|  |  |  |
| --- | --- | --- |
| https://lh3.googleusercontent.com/bp-GG9Gbeny_wpnPNjcSZybeULDTOgEKwJrlqheMuW4tbAaTIwGCY6CjnTKosyFYXLCpZjCaGoj8tVR6QqF1yhFJ44s3o_D616TZbgGuHQ63I6T1TEl90t7jV-piMsxYGvhZOcM | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** | https://lh6.googleusercontent.com/wuhTsaU9YeguPWKYBddtOUmq5ZRI0kRmjmGOx04yuEtwe4exqyOnlQz8mAaWY8Pu337-rfYSaTvlaae9sAV7W44jtyoMUOlmuuIN0dOMuJZNCEz_bC198lRPst3eUoRQvrORVlc |

Миле Праштало

**ВЕБ АПЛИКАЦИЈА ЗА ИНТЕРНЕТ ОБРАЗОВАЊЕ**

ДИПЛОМСКИ РАД

- Основне академске студије -

Нови Сад, 2020

|  |  |  |
| --- | --- | --- |
| https://lh6.googleusercontent.com/wuhTsaU9YeguPWKYBddtOUmq5ZRI0kRmjmGOx04yuEtwe4exqyOnlQz8mAaWY8Pu337-rfYSaTvlaae9sAV7W44jtyoMUOlmuuIN0dOMuJZNCEz_bC198lRPst3eUoRQvrORVlc | УНИВЕРЗИТЕТ У НОВОМ САДУ ●**ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Датум: |
|  |
| **ЗАДАТАК ЗА ИЗРАДУ ДИПЛОМСКОГ (BACHELOR) РАДА** | Лист/Листова: |
|  |

*(Податке уноси предметни наставник - ментор)*

|  |  |
| --- | --- |
| Врста студија: | ☐☒ Основне академске студије  ☐☐ Основне струковне студије |
| Студијски програм: | **Рачунарство и аутоматика** |
| Руководилац студијског програма: | **Проф. др Милан Видаковић** |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: | **Миле Праштало** | Број индекса: | **SW44/16** |
| Област: | **Веб програмирање** | | |
| Ментор: | **Проф. др Милан Видаковић** | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ  И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА  ДИПЛОМСКИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ  ДИПЛОМСКОГ  (BACHELOR)  РАДА:**

|  |
| --- |
| **Веб апликација за интернет образовање** |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
| Задатак рада представља развој веб апликације за интернет образовање. Серверски део апликације ће бити реализован у програмском језику Котлин, коришћењем Spring Boot окружења. Клијентски део задатка ће бити реализовати коришћењем Angular 9 окружења. |

|  |  |
| --- | --- |
| Руководилац студијског програма: | Ментор рада: |
|  |  |

|  |
| --- |
| Примерак за:  ☐☐ Студента;   ☐☐ Ментора |

Образац **Q2.НА.15-04** - Издање 1

Sadržaj

[1 Uvod 5](#_Toc52219533)

[1.1 Postojeća rešenja: 5](#_Toc52219534)

[1.2 Inspiracija 5](#_Toc52219535)

[2 Opis korišćenih tehnologija 6](#_Toc52219536)

[2.1 Kotlin 6](#_Toc52219537)

[2.1.1 Prednosti Kotlina 6](#_Toc52219538)

[2.2 Spring boot 8](#_Toc52219539)

[2.2.1 Spring Data 8](#_Toc52219540)

[2.2.2 Spring Security 10](#_Toc52219541)

[2.3 Angular 10](#_Toc52219542)

[2.3.1 Angular CLI 10](#_Toc52219543)

[2.3.2 Komponente 11](#_Toc52219544)

[2.3.3 Servisi 11](#_Toc52219545)

[2.3.4 Rutiranje 11](#_Toc52219546)

[2.3.5 Quill biblioteka 11](#_Toc52219547)

[3 Specifikacija 13](#_Toc52219548)

[3.1 Model podataka 13](#_Toc52219549)

[3.2 Kontroleri i servisi 13](#_Toc52219550)

[3.3 Angular komponente 13](#_Toc52219551)

[3.4 Tok komunikacije 13](#_Toc52219552)

[4 Implementacija 14](#_Toc52219553)

[4.1 Login i registracija 14](#_Toc52219554)

