tehtud serveripoolne osa.

3.1 Rakendus

3.1.1 Rakenduse võimalused

Rakendus võimaldab anda õppejõule nii kirjalik kui ka arvuline hinnang. Arvuline hinnang on vahemikus üks kuni viis punkti. Iga kasutaja saab anda ühe arvulise hinnangu. Õppejõu profiilis kuvatakse keskmist hinnangut. Kirjalik hinnang kujutab endast tavalist kommentaari, kuhu saab kirja panna, mida õppejõust arvatakse. Kirjalikke hinnanguid võib jätta rohkem kui üks. Kirjalikku hinnangut saab hinnata kas positiivselt või negatiivselt ning lisaks saab seda märkida ebasobivaks. Igat arvustust saab hinnata ja raporteerida vaid üks kord. Kõiki eelmainitud tegevusi saab teha anonüümselt, mis tähendab, et ei pea nägema vaeva enda registreerimiseks ja autentimiseks. Kõikidest antud hinnangutest moodustatakse graafik, mis näitab aasta lõikes, õppejõu keskmist arvulist hinnangut. See annab mugava ülevaate sellest, kuidas on õppejõu kvaliteet aastate jooksul muutunud.

Soovitava õppejõu leidmiseks kuvab rakendus süsteemis olemasolevate õppejõudude loendi, mis lisaks õppejõu nimele sisaldab õppejõu keskmist hinnangut ja tema kohta jäetud kommentaaride arvu ning otsimisfunktsiooni, kust saab õppejõudusid nimepidi otsida. Kui soovitud õppejõudu ei leidu, siis saavad kasutajad anonüümselt lisada uusi õppejõude.

Kuna rakendus on täiesti anonüümne, siis tuleb rohkem tegeleda ebasobivate andmete kustutamisega. Selle jaoks on loodud administraatori kasutaja, kes tohib kustutada kõiki kommentaare ja õppejõudusid. Lisaks omab ta ülevaadet raporteeritud kasutajatest.

Administraatoreid saab juurde teha ning kustutada kasutaja, kellel on kõige rohkem õigusi: superuser.

3.1.2 Rollid

Nagu eelnevast punktist võib välja lugeda, siis rakenduses on kokku kolm erinevat rolli:

Anonüümne kasutaja – autentimata kasutaja, kes pääseb ligi õppejõudude loendile ja nende andmetele ning tohib jätta arvustusi ja hinnanguid õppejõududele ja arvustustele.

Administraator – tohib kustutada õppejõude ning nende kohta käivaid arvustusi. Omab ülevaadet arvustustest, mida on märgitud ebasobivaks.

Superuser – omab administraatori õigusi ning tohib administraatoreid lisada ning kustutada.

3.2 Kasutajaliides

3.2.1 Vaated

Rakenduses on loodud 5 erinevat HTML vaadet ning üks põhileht, kuhu Angular sisestab vaateid. Igale vaatele vastab kindel URL ja Angulari kontroller, mis seda vaadet teenindama hakkab.

URL	Vaate nimi	Vaate eesmärk
/#!/admin	admin.html	Rakenduse administraatori paneel. Saab vaadata raporteeritud arvustusi, administraatoreid lisada/kustutada.
/#!/	index.html	Rakenduse avaleht.
/#!/login	login.html	Võimaldab administraatoril rakendusse sisse logida.
/#!/teacher	teacher.html	Kuvab ning võimaldab kõigil lisada õppejõu kohta käivaid hinnanguid ja arvustusi.
/#!/teachers	teachers.html	Kuvab süsteemis olevaid õppejõude ning võimaldab kõigil lisada uusi.

Tabel 1: Rakenduse vaated

Põhileht sisaldab kõiki vajaminevaid JavaScripti faile ning kujundusfaile. Selles lehes on defineeritud ka navigatsiooni riba komponent, mis on ühine kõigi vaadete jaoks. Lisaks sellele on loodud koht, kuhu Angular vaateid sisestama hakkab:

```
<div ng-view></div>
```

Kui kasutaja liigub mõnele URL-le, siis Angular sisestab vastava vaate ülalpool mainitud elementide vahele.

3.2.2 Andmete saamine

Kui kasutaja liigub vaatele, siis seda teenindav kontroller teeb asünkroonse AJAX päringu serverile. Server saadab tagasi andmed JSON formaadis. Saadud andmed salvestatakse muutujasse, mis on ligipääsetav ka vaatest. Kohe, kui muutuja on väärtustatud, ilmuvad saadud väärtused vaatesse.

