



Prolećni semestar, 2021/22

PREDMET:

IT355 - Web Sistemi 2

Aplikacija za upravljanje kursevima

Projektni zadatak

Asistent: **Jovana Jović**
Studenti: **Aleksa Cekić, Stefan Gogić**
Indeks: **4173, 4056**

Niš, 2022

Opis sistema	3
Tehničke specifikacije	3
Specifikacije zahteva	4
3.1 Funkcionalni zahtevi	4
3.1.1 Pristup sistemu (Login)	4
3.1.2 Registracija na sistem	4
3.1.3 Administrator	4
3.1.4 Autor	4
3.1.5 Korisnik	5
3.2 Nefunkcionalni zahtevi	5
3.2.1 Performanse	5
3.2.2 Bezbednost	6
3.2.3 Sigurnost	6
3.2.4 Integritet	6
3.2.5 Promenljivost	6
3.2.6 Ponovna upotrebljivost	6
Slučaj korišćenja	7
Dizajn sistema	7
5.1 Klasni dijagram	7
5.2 Arhitektura sistema	8
Dijagram sekvenci	8
6.1 Dijagram sekvenci dodavanje korisnika	9
6.1 Dijagram sekvenci kupovina kurseva	9
Intefejs aplikacije	9
7.1 Servisi	9
7.1 Korisnički interfejs	18
Tesitranje	23

Zaključak	27
Literatura	28

1. Opis sistema

Sistem koji se razvija služi kao aplikacija za upravljanje i kupovinu kurseva. Korisnici se dele na administratore, autore i korisnike. Administrator može da dodaje nove korisnike i autore u sistemu, upravlja komentarima, kao i da odbije ili da prihvati refund ukoliko neki korisnik zahteva refund. Autor ima ulogu postavke kurseva na sistemu koji korisnici mogu kupiti dok korisnik može da kupuje kurseve i da ostavi komentare na nekom kursu kao i da zahteva refund ukoliko nije zadovoljan kursom koji je kupio.

2. Tehničke specifikacije

Korisniku je potreban najobičniji računar neke normalne snage, sa minimalnim zahtevima kao što su:

CPU: 1 gigahertz (GHz) ili jače sa dva ili više jezgra na kompatibilnom 64-bitnom procesoru.

RAM: 4 gigabytes (GB) ili jače

GPU: bilo koja kompatibilna sa DirectX 12 ili kasnije

Internet konekcija: Internet konekcija je neophodna da bi se pristupilo sistemu s obzirom na to da je ovo web aplikacija. Koristiti neki od najnovijih verzija pretraživača (Google, Safari, Opera, Mozilla)

Tehnologije koje su korišćene prilikom izrade aplikacije su sledeće:

Spring Boot,

Thymeleaf,

MySQL,

BootstrapCSS

3. Specifikacije zahteva

3.1 Funkcionalni zahtevi

U ovom dokumentu funkcionalne zahteve označićemo sa REQ-Br. Funkcionalni zahtevi se pišu iz ugla korisnika.

3.1.1 Pristup sistemu (Login)

REQ-1: Svaki korisnik mora da se preko svojih kredencijala uloguje na sistem.

REQ-2: Korisnik unosi korisničko ime i lozinku prilikom logovanja

3.1.2 Registracija na sistem

REQ-1: Korisnik se registruje na sistem ukoliko nema nalog

REQ-2: Korisnik unosi korisničko ime, email i lozinku prilikom registracije

3.1.3 Administrator

REQ-1: Administratoru treba omogućiti pregled svih korisnika u sistemu.

REQ-2: Administratoru treba omogućiti unos podataka koji se traže za dodavanje novog korisnika.

Administrator unosi korisničko ime, lozinku, email i rolu.

REQ-3: Administratoru treba omogućiti izmenu podataka koji se traže pri izmeni korisnika.

Administrator unosi korisničko ime, lozinku, email i rolu.

REQ-4: Administratoru treba omogućiti brisanje korisnika.