[4.1.1 Registracija 14](#_Toc52219555)

[4.1.2 Login 16](#_Toc52219556)

[4.2 Početna strana 18](#_Toc52219557)

[4.2.1 Početna strana za profesora 18](#_Toc52219558)

[4.2.2 Početna strana za školu 18](#_Toc52219559)

[4.2.3 Početna strana za studenta 18](#_Toc52219560)

[4.3 Kreiranje kursa 18](#_Toc52219561)

[4.3.1 Tipovi kursa 18](#_Toc52219562)

[4.4 Prikaz kursa 18](#_Toc52219563)

[4.5 Pristupanje školi 18](#_Toc52219564)

[4.6 Pristupanje kursu 18](#_Toc52219565)

[4.7 Kreiranje lekcije 18](#_Toc52219566)

[4.7.1 Nova lekcija 18](#_Toc52219567)

[4.7.2 Linkovana lekcija 18](#_Toc52219568)

[4.8 Prikaz lekcije 18](#_Toc52219569)

[4.8.1 Komentari 18](#_Toc52219570)

[4.9 Rad sa ocenama 18](#_Toc52219571)

[4.10 Izostanci 18](#_Toc52219572)

[4.11 Testovi 18](#_Toc52219573)

[4.11.1 Kreiranje testa 18](#_Toc52219574)

[4.11.2 Rađenje testa 18](#_Toc52219575)

[4.11.3 Ocenjivanje testa 18](#_Toc52219576)

[4.11.4 Ručno ocenjivanje testa 18](#_Toc52219577)

[5 Literatura 19](#_Toc52219578)

# Uvod

Zadatak rada predstavlja razvoj softvera za online obrazovanje. Softver bi trebao da ima ulogu i u učenju na daljinu i kao alat prilikom regularnog izvođenja nastave.

Kako se razvija veb aplikacija, potrebno je razviti klijentski i serverski deo.

Prvo poglavlje je uvodno, u drugom se nalazi opis korišćenih tehnologija. Treće poglavlje sadrži detaljnu specifikaciju i UML dijagrame. Četvrto poglavlje predstavlja opis implementacije i delove programskog koda. U petom poglavlju se nalazi zaključak.

## Postojeća rešenja:

Softveri za online obrazovanje postoje već godinama. Na tržištu su najpopularniji

1. Google Classroom
2. TalentLMS
3. iTunes U

Google Classroom:

Google Classroom je trenutno vodeći softver na polju online obrazovanja. U martu 2020-e godine je dostigao 100 miliona korisnika[1]. Besplatan je za korišćenje za škole. Takođe postoje i brojne aplikacije koje se mogu integrisati sa Google Classroom-om.

TalentLMS

TalentLMS je softver čija je uloga između softvera za online obrazovanje i online obučavanja. Softver je dostupan u besplatnoj i verziji koja se plaća. Besplatna verzija ima ograničenje od maksimum 5 korisnika po kursu i 10 kurseva po korisniku. Cena verzije koja se plaća iznosi od 59 dolara mesečno do 429 dolara mesečno.

iTunes U

iTunes U je aplikacija koju razvija kompanija Apple. Postoji ograničenje od 12 kurseva i 50 učenika po kursu za besplatnu verziju. Takođe aplikacija je dostupna samo na uređajima koji imaju MacOS i iOS operativne sisteme.

## Inspiracija

Inspiracija za ovaj softver su prvenstveno web sajtovi edX i Coursera. To su web sajtovi na kojima najprestižniji svetski univerziteti postavljaju svoje kurseve besplatno. Takođe je kao inspiracija poslužio i Canvas kompanije Instructure koji se koristi na FTN-u.

# Opis korišćenih tehnologija

Serverski deo aplikacije je napisan u programskom jeziku Kotlin [1], verzija 1.3 korišćenjem Spring boot radnog okvira [2]. Za lakše upravljanje podacima je korišćen Spring Data [3], a za kontrolu pristupa aplikaciji je korišćen Spring Security [4].

Za implementaciju klijentskog dela aplijacije je korišćen Angular 9 [5]. Kako bi se pojednostavi rad sa Angularom, korišćen je Angular CLI (*Command Line Interface*) [6]. Za stilizovanje aplikacije je korišćen Angular Material [7]. Takođe je korišćena i biblioteka Quill, odnosno njena adaptacija za Angular ngx-quill [8].