Olgu näiteks muutuja nimega teachers, kuhu salvestatakse serverilt tulnud andmed:

```
[
   {
      "firstName": "Andres",
      "lastName": "Kaljuveer"
   },
   {
      "firstName": "Paul",
      "lastName": "Ariste"
   }
]
```

```
<li ng-repeat="teacher in teachers">
        {{teacher.firstName}} {{teacher.lastName}}
</li>
```

Antud näite puhul tekitab Angular kaks listi elementi, kus kumbki neist sisaldab ühte eesnime ja perenime paari. Arendaja ei pea eraldi koodi kirjutama, et serverilt saadud andmed vaatesse edastada, vaid piisab nende salvestamisest muutujasse ning vaates andmed muutujast kätte saada.

3.2.3 Autentimine

Antud rakenduses nõuab administraatori vaade autentimist ning kõik ülejäänud vaated võivad olla ligipääsetavad kõigi poolt. Kasutaja autendib ennast kasutajanime ja parooliga. Kui autentimine õnnestub, siis saadab server sisse loginud kasutaja andmed JSON formaadis, mida Angular salvestab küpsisesse (i.k cookie). Kuigi autentimist vajavatele ressurssidele ligipääsu kontrollitakse eraldi nii kasutajaliideses kui ka serveris, siis kasutajaliides ei lase serveriga ühendust võtta, kui kasutaja on autentimata.

Igal vaatel on märgitud, mis õigusi peab kasutaja omama, et antud vaadet näha. Administraatori vaadet saavad näha kasutajad, kes omavad rolli *admin* või *superuser*. Kui rakenduse kasutaja liigub ühelt vaatelt teisele, siis Angular kontrollib, kas kasutajal on õigusi, et soovitud vaadet kuvada. Vaadatakse, kas küpsisesse salvestatud kasutaja roll on sama, mida

nõuab vaade. Kui kasutajal on õigused olemas, siis lastakse ta läbi, vastasel juhul suunatakse kasutaja sisse logimise vaatele.

Kui kasutaja logib välja, siis tema kohta küpsisesse salvestatud andmed kustutatakse ning suunatakse avalehele.

3.2.4 Tõlkimine

Rakendus on loodud kakskeelsena, pakkudes nii eesti kui ka inglise keelset kasutajaliidest. Selle teostamiseks kasutati Angulari moodulit nimega angular-translate. Keelt vahetades asendatakse olemasolevad tekstid tõlgitavate tekstidega, mis tähendab, et lehte uuesti ei laadita, sest tõlkimine toimub dünaamiliselt .

Tõlked on paigutatud iga keele jaoks eraldi JSON faili, antud rakenduses *en.json* (inglise keel) ning *et.json* (eesti keel). Fail koosneb võtmest ning talle vastavast väärtusest. Väärtuseks on tõlgitav tekst ning võtmeks selle teksti unikaalne identifikaator. Moodulile tuleb anda ette JSON faili nimi, kust soovitakse teksti lugeda ning faili asukoht.

Näiteks

```
{
   "HOME": "Home",
   "TEACHERS": "Teachers"
}
```

```
{
   "HOME": "Avaleht",
   "TEACHERS": "Õppejõud"
}
```

Vaatesse tuleb kirjutada Angulari filter koos tõlke identifikaatoriga, mis kuvab teksti valitud keeles. Näiteks `<p>{{'HOME' | translate}}</p>` kuvab paragrahvi koos tekstiga *Home* või *Avaleht,* sõltuvalt sellest, milline keel antud hetkel kasutaja poolt valitud on. Sellisel moel tõlgete pakkumine on väga lihtne, sest uue keele lisamiseks tuleb lisada uus JSON fail ja kirjutada sinna soovitud keele tõlked ning muuta see keel valitavaks läbi kasutajaliidese.

3.2.5 Local storage

Rakendus võimaldab anonüümsetel kasutajatel hinnata kommentaare ja õppejõude ning raporteerida kommentaare. Igat kommentaari või õppejõudu tohiks hinnata vaid üks kord. Kuna kasutajad on anonüümsed, siis server ei saa kindlaks teha, kas mõni kasutaja on juba

hinnanguid jätnud või ei. Server ei jäta meelde ka kasutaja IP aadressi. Järelikult on kasutajaliidese ülesanne tagada, et ei jäetaks topelt hinnanguid.