REQ-5: Administratoru treba omogućiti odobrenje ili odbijanje vraćanje novca

REQ-6: Administratoru treba omogućiti brisanje komentara.

3.1.4 Autor

REQ-1: Autoru treba omogućiti pregled svih kurseva u sistemu.

REQ-2: Autoru treba omogućiti unos podataka koji se traže za dodavanje novog kursa.

Autor unosi naziv kursa, opis kursa, cena i kategoriju.

REQ-3: Administratoru treba omogućiti izmenu podataka koji se traže pri izmeni kursa.

Autor unosi naziv kursa, opis kursa, cena i kategoriju.

REQ-4: Autoru treba omogućiti brisanje kurseva.

3.1.5 Korisnik

REQ-1: Korisniku treba omogućiti pregled svih kurseva sa sajta.

REQ-2: Korisniku treba omogućiti kupovinu kursa sa sajta.

REQ-3: Korisniku treba omogućiti pregled detalja izabranog kursa.

REQ-4: Korisniku treba omogućiti kupovinu kursa

REQ-5: Korisniku treba omogućiti pregled svih kupljenih kurseva

REQ-6: Korisniku treba omogućiti da traži povraćaj novca ukoliko traži refund i ukoliko je njegov razlog validan.

3.2 Nefunkcionalni zahtevi

Nazivi nefunkcionalnih zahteva biće označeno prefiksima:

PER - Performanse

SAF - Bezbednost

SEC - Sigurnost

INT - Integritet

MOD - Promenljivost

3.2.1 Performanse

PER-1: Vreme potrebno da se korisniku vrati lista svih podataka neće trajati duže od 2.0 sekunde, ako je internet veza stabilna.

PER-2: Čekanje od klika za kupovinu kursa do izvršenje transakcije ne sme da bude više od 10s, ako je internet veza stabilna.

3.2.2 Bezbednost

SAF-1: Privatne informacije o korisnicima ne smeju biti prikazane javno, lozinke korisnika uvek moraju biti enkriptovane I ne smeju da se prikazuju.

SAF-2: Ukoliko je korisnik neaktivan više od 10 minuta, sistem će ga automatski izlogovati I naterati da se uloguje ponovo.

3.2.3 Sigurnost

SEC-1: Sistem će zaključati korisnički nalog nakon pet uzastopna neuspešna pokušaja prijave u roku od 5 minuta.

3.2.4 Integritet

INT-1: Sistem vrši sigurnosne kopije podataka prilikom kupovine kursa.

INT-2: Sistem se štiti od neovlašćenog dodavanja, brisanja ili modifikacije podataka.

INT-3: Svako čuvanje novog stanja će raditi duplu proveru, što će povećati sigurnost pri čuvanju podatka.

3.2.5 Promenljivost

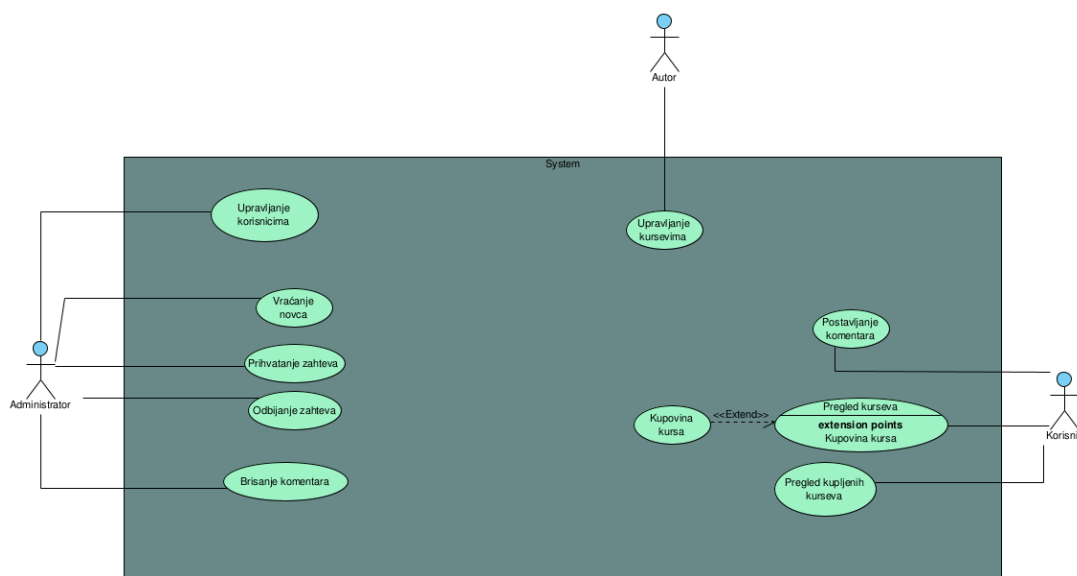
MOD-1: Provera validnosti biće znatno umanjena zbog toga što će za svaki deo koda biznis logike biti pisan odgovarajući test.

