



## **IT355 Web sistemi 2**

Projekat: System for managing employees

**Vanja Cekić 4835**

**[vanja.cekic.4835@metropolitan.ac.rs](mailto:vanja.cekic.4835@metropolitan.ac.rs)**

**U Nišu, 19.06.2024.**

# SADRŽAJ

PREDLOG TEMA.....	3
SVRHA .....	3
KORIŠĆENI ALATI.....	3
FUNKCIONALNI ZAHTEVI .....	3
GOST .....	3
KORISNIK (EMPLOYEE) .....	4
ADMIN (EMPLOYEE KOJI JE CEO,DIREKTOR...) .....	4
NEFUNKCIONALNI ZAHTEVI .....	5
SLUČAJEVI KORIŠĆENJA.....	6
ARHITEKTURA .....	7
STRUKTURA PODATAKA.....	8
BACKEND.....	8
FRONTEND.....	8
KLJUČNE KOMPONENTE I NJIHOVI OPISI.....	8
ENTITETI.....	8
REPOZITORIJUMI .....	9
SERVISI.....	9
KONTROLERI.....	9
SIGURNOST.....	9
BAZA PODATAKA .....	9
TESTIRANJE .....	10
TESTIRANJE HTTP STRANICA.....	12
IMPLEMENTACIJA I PRIKAZ SISTEMA .....	15
BUĐUĆA POBOLJŠANJA .....	17
ZAKLJUČAK.....	17
LITERATURA .....	18

## PREDLOG TEMA

Sistem za Upravljanje Zaposlenima je dizajniran da olakša menadžment zaposlenih unutar kompanije i da pojednostavi administrativne zadatke vezane za upravljanje zaposlenima, čineći nadzor nad radnom snagom efikasnijim.. Pruža funkcionalnosti za pregled svih zaposlenih, upravljanje njihovim odeljenjima, praćenje radnihsati, plata i rukovanje zahtevima za odsustvo.

## SVRHA

Glavna svrha ovog projekta je kreiranje sistema koji omogućava upravljanje zaposlenima. Ovo uključuje:

- Pregled svih zaposlenih i njihovih detalja.
- Povezivanje zaposlenih sa njihovim odeljenjima.
- Praćenje mesečnih radnih sati i plata zaposlenih.
- Upravljanje zahtevima za odsustvo i odobravanje istih zahteva.

Sistem je dizajniran da poveća efikasnost upravljanja zaposlenima u bilo kojoj organizaciji. Skalabilan je i može se dalje razvijati kako bi zadovoljio specifične potrebe različitih kompanija.

## KORIŠĆENI ALATI

Za izradu projekta korišćeni su sledeći alati i tehnologije:


- Spring Boot 3.3.0
- Angular 17

## FUNKCIONALNI ZAHTEVI

Funkcionalni zahtevi su specifični zahtevi koji opisuju funkcionalnosti ili operacije koje sistem treba da obavi. Onidefinišu šta sistem treba da radi kako bi zadovoljio potrebe korisnika ili rešio određene probleme.

### GOST

Gost je neautentifikovani korisnik sistema. Gostu je omogućeno da pristupi samo login formi kako bi nastavio sadaljnim koršćenjem sistema.

1	 <b>1.</b>	<b>Guest</b>	REQ_0001	Undefined	0	Undefined	Draft	
2	<b>1.1</b>	<b>Login</b> Guest can fill out login form to log into the system and to start his session.	REQ_0002	Undefined		Undefined	Draft	

## KORISNIK (EMPLOYEE)

Korisnik nakon što se ulogovao, ima mogućnost da vidi svoj profil i sve detalje vezane za njega. Takođe može podneti zahtev za odsustvo sa opravdanim razlogom. Korisnik treba da izabere datum kada će uzeti odsustvo i koliko dana će biti odsutan, uz napomenu sa objašnjenjem razloga za odsustvo...

3	2.	<b>User</b>	REQ_0003	Undefined	0	Undefined	Draft
4	2.1	<b>See profile</b> User is available to see his profile and all details for him.	REQ_0004	Undefined		Undefined	Draft
5	2.2	<b>Submit leave request/require</b> User can submit his leave request with justified reason (select date when he will take the leave and how many day will he be gone, note with explanation, reason for leave...).	REQ_0005	Undefined		Undefined	Draft

## ADMIN (EMPLOYEE KOJI JE CEO,DIREKTOR...)

Administrator ima mogućnost da vidi sve zaposlene u kompaniji sa svim detaljima za svakog zaposlenog. Takođe mogućnost da doda novog zaposlenog u sistem, da ažurira podatke o zaposlenima u sistemu i da obriše zaposlenog iz sistema ukoliko taj zaposleni više ne radi u kompaniji. Administrator ima mogućnost da vidi sva odeljenja u kompaniji. Može da doda novo odeljenje u sistem, da ažurira podatke o odeljenjima u sistemu i da obriše odeljenje iz sistema. Administrator ima mogućnost da vidi sve tipove odsustva u kompaniji. Administrator može da vidi sve podnete zahteve za odsustvo i da odobri ili odbije zahtev za odsustvo.