## Kotlin

Kotlin je višeplatformski statički tipiziran programski jezik koji je razvila kompanija JetBrains. Prvi put je prikazan 2011. godine, a verzija 1 je izašla 2015. godine. Kotlin može da se koristi za programiranje:

* Mobilnih uređaja(Mobile cross-platform)
* Native – Kada ne želimo da koristimo virtuelne mašine
* Data science
* Server-side – Kotlin je interoperabilan sa Java bibliotekama pa je moguće koristiti web framework-e koji su razvijeni za Javu. Takođe se razvija i Ktor kao server-side framework.
* Web frontend – Kotlin može da se transpajlira u javascript i da se koristi za izradu frontend aplikacija.
* Android – Kotlin je postao dominantan jezik za izradu android aplikacija

### Prednosti Kotlina

Kotlin je moderniji programski jezik i pojednostavljuje i uvodi neke nove stvari u odnosu na Javu. Detaljnije poređenje se može pronaći na [9].

Type inference za promenljive

Kotlin kompajler može da prepozna tip promenljive kojoj je dodeljena vrednost prilikom inicijalizacije.

val pi = 3.14 // Double

val e = 2.7182818284 // Double

val eFloat = 2.7182818284f // Float, actual value is 2.7182817

Funkcije

Funkcije u kotlinu su first-class, što znači da se mogu smeštati u promenljive, strukture podataka, prosleđivati kao argumenti i biti povratne vrednosti funkcija višeg reda.

Smart casts

Ako želimo da proverimo da li je objekat instanca neke pod klase možemo da koristimo operator is. Ako koristimo operator is u if ili when uslovu objekat je automatski kastovan u pod klasu.

Primer:

Imamo klasu ChildClass koja nasleđuje klasu ParentClass i koja ima metodu „testMethod“.

Java:

ParentClass a = new ChildClass();

If(a instanceof ChildClass){

ChildClass child = (ChildClass) a;

child.testMethod(); ki

}

Kotlin:

val a:ParentClass = ChildClass() //Može se i izbeći navođenje tipa

if(a is ChildClass){

a.testMethod()  
}

Stream funkcije.

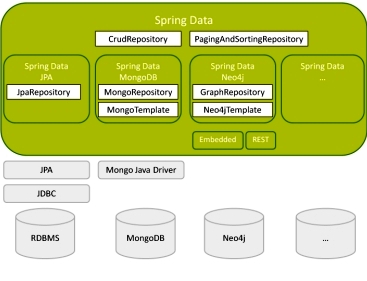
Stream funkcije poput filter, map i drugih su u Kotlinu urađene čitljivije nego u Javi

## Spring boot

Spring boot je radni okvir koji služi da se pojednostavi razvoj spring aplikacija. Prilikom inicijalizacije projekta na https://start.spring.io/ Mogu da se izaberu build alati (Maven ili Gradle), programski jezik (Java, Kotlin ili Groovy) kao i verzija Spring boot-a.

### Spring Data

Spring Data je poseban projekat čiji je cilj da pojednostavi rad sa bazama podataka u Spring projektu. Omogućuje lakši rad sa SQL i NoSQL bazama podataka koristeći repozitorijume.