Selle probleemi lahendamiseks salvestatakse hinnangute andmise andmed veebilehitsejas, kus andmeid võib hoida kahte moodi: sessioonis või lokaalselt. Esimesel juhul andmed kustuvad, kui brauseri aken kinni panna. Teisel juhul andmed säilivad ka peale brauseri sulgemist. Antud töös kasutatakse lokaalset andmehoidlat.

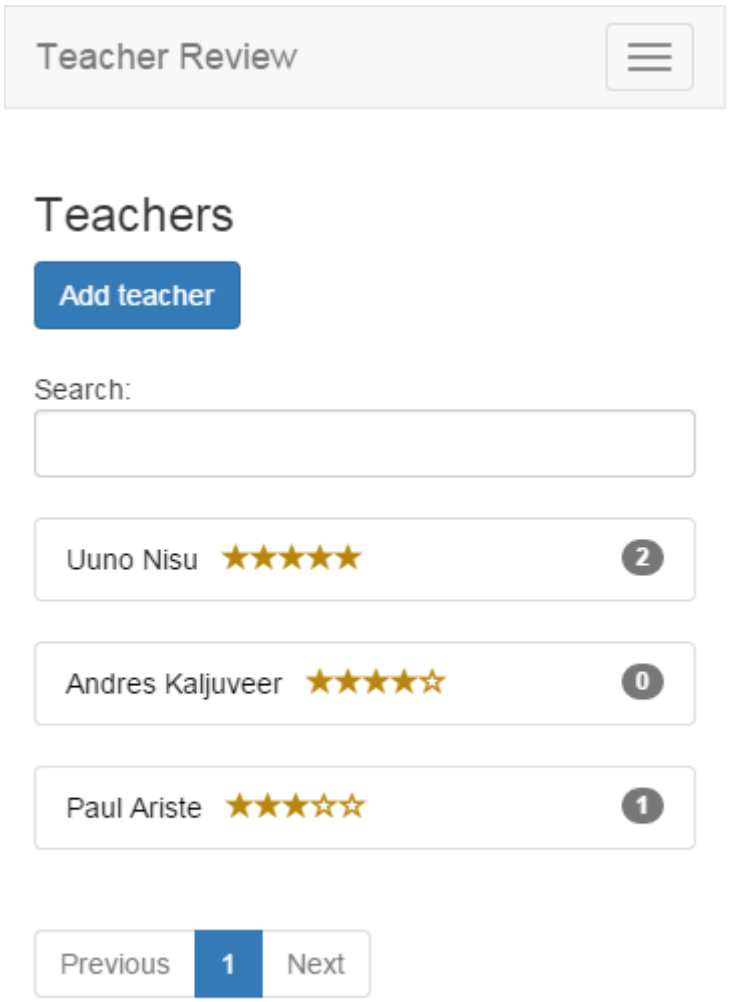
Kui kasutaja jätab hinnangu kommentaarile või õppejõule, siis salvestatakse selle unikaalne identifikaator lokaalselt. Iga kord, kui kasutaja soovib hinnangut jätta, siis kontrollitakse, kas sellise identifikaatoriga õppejõudu/kommentaari on hinnatud. Kui on, siis uuesti hinnata ei lasta. Kui ei ole, siis hinnang arvestatakse saates see serverisse ning märgitakse lokaalselt ära, et hinnang on antud.

Sellel on loomulikult omad nõrkused, nimelt saab kasutaja jätta hinnangut mitmest erinevast brauserist. Samas IP aadresside salvestamine serveri poolt on keerukam ning lubab alamvõrgust anda hinnangut vaid ühel kasutajal. Kõige turvalisem oleks loomulikult anonüümsuse kaotamine sundides kasutajaid kontosid registreerima.