3.	<b>Admin</b>	REQ_0006	Undefined	0	Undefined	Draft
3.1	<b>Managing employee</b>	REQ_0011	Undefined	0	Undefined	Draft
3.1.1	<b>See every employee</b> Admin is allowed to see all employees in the company (every details for each employee)	REQ_0007	Undefined		Undefined	Draft
3.1.2	<b>Add employee</b> Admin is able to add new employee to system	REQ_0008	Undefined		Undefined	Draft
3.1.3	<b>Update employee</b> Admin is able to update any employee from system	REQ_0009	Undefined		Undefined	Draft
3.1.4	<b>Delete employee</b> Admin can delete any employee from system if that employee is not working in company	REQ_0015	Undefined		Undefined	Draft
3.2	<b>Managing department</b>	REQ_0010	Undefined	0	Undefined	Draft
3.2.1	<b>See every department</b> Admin is allowed to see all departments in company	REQ_0012	Undefined		Undefined	Draft
3.2.2	<b>Add department</b> Admin is able to add new department to system	REQ_0013	Undefined		Undefined	Draft
3.2.3	<b>Update department</b> Admin is able to update any department from system	REQ_0014	Undefined		Undefined	Draft
3.2.4	<b>Delete department</b> Admin can delete any department from system	REQ_0017	Undefined		Undefined	Draft
3.3	<b>Managing leave</b>	REQ_0016	Undefined	0	Undefined	Draft
3.3.1	<b>See every leave</b> Admin is allowed to see every type of leave in the company	REQ_0018	Undefined		Undefined	Draft
3.3.2	<b>See every submitted leave request/require</b> Admin is able to see every submitted form for leave request/require	REQ_0019	Undefined		Undefined	Draft
3.3.3	<b>Approve leave request/require</b> Admin can approve leave request/require	REQ_0020	Undefined		Undefined	Draft

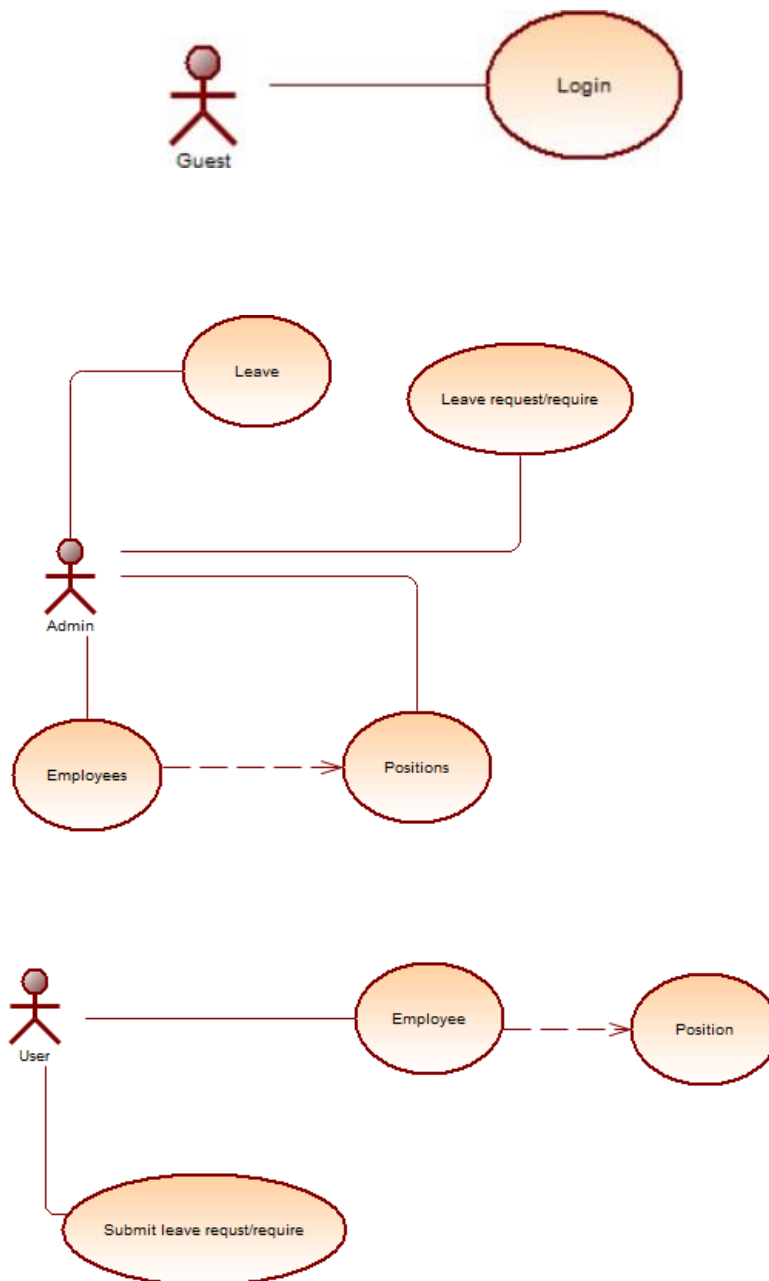
## NEFUNKCIONALNI ZAHTEVI

Nefunkcionalni zahtevi definišu karakteristike i ograničenja sistema koja nisu direktno povezana sa funkcionalnostima sistema. Oni se odnose na aspekte performanse, pouzdanosti, bezbednosti, korisničkog iskustva i drugih nefunkcionalnih atributa sistema. Ovi zahtevi su usmereni na obezbeđivanje kvaliteta i performansi sistema.

	Title ID	Full Description	Code	Priority	Workload	Risk	Status
→	1.	<b>Security</b> User and employee data must be protected by encryption in accordance with industry standards. The system must support security measures such as protection against SQL injection, XSS, and CSRF attacks. Role-based access control must be implemented to ensure appropriate access levels for different users.	REQ_0001	Undefined		Undefined	Draft
2	2.	<b>Performance</b> The system must support up to 1000 simultaneous users without significant performance degradation.	REQ_0002	Undefined		Undefined	Draft
3	3.	<b>Reliability</b> The system must be available 99.9% of the time during operational hours.	REQ_0003	Undefined		Undefined	Draft