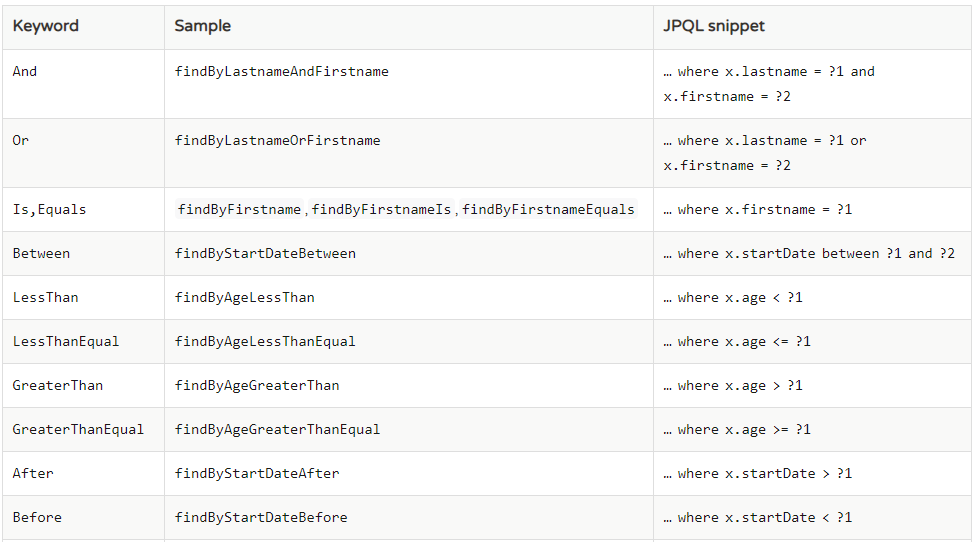


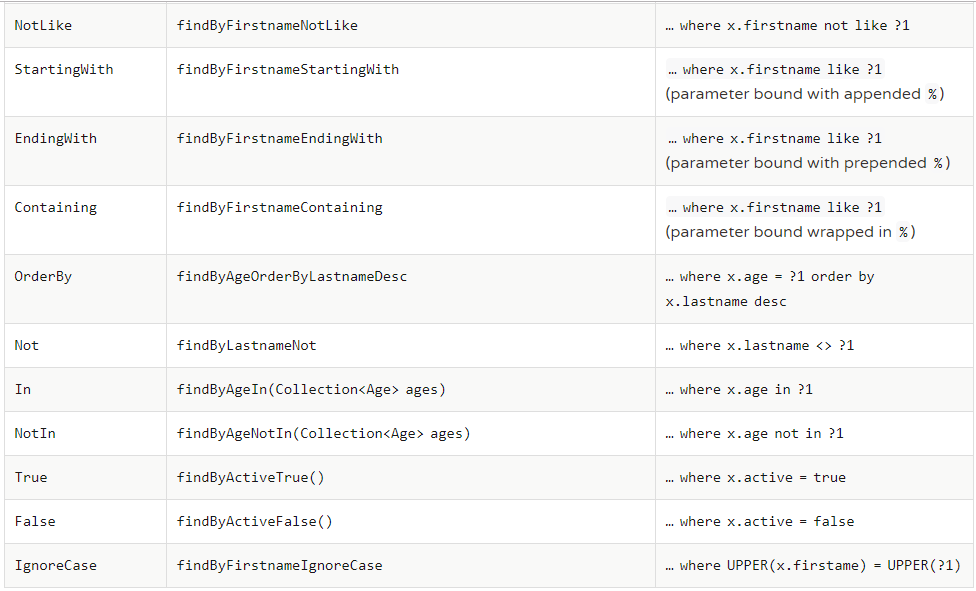
Slika 1.1 Spring Data https://www.infoq.com/articles/spring-data-intro/

Spring Data JPA omogućava kompatibilnost sa JPA (Java Persistance API). Uvodi repozitorijume za rad sa entitetima. Repotitorijum je interfejs koji nasleđuje klasu Repository. Da bi definisali novi repozitorijum za entitet potrebo je kreirati novi interfejs koji će nasleđivati neku od specijalizacija interfejsa Repository i proslediti klasu entiteta i klasu id-a.  
Specijalizacije Repository interfejsa su:

* *CrudRepository* – Specijalizacija Repository interfejsa. Sadrzi CRUD  (*create, update, delete*) operacije nad entitetom,
* *PagingAndSortingRepository* – specijalizacija CRUD repozitorijuma,  omogućuje paginaciju i sortiranje i
* *JPARepository* – specijalizacija *PagingAndSorting* interfejsa. Sadrži dodatnu podršku za JPA.

Spring Data eleminiše potrebu ručnog pisanja upita za bazu podataka time što se upiti automatski generišu iz naziva metoda koje su deklarisane u interfejsu koristeći nazive atributa i ključne reči.





<https://www.javaguides.net/2018/11/spring-data-jpa-query-creation-from-method-names.html>

### Spring Security

Spring Security je radni okvir koji omogućuje autentifikaciju i autorizaciju u spring aplikacijama. Omogućava konfigurisanje zaštite pristupa REST ulaznim tačkama aplikacije specificiranjem korisničkih uloga koje imaju pristup.

Pored zaštite ulaznih tačaka, Spring Security sadrži i password encoder.