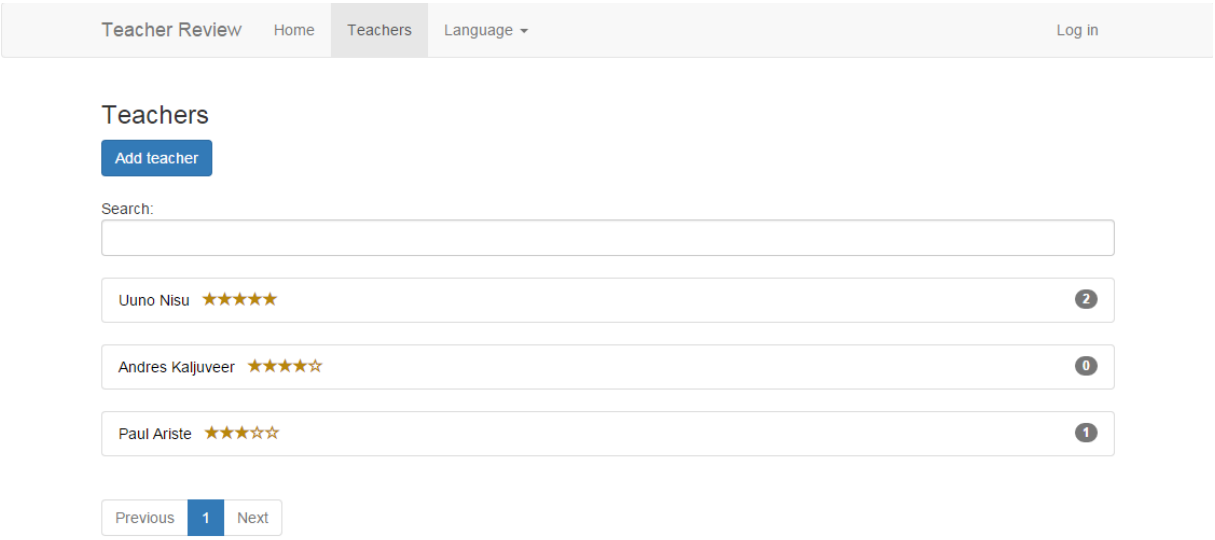
3.2.6 Responsive disain

Rakendus peab olema *responsive*, mis tähendab seda, et seda on mugav kasutada nii suurte seadmetega, millel on suured ekraanid kui ka mobiilidega, millel on ekraanid väikesed. Väiksema ekraaniga seadmetele ei tehtud eraldi rakendust, vaid rakendus muudab oma elementide paigutust vastavalt ekraani suurusele. Selle eesmärgi saavutamiseks kasutati Bootstrap raamistiku pakutavaid võimalusi.

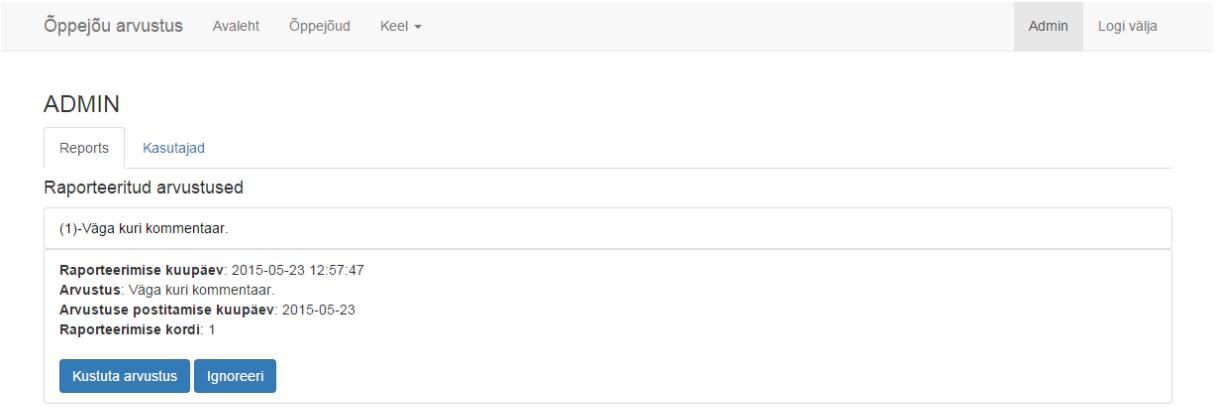
Järgnevalt võrreldakse omavahel nii tavalist õppejõudude loendi vaadet (Joonis 1) kui ka mobiilivaadet (Joonis 2). Joonisel 3 on superkasutaja paneel tavavaade ning joonisel 4 on sama vaade mobiilis.