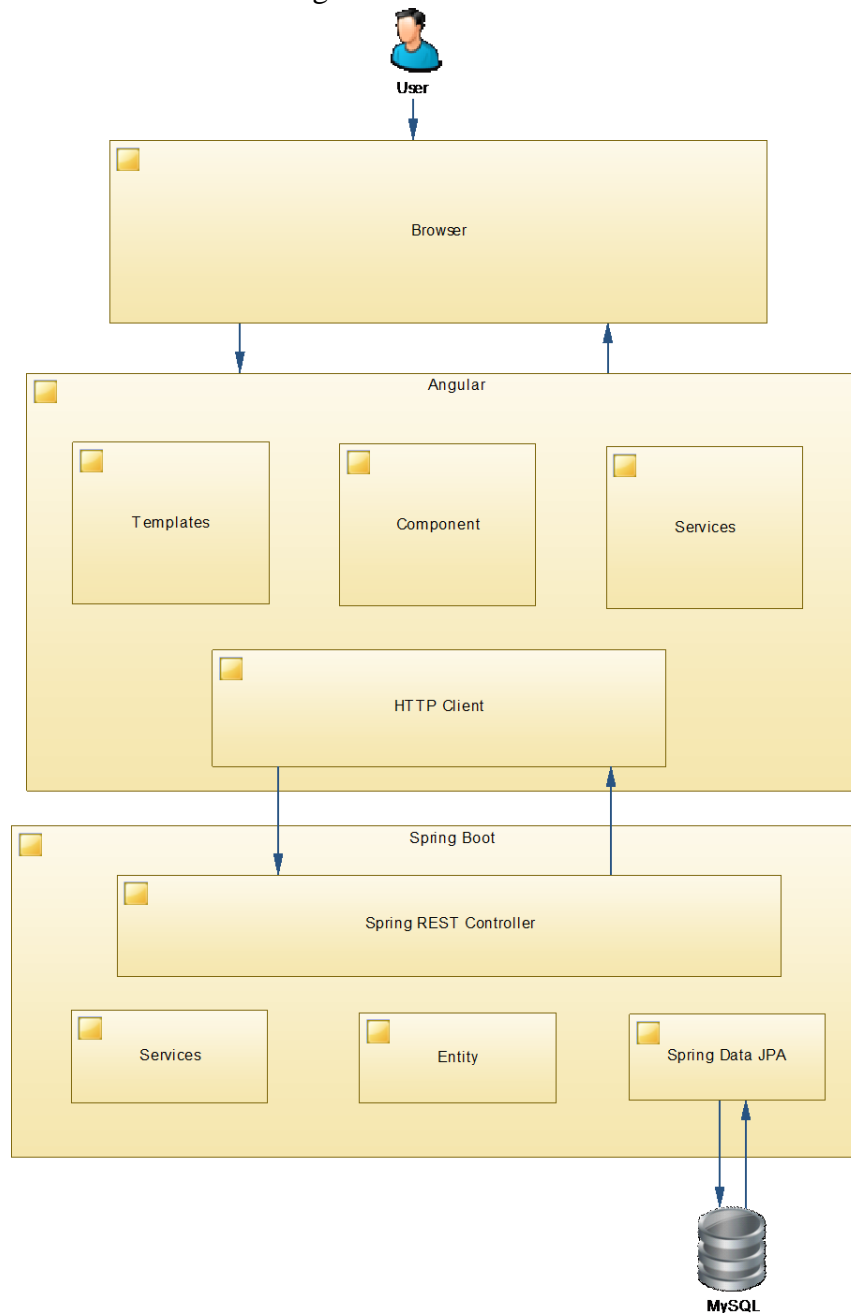
## SLUČAJEVI KORIŠĆENJA

Dijagram slučaja korišćenja je grafički prikaz interakcija između korisnika sistema i samog sistema. Ovaj dijagram služi za modeliranje funkcionalnosti sistema i identifikaciju različitih uloga i njihovih interakcija sa sistemom.



## ARHITEKTURA

Arhitektura aplikacije koja koristi Angular za front-end i Spring Boot za back-end omogućava razvoj snažnih web aplikacija. Angular nudi fleksibilnost za kreiranje interaktivnih korisničkih interfejsa, dok Spring Boot omogućava brzu izgradnju sigurnih i skalabilnih back-end funkcionalnosti. Komunikacija između front-enda i back-enda se odvija putem HTTP protokola, omogućavajući prenos podataka i izvršavanje akcija. Ova arhitektura jasno razdvaja odgovornosti između front-enda i back-enda, olakšava održavanje i proširivanje aplikacije, te omogućava integraciju sa različitim sistemima i uslugama.



## STRUKTURA PODATAKA

Projekat je strukturiran u nekoliko komponenti, od kojih je svaka odgovorna za različite aspekte sistema:

### BACKEND

Backend projekta je razvijen koristeći Spring Boot. Uključuje sledeće glavne komponente:

- **Entiteti:** Predstavljaju osnovne podatke sistema, kao što su Zaposleni, Odeljenje, Odsustvo, itd.
- **Repozitorijumi:** Interfejsi za operacije nad bazom podataka, omogućavaju CRUD operacije nad entitetima.
- **Servisi:** Sloj poslovne logike koji upravlja interakcijom između repozitorijuma i kontrolera.
- **Kontroleri:** Rukovode HTTP zahtevima i mapiraju ih na odgovarajuće metode servisa.
- **Sigurnost:** Implementira autentifikaciju i autorizaciju korisnika koristeći Spring Security.

### FRONTEND

Frontend projekat je razvijen koristeći Angular. Uključuje sledeće glavne komponente:

- **Komponente:** Upravlja delovima korisničkog interfejsa, kao što su formulari i liste.
- **Moduli:** Grupišu srodne komponente i servise za laku ponovnu upotrebu.
- **Navigaciona traka:** Sadrži komponentu za navigaciju kroz aplikaciju.
- **Servisi:** Pružaju logiku za pristup podacima, koristeći HTTP zahteve.
- **Okruženja:** Definišu konfiguracije za različita radna okruženja aplikacije.

## KLJUČNE KOMPONENTE I NJIHOVI OPISI

### ENTITETI

- ❖ **Zaposleni:** Predstavlja zaposlenog, uključujući detalje kao što su ime, odeljenje, mesečni radni sati i plata.
- ❖ **Odeljenje:** Predstavlja odeljenje unutar kompanije.
- ❖ **Odsustvo:** Predstavlja zahtev za odsustvo, uključujući detalje kao što su tip odsustva, trajanje i status odobrenja.