## Angular

Angular je radni okvir za razvoj klijentskih veb aplikacija. Razvijen je od strane Google-a, nastao je temeljima Angular JS, sa kojim ima malo sličnosti. Inicijalna verzija 2.0 je predstavljena 2016. godine. Trenutnoje najnovija verzija verzija 10.1. U ovoj projektu je korišćena verzija 9, prvenstveno zbog kompatibilnosti sa bibliotekama.

### Angular CLI

Angular CLI je alat koji se koristi iz komandne linije. Ubrzava razvijanje Angular aplikacija tako što pojednostavljuje kreiranje novog projekta, novih komponenti i servisa. Takođe olakšava održavanje sa build i lint funkcijama.

### Komponente

Komponente su osnovni gradivni blokovi Angular aplikacije. Komponente se sastoje od TypeScript klase, Html fajla i CSS fajla. Komponenta u TS fajlu je obeležena sa @Component dekoratorom. Takođe se u TS fajlu nalaze i:

* *selector* – naziv HTML elementa koji koristimo prilikom kreiranja komponente,
* *templateUrl* – Putanja do HTML template fajla
* *styleUrls* – Putanja do CSS fajla koji se primenjuju za tu komponentu

### Servisi

Angular servisi su singleton objekti koji se instanciraju samo jednom tokom života aplikacije. Osnovna uloga servisa je da organizuju poslovnu logiku, podatke i funkcionalnosti između različitih komponenti.

### Rutiranje

Rutiranje služi za navigaciju između komponenti. Na osnovu URL adrese se određuje koja komponenta će se učitati. Putanje se specificiraju u app-routing.module.ts fajlu. Prikaz dela putanja u projektu se nalazi ispod.

const routes: Routes = [

  {path: '', component: IndexComponent, canActivate: [NotLoggedInGuard]},

  {path: 'login', component: LoginComponent, canActivate: [NotLoggedInGuard]},

  {path: 'register', component: RegisterComponent, canActivate: [NotLoggedInGuard]},

  {path: 'home', component: HomeComponent, canActivate: [LoggedInGuard]},

  {path: 'create-course', component: CreateCourseComponent, canActivate: [LoggedInGuard]},

  {path: 'edit-course/:id', component: EditCourseComponent, canActivate: [LoggedInGuard]},

  {path: 'course/:id', component: CourseDetailsComponent, canActivate: [LoggedInGuard]},

  {path: 'school/:id', component: SchoolDetailsComponent, canActivate: [LoggedInGuard]}

];

### Quill biblioteka

Quill editor je besplatan open source text editor. Prilikom izrade aplikacije je korišćen ngx-quill, wrapper oko Quill biblioteke za Angular. Koristi se tako što se u html fajlu doda element „quil-editor“. To je dovoljno da se kreira polje za unos teksta sa toolbar-om. Ovo će kreirati editor sa svim elementima toolbar-a. Moguće je definisati toolbar samo sa određenim elementima. U tom slučaju je elementu potrebno proslediti elemente toolbara. Svaka promena prilikom pisanja ili menjanja teksta pokreće događaj „onContentChanged“. U objektu koji se prosledi funkciji se nalazi i sadržaj editora u html formatu.