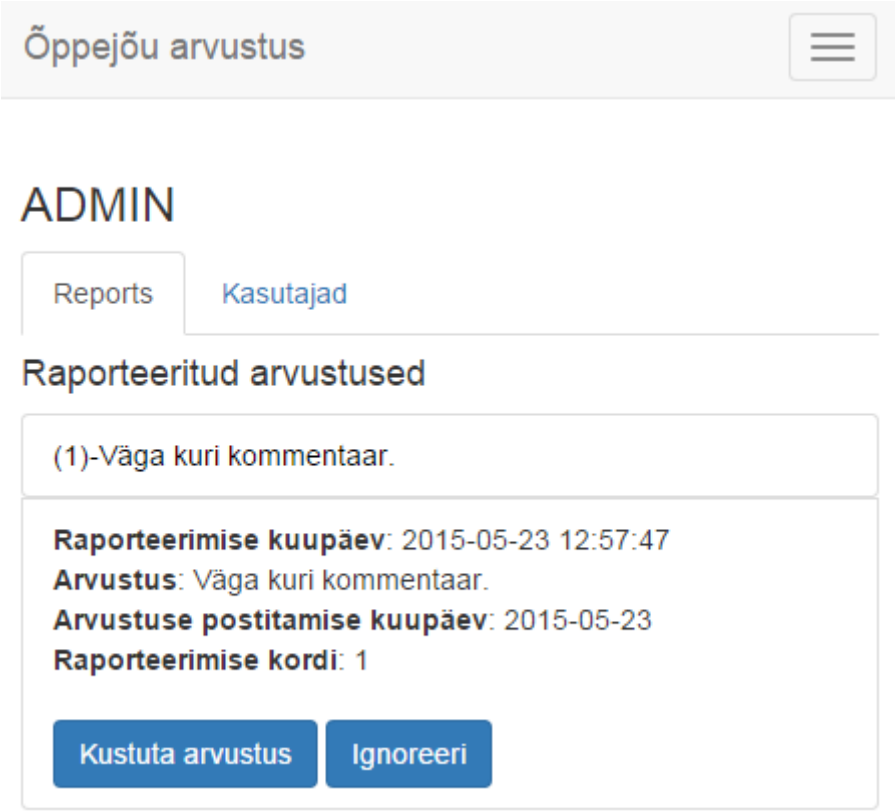
Joonis 1: Õppejõudude loend mobiilivaates



Joonis 2: Õppejõudude loend tavavaates



Joonis 3: Superkasutaja paneel tavavaates



Joonis 4: Superkasutaja paneel mobiilivaates

3.3 Serverirakendus

3.3.1 Autentimine

Kasutajate autentimiseks kasutatakse Spring raamistiku moodulit nimega Spring Security.

Rakenduses on loodud meetodid saamaks kätte kasutaja andmed andmebaasist kasutajanime järgi. Kui tuleb päring sooviga kasutaja autentida, siis Spring automaatselt pöördub selle meetodi poole, mis tagastab kasutaja kasutajanime, räsitud parooli ja kasutaja rollid. Seejärel Spring iseseisvalt räsib päringuga kaasa tulnud parooli etteantud räsialgoritmiga ning võrdleb saadud tulemust andmebaasist saadud räsi väärtusega.

Juhul, kui autentimine õnnestub, siis saadab server tagasi autenditud kasutaja kasutajatunnuse ja kasutaja rollid ning lisaks sellele tagastab HTTP koodi *200 OK*, mis tähistab edukat päringut. Spring jätab meelde kasutaja andmed sessiooni piires. Seega järgmise pöördumise ajal võib näiteks kasutaja rolle küsida Springilt selle asemel, et sooritada uut andmebaasi päringut.

Kui aga kasutajat andmebaasis ei eksisteeri või päringuga tuli vale parool, siis vastab server HTTP koodiga *401 Unauthorized*, mis tähistab ebaõnnestunud autentimist. Sama kood tagastatakse ka juhul, kui autentimata kasutaja sooritab päringu meetoditele, mis nõuavad autentimist.

3.3.2 REST otspunktid

Järgnevalt kirjeldatakse kõiki rakenduse otspunkte. Kõiki andmeid tagastatakse JSON formaadis. Kuna tegemist on REST API-ga, siis selle kasutamine ei ole mõeldud vaid vaadeldava rakenduse jaoks. Seda võivad kasutada ka teised, et saada andmeid enda rakenduse tarvis.