## REPOZITORIJUMI

- ❖ EmployeeRepository: Interfejs za izvođenje CRUD operacija nad entitetima Zaposleni.
- ❖ DepartmentRepository: Interfejs za izvođenje CRUD operacija nad entitetima Odeljenje.
- ❖ LeaveRepository: Interfejs za izvođenje CRUD operacija nad entitetima Odsustvo.

## SERVISI

- ❖ EmployeeService: Sadrži poslovnu logiku za upravljanje zaposlenima.
- ❖ DepartmentService: Sadrži poslovnu logiku za upravljanje odeljenjima.
- ❖ LeaveService: Sadrži poslovnu logiku za upravljanje zahtevima za odsustvo.

## KONTROLERI

- ❖ EmployeeController: Rukovodi HTTP zahtevima vezanim za upravljanje zaposlenima.
- ❖ DepartmentController: Rukovodi HTTP zahtevima vezanim za upravljanje odeljenjima.
- ❖ LeaveController: Rukovodi HTTP zahtevima vezanim za upravljanje odsustvima.

## SIGURNOST

- ❖ SecurityConfig: Konfiguriše sigurnosna podešavanja, uključujući autentifikaciju i autorizaciju.

## BAZA PODATAKA

- ❖ Projekat uključuje SQL datoteku (it355-pz.sql) za postavljanje šeme baze podataka. Definiše neophodne tabele i relacije za čuvanje podataka o zaposlenima, odeljenjima i odsustvima

## TESTIRANJE

Mockito je popularni okvir za testiranje u Java razvoju koji omogućava lako i efikasno pisanje testova sa lažnim objektima (mock objektima). Mock objekti se koriste kako bi se simulirala interakcija sa stvarnim objektima u okruženju testiranja. Mockito pruža bogat set metoda za kreiranje i konfigurisanje mock objekata, kao i metode zadefinisanje očekivanog ponašanja tih objekata tokom testiranja.

```
class LeaveServiceImplTest {
    @Mock // 12 usages
    private LeaveRepository leaveRepository;
    @InjectMocks // 6 usages
    private LeaveServiceImpl leaveService;
    @BeforeEach
    void setUp() { MockitoAnnotations.openMocks(this); }

    @Test
    void findAllLeaves() {
        Leave leave1 = new Leave();
        leave1.setId(1);
        Leave leave2 = new Leave();
        leave2.setId(2);
        List<Leave> leaves = Arrays.asList(leave1, leave2);

        when(leaveRepository.findAll()).thenReturn(leaves);

        List<Leave> result = leaveService.findAllLeaves();
        assertEquals("expected: 2, result.size()",
            2, result.size());
        verify(leaveRepository, times(wantedNumberOfInvocations: 1)).findAll();
    }

    @Test
    void findLeaveById() {
        Leave leave = new Leave();
        leave.setId(1);

        when(leaveRepository.findById(1)).thenReturn(Optional.of(leave));

        Leave result = leaveService.findLeaveById(leaveId: 1);
        assertNotNull(result);
        assertEquals("expected: 1, result.getId()",
            1, result.getId());
        verify(leaveRepository, times(wantedNumberOfInvocations: 1)).findById(1);
    }

    @Test
    void findLeaveById_NotFound() {
        when(leaveRepository.findById(1)).thenReturn(Optional.empty());

        Exception exception = assertThrows(IllegalArgumentException.class, () -> {
            leaveService.findLeaveById(leaveId: 1);
        });

        assertEquals("expected: 'Leave not found', exception.getMessage()",
            "Leave not found", exception.getMessage());
        verify(leaveRepository, times(wantedNumberOfInvocations: 1)).findById(1);
    }
}
```

```
@Test
void saveLeave() {
    Leave leave = new Leave();
    leave.setId(1);

    when(LeaveRepository.save(any(Leave.class))).thenReturn(leave);

    Leave result = leaveService.saveLeave(leave);
    assertNotNull(result);
    assertEquals( expected: 1, result.getId());
    verify(LeaveRepository, times( wantedNumberOfInvocations: 1)).save(leave);
}

@Test
void updateLeave() {
    Leave leave = new Leave();
    leave.setId(1);

    when(LeaveRepository.save(any(Leave.class))).thenReturn(leave);

    Leave result = leaveService.updateLeave(leave);
    assertNotNull(result);
    assertEquals( expected: 1, result.getId());
    verify(LeaveRepository, times( wantedNumberOfInvocations: 1)).save(leave);
}

@Test
void deleteLeaveById() {
    int leaveId = 1;
    doNothing().when(LeaveRepository).deleteById(leaveId);

    leaveService.deleteLeaveById(leaveId);

    verify(LeaveRepository, times( wantedNumberOfInvocations: 1)).deleteById(leaveId);
}
```

## TESTIRANJE HTTP STRANICA

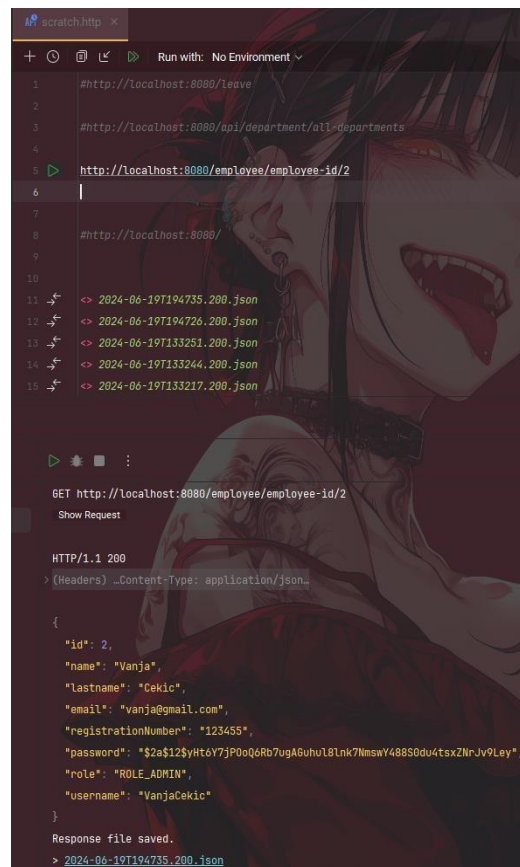
“scratch.http” fajl u IntelliJ IDEA omogućava pravljenje i testiranje HTTP zahteva direktno u IDE-u.