<quill-editor (onContentChanged)="quillEditorChange($event)" ()"></quill-editor>

# Specifikacija

## Model podataka

## Kontroleri i servisi

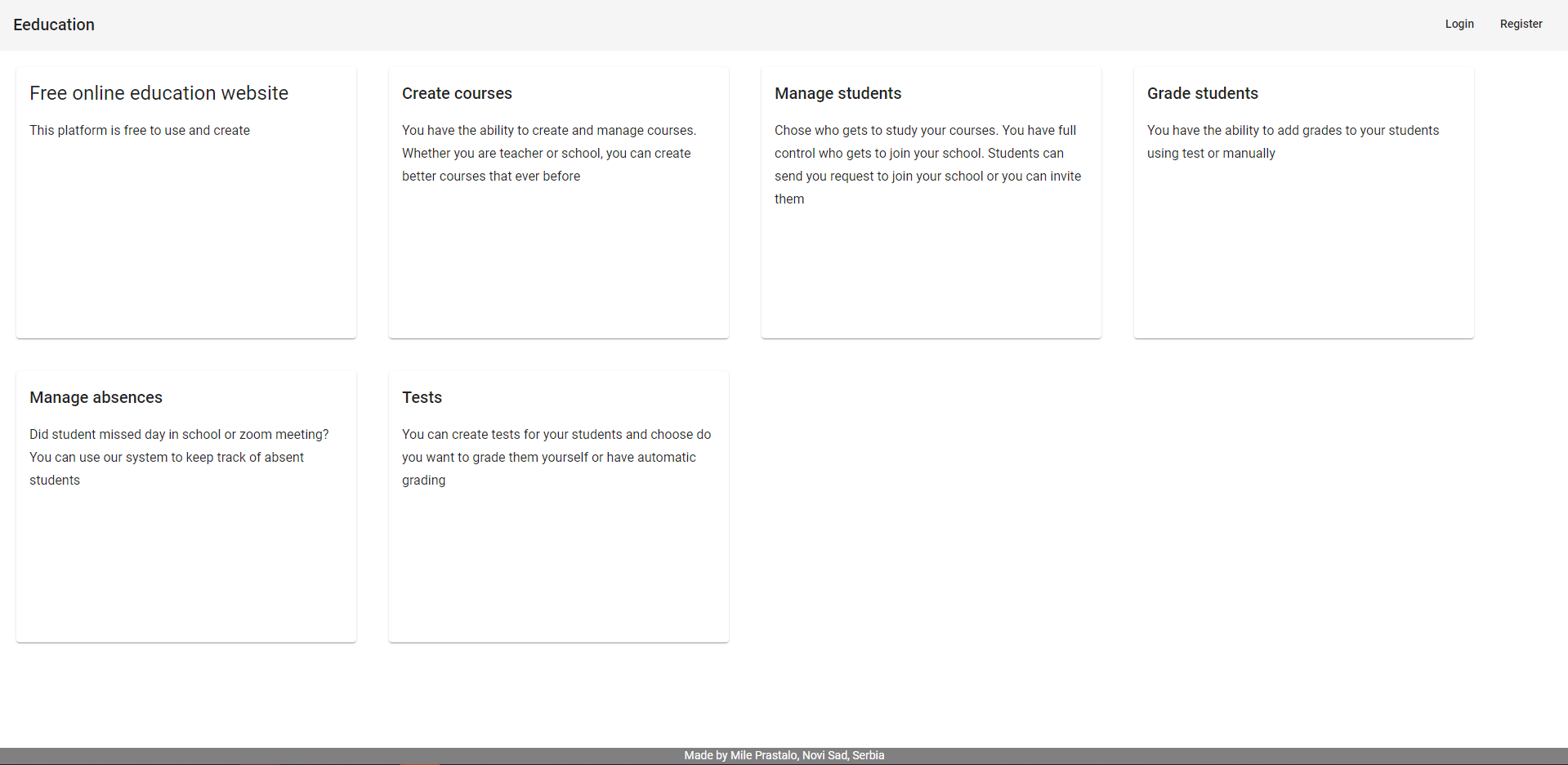
## Angular komponente

## Tok komunikacije

# Implementacija

## Login i registracija

Kako bi korisnik mogao da pristupi funkcionalnostima aplikacije potrebno je da ima nalog u sistemu. Neregistrovani korisnik ima pristup početnoj stranici na kojoj se nalazi promotivni sadržaj, login stranici i stranici za registraciju.



Slika: Početna stranica

Klikom na dugme login korisnik je prebačen na formu za logovanje, a klik na dugme register ga prebacuje na formu za registraciju.

### Registracija

U formi za registraciju korisnik unosi svoje podatke. Potrebno je da unese svoje

* ime,
* email adresu,
* šifru,
* zatim da ponovi šifru,
* izabere državu i
* izabere tip korisnika

Tipovi korisnika mogu biti:

* Student,
* Nastavnik (Teacher) i
* Škola (School)

Nakon što korisnik unese sve podatke i pritisne dugme za registraciju, poziva se metoda register iz Authentication Controller-a. Ispod je prikazan deo AuthenticationController-a sa metodom register.