URL **HTTP meetod**	**Kirjeldus**	**Kaasa antud info**
/rest/report/review /{id} POST	Märgib arvustuse ebasobivaks.	{id} – arvustuse identifikaator.
/rest/report/review GET	Tagastab kõik ebasobivaks märgitud arvustused.	-
/rest/report/review/{reportId} PUT	Märgib raporteeritud arvustuse loetuks	{reportId} – raporti unikaalne identifikaator.
/rest/review/{id} POST	Lisab uue arvustuse	{id} – õppejõu unikaalne identifikaator. Arvustuse andmed JSON formaadis.
/rest/review/upvote/{id} PUT	Annab arvustusele positiivse hääle.	{id} – õppejõu unikaalne identifikaator.
/rest/review/downvote/{id} PUT	Annab arvustusele negatiivse hääle.	{id} – õppejõu unikaalne identifikaator.
/rest/review/{id} DELETE	Kustutab arvustuse.	{id} – arvustuse unikaalnde identifikaator.
/rest/score/{id}/{score} PUT	Annab õppejõule hinnangu	{id} – õppejõu unikaalne identifikaator. {score} – õppejõule antud hinnang.
/rest/score/{id} GET	Tagastab õppejõu keskmise hinnangu.	{id} - õppejõu unikaalne identifikaator.

/rest/yearlyavg/{id} GET	Tagastab õppejõu keskmise hinnangu aastate lõikes.	{id} - õppejõu unikaalne identifikaator.
/rest/teacher GET	Tagastab info kõigi õppejõudude kohta.	-
/rest/teacher/{id} GET	Tagastab info õppejõu kohta.	{id} - õppejõu unikaalne identifikaator.
/rest/teacher POST	Lisab uue õppejõu.	Õppejõu andmed JSON formaadis.
/rest/teacher/{id} DELETE	Kustutab õppejõu.	{id} - õppejõu unikaalne identifikaator.
/rest/user POST	Lisab uue administraatori.	Administraatori andmed JSON formaadis.
/rest/user GET	Tagastab kõik administraatorid.	-
/rest/user/{id} DELETE	Kustutab administraatori.	{id} – administraatori unikaalne identifikaator.

Tabel 2: REST otspunktid

3.3.3 Rakenduse kihid

Rakenduses on kokku kolm kihti:

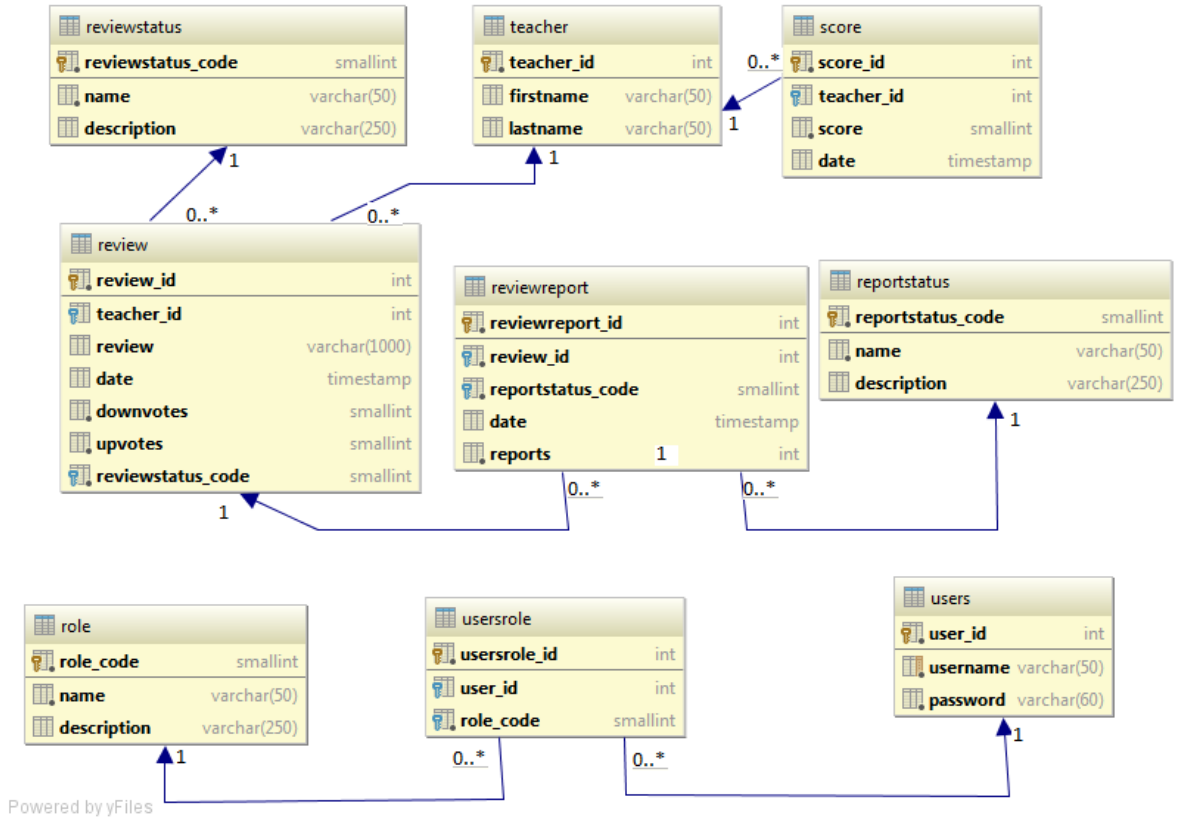
Kontroller – võtab vastu rakendusesse tulnud päringud ning tagastab andmeid. Andmete muutmiseks või saamiseks pöördub service kihi poole. Lisaks autoriseerib kasutajat.