- **Pisanje zahteva:** Unosiš HTTP metode (GET, POST, PUT, DELETE) i URL.
- **Pokretanje zahteva:** Klikom na zeleno dugme pored zahteva izvršavaš ga i vidiš odgovor.
- **Čuvanje odgovora:** Odgovori se automatski čuvaju u JSON fajlovima.
- **Konfiguracija okruženja:** Možeš lako menjati između različitih okruženja (npr. razvojno, produkcijsko).
- **Kolačići i sesije:** IDE čuva kolačiće između zahteva, što je korisno za testiranje autentifikovanih API-ja.

Na primeru iz slike, zahtev je poslat na `http://localhost:8080/employee/employee-id/2` i vraćen je JSON odgovor sa detaljima o zaposlenom.

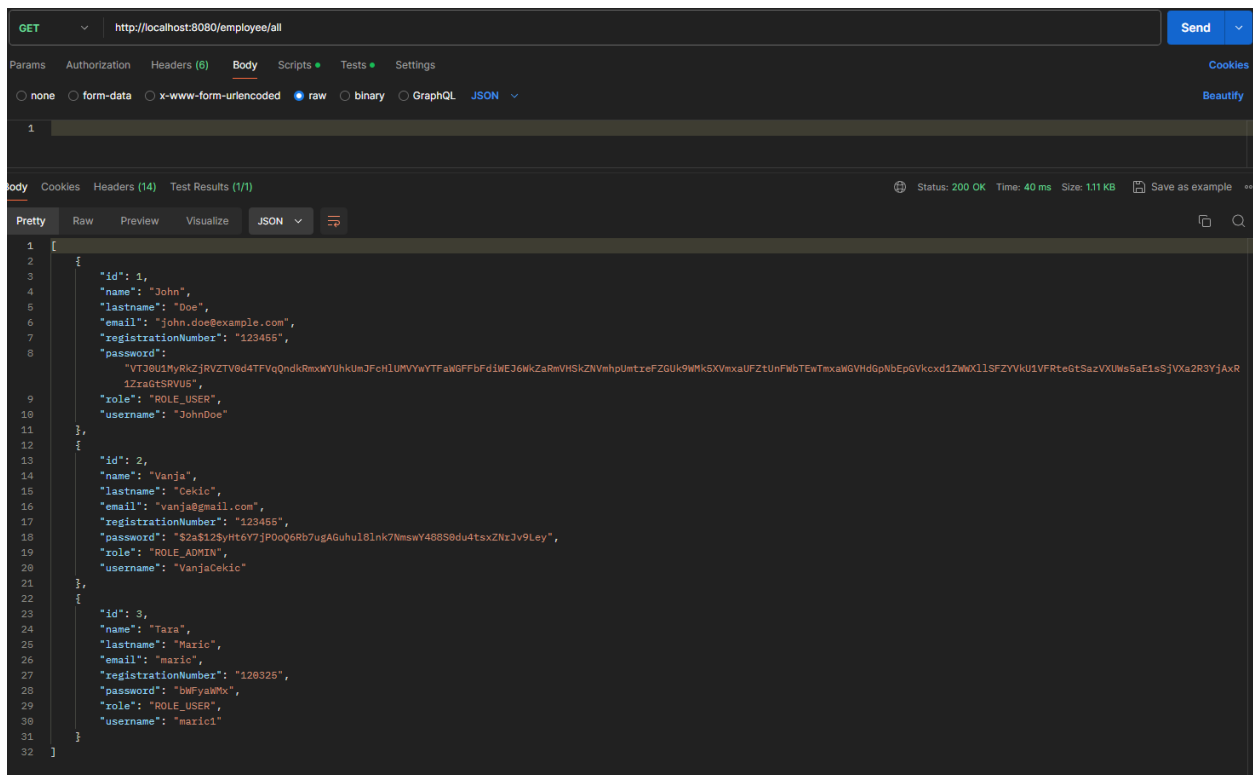
Takođe sam koristila Postman za slanje PUT, POST i DELETE zahteva, što mi omogućava detaljno testiranje i analizu API odgovora u različitim scenarijima.

“scratch.http”:

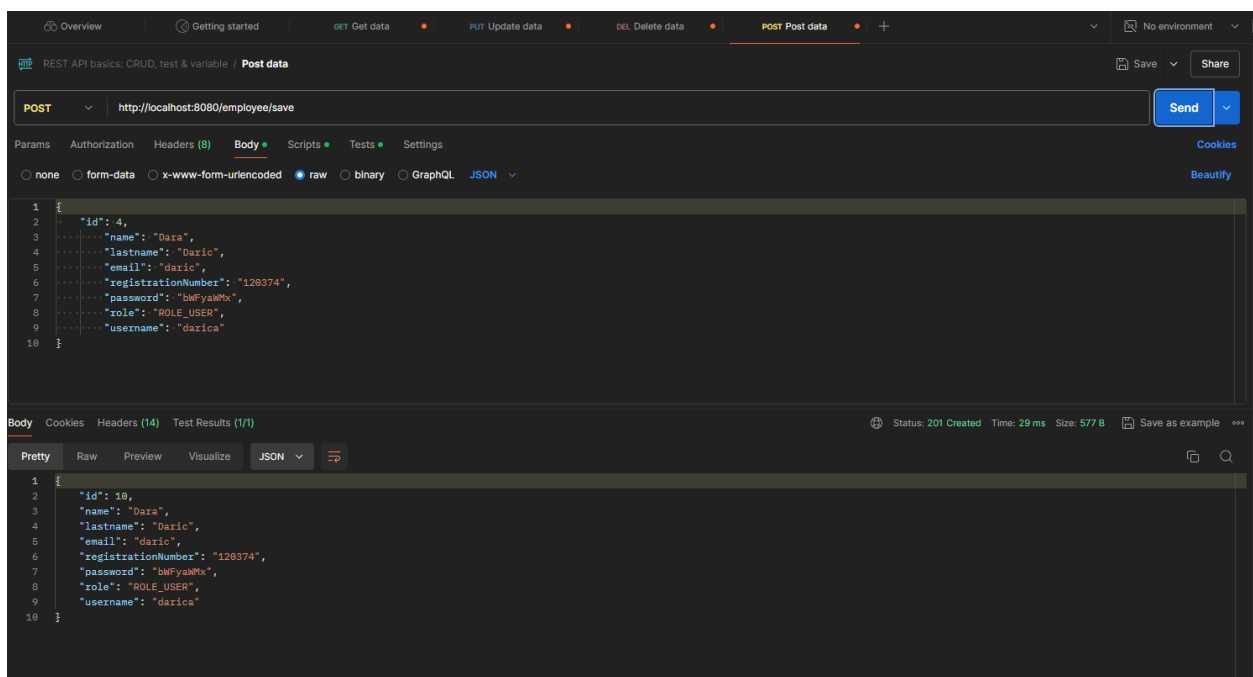


## Postman:

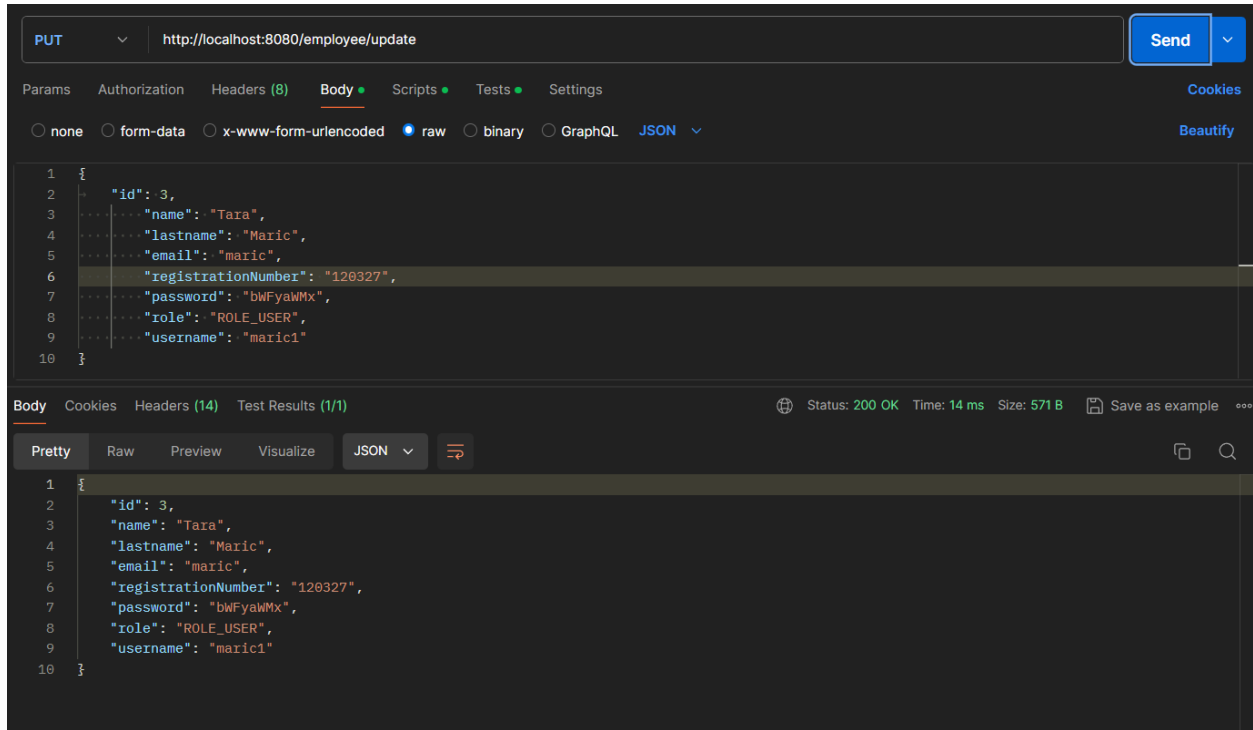
### 1. GET



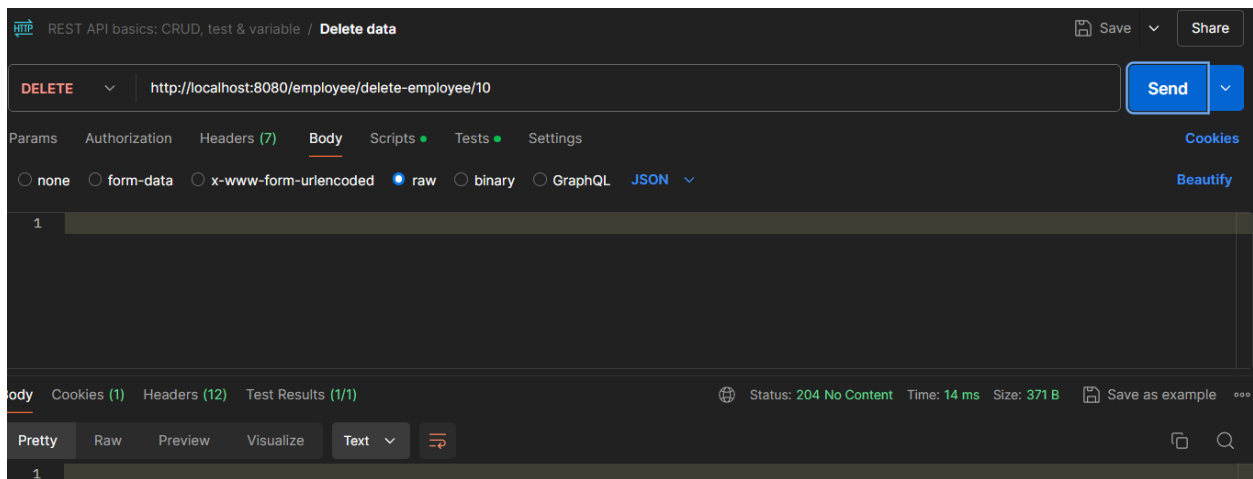
### 2. POST



### 3. PUT

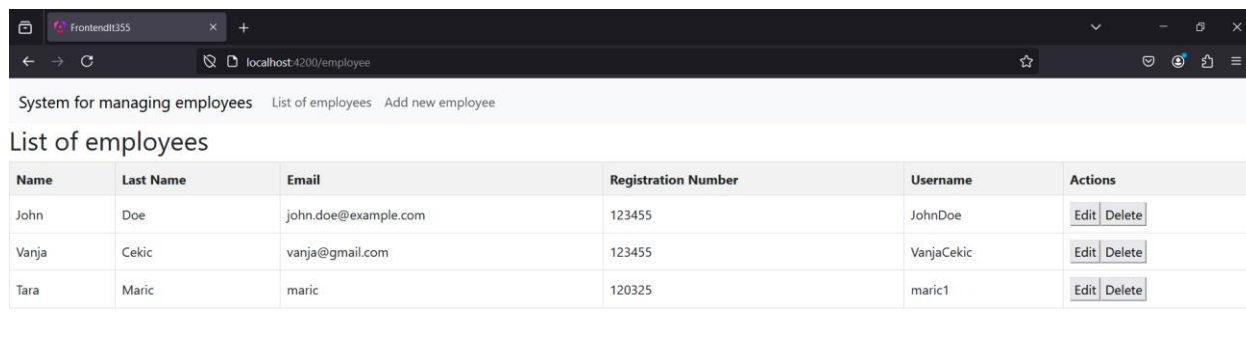


### 4. DELETE



## IMPLEMENTACIJA I PRIKAZ SISTEMA

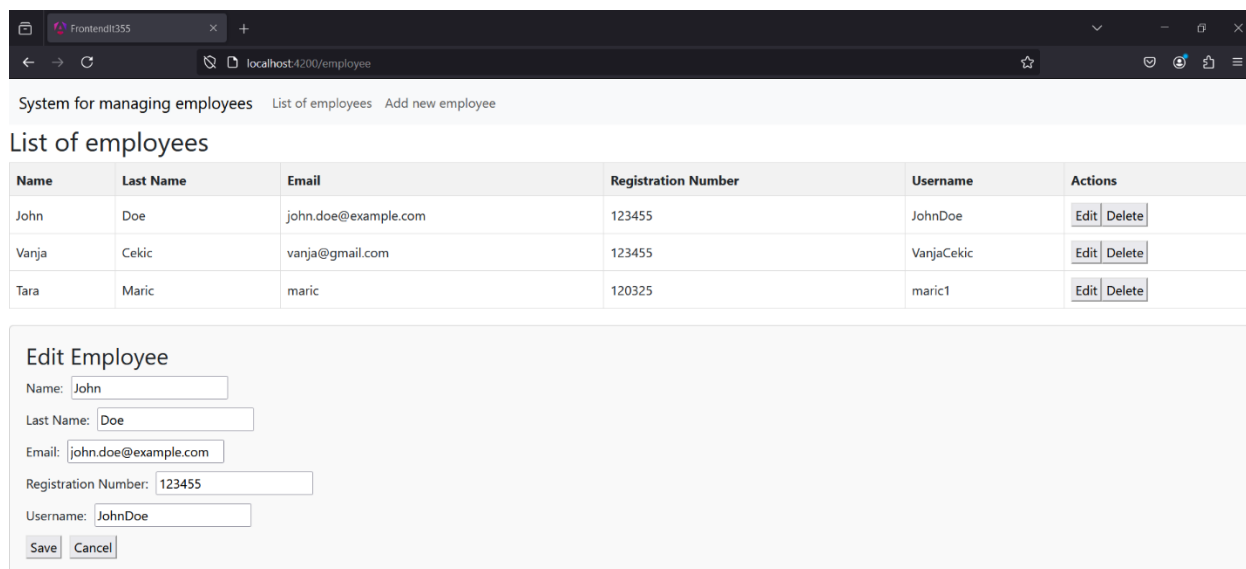
Korisniku je omogućen prikaz tabele svih zaposlenih sa svim svojim podacima.



Name	Last Name	Email	Registration Number	Username	Actions
John	Doe	john.doe@example.com	123455	JohnDoe	<a href="#">Edit</a> <a href="#">Delete</a>
Vanja	Cekic	vanja@gmail.com	123455	VanjaCekic	<a href="#">Edit</a> <a href="#">Delete</a>
Tara	Maric	maric	120325	maric1	<a href="#">Edit</a> <a href="#">Delete</a>

Na dugme “Delete” korisnik briše zaposlenog.

Ukoliko korisnik želi da promeni neki podataka kod zaposlenog, pritiskom na dugme edit prikazaće se polja za izmenu podataka zaposlenog.