@RestController  
@RequestMapping("auth")  
@CrossOrigin  
class AuthenticationController(private val authenticationService: AuthenticationService) {  
  
 @PostMapping(value = ["register"], consumes = [MediaType.*APPLICATION\_JSON\_VALUE*], produces = [MediaType.*APPLICATION\_JSON\_VALUE*])  
 fun register(@RequestBody registrationRequest: RegistrationRequest): ResponseEntity<RegistrationResponse> {  
 val response = authenticationService.register(registrationRequest)  
 return ResponseEntity(RegistrationResponse(response), HttpStatus.OK)  
 }  
}

Metoda za registraciju u kontroleru

Metoda u kontroleru poziva metodu u Authentication Service-u i od nje kao povratnu vrednost dobija string sa porukom u slučaju uspešne registracije.

@Service  
class AuthenticationService(private var authenticationManager: AuthenticationManager,  
 private var jwtUserDetailsService: JwtUserDetailsService,  
 private val jwtTokenUtil: JwtTokenUtil,  
 private val authorityRepository: AuthorityRepository,  
 private val schoolService: SchoolService,  
 private val teacherService: TeacherService,  
 private val studentService: StudentService,  
 private val userService: UserService,  
 private val emailService: EmailService) {  
  
  
 fun register(registrationRequest: RegistrationRequest): String {  
 val authorities = HashSet<Authority>()  
 val bc = BCryptPasswordEncoder()  
 return when (registrationRequest.userType) {  
 UserType.SCHOOL -> {  
 authorities.add(authorityRepository.findByName("SCHOOL")!!)  
 var school = School(null, registrationRequest.email, bc.encode(registrationRequest.password), registrationRequest.name, authorities, false, HashSet(), HashSet(), HashSet(), HashSet(), HashSet())  
 school = schoolService.save(school)  
 emailService.sendRegistrationEmail(school)  
 "School has been successfully registered"  
 }  
 UserType.TEACHER -> {  
 authorities.add(authorityRepository.findByName("TEACHER")!!)  
 var teacher = Teacher(null, registrationRequest.email, bc.encode(registrationRequest.password), registrationRequest.name, authorities, false, HashSet(), HashSet(), HashSet())  
 teacher = teacherService.save(teacher)  
 emailService.sendRegistrationEmail(teacher)  
 "Teacher has been successfully registered"  
 }  
 UserType.STUDENT -> {  
 authorities.add(authorityRepository.findByName("STUDENT")!!)  
 var student = Student(null, registrationRequest.email, registrationRequest.name, bc.encode(registrationRequest.password), authorities, false, HashSet(), HashSet(), HashSet(), HashSet(), HashSet())  
 student = studentService.save(student)  
 emailService.sendRegistrationEmail(student)  
 "Student has been successfully registered"  
 }  
 }  
 }  
 }  
 }

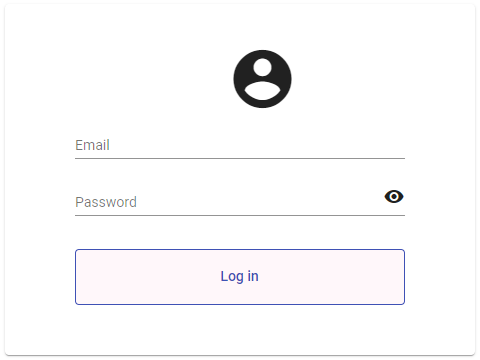
}

Metoda register u authentication service-u

Nakon što je korisnik uspešno registrovan, na email adresu mu stiže link na koji treba da klikne kako bi potvrdio svoj nalog. Nakon čega može da se uloguje na sistem.

### Login

Ukoliko korisnik ima nalog, potrebno je da se uloguje. Forma za loin se sastoji od polja za email adresu i polja za lozinku.



Forma za login

Nakon što korisnik pritisne dugme za login poziva se metoda onLoginSubmit na klijentskoj strani u fajlu login.component.ts

onLogInSubmit() {  
 const loginData = new AuthenticationRequest(this.email, this.password);  
 this.authenticationService.login(loginData).subscribe(  
 (response => {  
 if (response != null) {  
 ***localStorage***.setItem('token', response.token);  
 this.snackBar.open('Logged In successfully.');  
 const jwt: JwtHelperService = new JwtHelperService();  
 const info = jwt.decodeToken(response.token);  
 ***console***.log(info);  
 const role = info.role[0].authority;  
 ***localStorage***.setItem('role', info.role[0].authority);  
 ***localStorage***.setItem('id', ***String***(response.user.id));  
 this.snackBar.open('Logged In successfully.');  
  
 if (role === 'SCHOOL') {  
 this.router.navigateByUrl('/school-home');  
 } else if (role === 'STUDENT') {  
 this.router.navigateByUrl('/student-home');  
 } else if (role === 'TEACHER') {  
 this.router.navigateByUrl('/teacher-home');  
 }  
 }  
 }),  
 (error => {  
 this.snackBar.open(error.error.message);  
 }));  
}