Service – valmistab ette andmeid, et pöörduda andmebaasi poole. Siis pöördub DAO kihi poole.

DAO – (Data Access Object) suhtleb andmebaasiga.

3.4 Andmebaas

Rakenduse tarvis loodi 9 andmebaasi tabelit, mis on kujutatud joonisel 5. Kasutaja rollide hoidmiseks on loodud vahetabel *usersrole*, sest kasutajatel võib olla enam kui üks roll. Lisaks tabelitele loodi kuus vaadet, mis lihtsustavad andmete pärimist rakendusest ning kolm funktsiooni, mis tagavad kommentaari hindamist ja raporteerimist.



Joonis 5: Andmebaasi diagramm

4. Hinnang rakendusele

4.1 Tagasiside

Valminud rakendust anti testida ka mitmele kasutajale. Testimise käigus tuvastati mõned vead ning tehti ka huvitavaid ettepanekuid rakenduse mugavamaks kasutamiseks. Järgnevalt tutvustataksegi saadud tagasisidet.

Kasutajad leidsid vigu arvustuse jätmise juures. Näiteks ühel kindlal moel kirjutatud arvustus lõhkus ära arvustuste kuvamise. Lisaks kurdeti ka selle üle, et mobiilivaates on navigatsiooniriba liiga suur ning võtab liiga palju ruumi. Jäi silma ka see, et õppejõudude, kommentaaride ning kasutajate kustutamisel ei küsitud kasutajalt kinnitust, mille tõttu võis nad kogemata süsteemist ära kustutada. Tagasisidet võeti kuulda ning vead tehti korda.

Peale vigade pakuti ka huvitavaid võimalusi rakenduse mugavamaks ning huvitavamaks tegemiseks. Näiteks pakuti välja, et kogunenud andmeid saaks huvitavalt kuvada. Sellest mõttest lähtudes lisati võimalus vaadata õppejõududele antud hinnanguid möödunud aastate lõikes, et näha, kuidas on õppejõu kvaliteet muutunud. Veel üheks huvitavaks ideeks oli täiustada õppejõu lisamise protsessi. Õppejõudu lisades kuvatakse sama nimega olemasolevaid õppejõudusid vältimaks samade õppejõudude mitmekordset sisestamist.

4.2 Andmemaht

Rakenduse õppejõudude loendi vaate avamisel tehakse päring serveri poole, et saada õppejõudude loendi koos iga õppejõu keskmise hindega ja arvustuste arvuga. Kuigi antud vaates on lisatud lehekülgedeks jagamise funktsionaalsus, saadab server ikkagi loendi, mis koosneb kõigist õppejõududest ning lehekülgedeks jagamine toimub juba kasutajaliidese poolt.

Kui süsteemis on õppejõudude arv tuhandetes, siis võtab loendi genereerimine märgatavalt aega. Seda põhjustab asjaolu, et iga õppejõu kohta tuleb eraldi arvutada tema keskmine hinne ning arvustuste arv. Kui õppejõudusid on süsteemis üle mõne tuhande, siis tuleb oodata mitmeid sekundeid enne kui andmed tagasi tulevad. Juhul kui on vaja väga suurt õppejõudude

hulka toetada, siis tuleb rakenduse struktuuri muuta ning küsida serverilt andmeid osade kaupa.

5. Kokkuvõte

Käesoleva töö põhieesmärke oli kaks. Esiteks, teha valmis veebirakendus, mis võimaldaks õppejõudude kohta tagasisidet jätta. Veebirakendus peab olema mugavalt kasutatav nii arvutite kui ka nutiseadmetega. Teiseks eesmärgiks on õppida kasutama Spring, AngularJS ning Bootstrap raamistikke.