Name	Last Name	Email	Registration Number	Username	Actions
John	Doe	john.doe@example.com	123455	JohnDoe	<a href="#">Edit</a> <a href="#">Delete</a>
Vanja	Cekic	vanja@gmail.com	123455	VanjaCekic	<a href="#">Edit</a> <a href="#">Delete</a>
Tara	Maric	maric	120325	maric1	<a href="#">Edit</a> <a href="#">Delete</a>

### Edit Employee

Name:

Last Name:

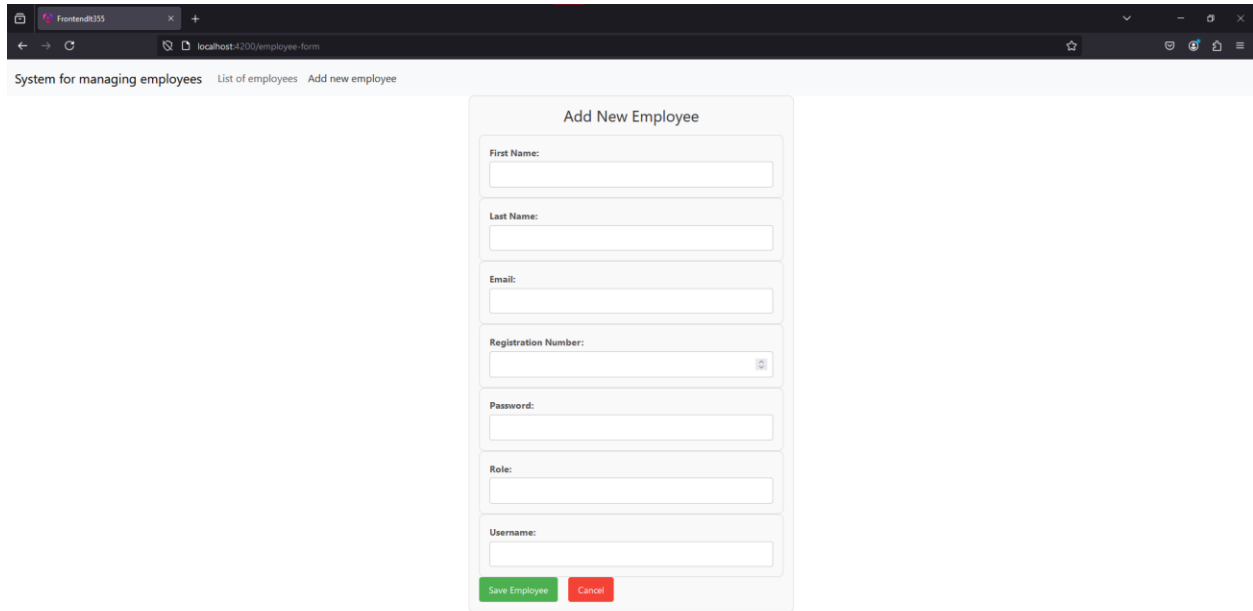
Email:

Registration Number:

Username:

[Save](#) [Cancel](#)

Ako korisnik želi da doda zaposlenog, može putem pritiskom na dugme “Add new employee” koji će da odvede korisnika na novu stranicu za dodavanje zaposlenog. Pretiskom dugme “Cancel” odkazaće dodavanje zaposlenog.



The screenshot shows a web browser window with the address bar displaying 'localhost:4200/employee-form'. The browser's title bar indicates 'Frontend0355'. The page content includes a navigation bar with the text 'System for managing employees' and two links: 'List of employees' and 'Add new employee'. The main content area features a form titled 'Add New Employee'. This form contains several input fields: 'First Name:', 'Last Name:', 'Email:', 'Registration Number:' (with a small calendar icon on the right), 'Password:', 'Role:', and 'Username:'. At the bottom of the form, there are two buttons: a green 'Save Employee' button and a red 'Cancel' button.



## BUDUĆA POBOLJŠANJA

**Kompletan Razvoj Frontend-a:** Neophodno je razviti Angular frontend i integrisati ga sa backend servisima kako bi se postigla potpuna funkcionalnost aplikacije.

**Napredni Izveštaji:** Implementacija naprednih funkcionalnosti za izveštavanje je potrebna za bolju vizualizaciju i analizu podataka, čime bi se unapredilo donošenje odluka

**Kontrola Pristupa na Osnovu Uloga:** Poboljšanje sigurnosnog modula kako bi podržao detaljniju kontrolu pristupa na osnovu uloga, što će povećati sigurnost sistema i prilagoditi ga specifičnim potrebama korisnika.

**Sistem Obaveštenja:** Dodavanje sistema obaveštenja koji će korisnike informisati o važnim događajima, kao što su odobrenja odsustva, čime bi se poboljšala korisnička interakcija i efikasnost komunikacije unutar sistema.

## ZAKLJUČAK

Sistem za Upravljanje Zaposlenima je robustno rešenje dizajnirano da pojednostavi upravljanje zaposlenima unutar organizacije. Dok je projekat trenutno fokusiran na backend, buduća poboljšanja i integracija frontenda dodatno će unaprediti njegovu funkcionalnost i korisničko iskustvo.

## LITERATURA

1. <https://start.spring.io> – Kreiranje projekta
2. <https://www.youtube.com/watch?v=HYGnVeCs0Yg&t=453s> - Spring Boot Thymeleaf Web Application | FullCourse
3. <https://spring.io/projects/spring-security> - Spring Security
4. <http://lams.metropolitan.ac.rs:8080/lams/> - Lams lekcije predmeta IT355 – Web 2
5. <https://spring.io/guides/gs/securing-web> - Security-web
6. [https://medium.com/@Lakshitha\\_Fernando/spring-security-6-and-spring-boot-3-with-simple-project-91389cc13119](https://medium.com/@Lakshitha_Fernando/spring-security-6-and-spring-boot-3-with-simple-project-91389cc13119) - Spring Security 6 and Spring Boot 3 with Simple Project
7. Stackoverflow
8. <https://getbootstrap.com/docs/5.3/components/navbar/#how-it-works> - Bootstrap