Login frontend

Metoda login u Authentication Service-u na klijentskoj strani poziva metodu login u AuthenticationController-u

@RestController  
@RequestMapping("auth")  
@CrossOrigin  
class AuthenticationController(private val authenticationService: AuthenticationService) {  
  
 @PostMapping(value = ["login"], consumes = [MediaType.*APPLICATION\_JSON\_VALUE*], produces = [MediaType.*APPLICATION\_JSON\_VALUE*])  
 fun login(@RequestBody authenticationRequest: AuthenticationRequest): ResponseEntity<AuthenticationResponse> {  
 val token = authenticationService.login(authenticationRequest)  
 val user = authenticationService.getUserByEmail(authenticationRequest.email)  
 val userDto = UserBasicDto(user.id, user.email, user.name)  
 val authenticationResponse = AuthenticationResponse(token, userDto)  
 return ResponseEntity(authenticationResponse, HttpStatus.OK)  
 }

}

Metoda login u Authentication Service-u na serverskoj strani proverava da li korisnik postoji, yatim generiše JSON Web Token koji pored osnovih podataka sadrži i ulogu korisnika. Kao odgovor se šalje objekat tipa AuthenticationResponse koji sadrži token i osnovne podatke o korisniku (id, email i ime). Kada odgovor stigne klijentu, klijent sačuva token, ulogu i id korisnika u local storage i na osnovu uloge korisnik je preusmeren na početnu stranu.

## Početna strana

### Početna strana za profesora

Nastavnik nakon uspešno logovanja vidi tab sa kursevima gde može da vidi sve kurseve na kojima je prijavljen. Pored taba sa kursevima se nalazi tab sa Školama gde nastavnik može da vidi škole u kojima predaje. Pored škola se nalazi tab sa pretragom.

### Početna strana za školu

Škola nakon uspešnog logovanja vidi tabove

* Courses – Škola može da vidi listu svojih kurseva,
* Teachers – Lista nastavnika koji su deo škole
* Teacher join request – Lista nastavnika koji su poslali zahtev za učlanjenje u školu. Škola može da prihvati ili da odbije zahtev.
* Student join request - Lista studenata koji su poslali zahtev za učlanjenje u školu. Škola može da prihvati ili da odbije zahtev.
* Teacher invite – Škola može da pošalje pozivnicu nastavniku
* Student invite – Škola može da pošalje pozivnicu studentu

### Početna strana za studenta

Student nakon uspešno logovanja vidi tab sa kursevima gde može da vidi sve kurseve na kojima je prijavljen. Pored taba sa kursevima se nalazi tab sa Školama gde student može da vidi škole koje pohađa. Pored škola se nalazi tab sa ocenama gde student može da vidi svoje ocene grupisane po kursu. Pored Ocena se nalazi pretraga gde student može da pretraži škole i kurseve.

## Kreiranje kursa

### Tipovi kursa

## Prikaz kursa

## Pristupanje školi

## Pristupanje kursu

## Kreiranje lekcije

### Nova lekcija

### Linkovana lekcija

## Prikaz lekcije

### Komentari

## Rad sa ocenama

## Izostanci

## Testovi

### Kreiranje testa

### Rađenje testa

### Ocenjivanje testa

### Ručno ocenjivanje testa

# Literatura

[1] <https://kotlinlang.org/>

[2] <https://spring.io/projects/spring-boot>

[3] <https://spring.io/projects/spring-data>

[4] <https://spring.io/projects/spring-security>

[5] <https://angular.io/>

[6] <https://cli.angular.io/>

[7] <https://material.angular.io/>

[8] <https://github.com/KillerCodeMonkey/ngx-quill>

[2] Comparison to Java Programming Language <https://kotlinlang.org/docs/reference/comparison-to-java.html>