Töö tulemusena valmis plaanitud veebirakendus. Rakendus võimaldab jätta tagasisidet õppejõudude kohta ja õppejõude hinnata ning õppejõudusid süsteemi juurde lisada. Saab lugeda teiste tagasisidet. Loodud rakendus on mugavalt kasutatav nii mobiiliga kui ka suure ekraaniga. Rakendus loodi kasutades erinevaid tänapäevaseid tehnoloogiaid. Loodi kasutajaliides kasutades raamistikke AngularJS lihtsamaks andmete esitamiseks ja serveriga suhtlemiseks ning Bootstrap lehtede kujundamiseks. Serveripoolne rakendus, mis toimib kui REST API, tehti kasutades Spring raamistikku lihtsamaks päringute töötlemiseks ning kasutajate autentimiseks. Kõiki andmeid hoitakse kasutades PostgreSQL andmebaasisüsteemi.

Rakendust on võimalik veel täiendada. Praeguses rakenduses ei ole võimalik eristada sama nimega õppejõude.

Summary

This work had two main aims. Firstly to create an application for the purpose of reviewing teachers. Application has to be modern and easy to use on both smartphones/tablets and computers. Secondly to learn more about different software frameworks (Spring, AngularJS and Bootstrap) and how to implement them.

As a result of the work the originally planned web application was created. Users can browse teachers' ratings and reviews and add their own ratings, reviews and teachers. Application was developed with both smartphones/tablets and computers in mind. Application was created using aforementioned frameworks. Front-end part of the application was created by using AngularJS for easier data presentation and communication with application server and Bootstrap for user interface design. Back-end, which acts as a REST API, was created using Spring framework for handling requests and user authentication. All the data is persisted using PostgreSQL database system.

Application could be enhanced. Right now teachers with the same name cannot be distinguished.

Kasutatud kirjandus

[1] Aprill 2015. [Võrgumaterjal]. Available: http://rest.elkstein.org/.

[2] Aprill 2015. [Võrgumaterjal]. Available: https://github.com/spring-projects/spring-framework.

[3] „Spring Framework,“ Aprill 2015. [Võrgumaterjal]. Available: http://projects.spring.io/spring-framework/. [Kasutatud 11 Aprill 2015].

[4] Aprill 2015. [Võrgumaterjal]. Available: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html.

[5] „Spring Security,“ Aprill 2015. [Võrgumaterjal]. Available: http://projects.spring.io/spring-security/.

[6] „JSON,“ Aprill 2015. [Võrgumaterjal]. Available: http://www.json.org/.

[7] Aprill 2015. [Võrgumaterjal]. Available: http://jackson.codehaus.org/.

[8] „Angular Github,“ Aprill 2015. [Võrgumaterjal]. Available: https://github.com/angular/angular.js.

[9] „Angular Controller,“ Aprill 2015. [Võrgumaterjal]. Available: https://docs.angularjs.org/guide/controller.

[10] „Angular Expression,“ Aprill 2015. [Võrgumaterjal]. Available: https://docs.angularjs.org/guide/expression.

[11] „Angular Directive,“ Aprill 2015. [Võrgumaterjal]. Available: https://docs.angularjs.org/guide/directive.

[12] „Angular Filter,“ Aprill 2015. [Võrgumaterjal]. Available: https://docs.angularjs.org/guide/filter.

[13] „Bootstrap Docs,“ Aprill 2015. [Võrgumaterjal]. Available: http://bootstrapdocs.com/v3.0.0/docs/getting-started.

[14] N. Provos ja D. Mazières, Aprill 2015. [Võrgumaterjal]. Available: https://www.usenix.org/legacy/events/usenix99/provos/provos.pdf.

[15] Aprill 2015. [Võrgumaterjal]. Available: http://docs.spring.io/autorepo/docs/spring-security/4.0.0.RELEASE/apidocs/org/springframework/security/crypto/bcrypt/BCrypt.html.

[16] „PostgreSQL,“ [Võrgumaterjal]. Available: http://www.postgresql.org/about/. [Kasutatud Aprill 2015].

[17] „JDBC Oracle,“ Aprill 2015. [Võrgumaterjal]. Available: http://www.oracle.com/technetwork/java/javase/jdbc/index.html.