3.2.6 Ponovna upotrebljivost

REU-1: Aplikacija će koristiti komponente kao što su tabele, status barove, kao i bootstrap stilove, koje mogu da se ponovo koriste u aplikaciji.

REU-2: Kod će biti pisan uz praćenje OOp pristupa razvoja što će olakšati reuzibilnost.

4. Slučaj korišćenja

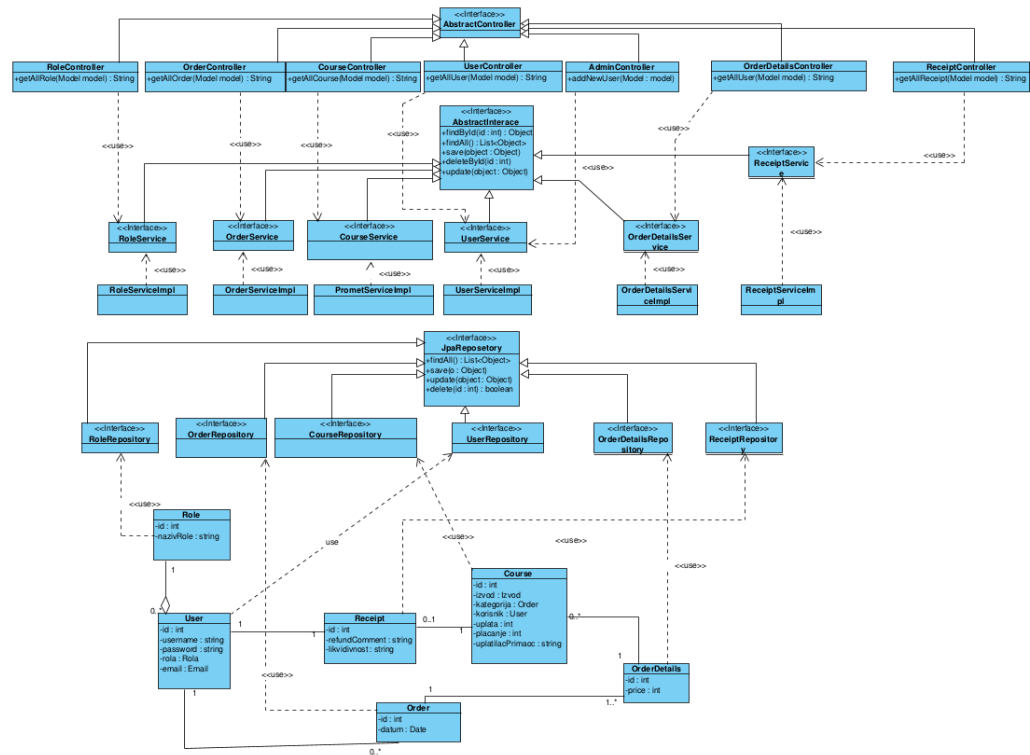


Slika 1. - prikaz slučajeva korišćenja

5. Dizajn sistema

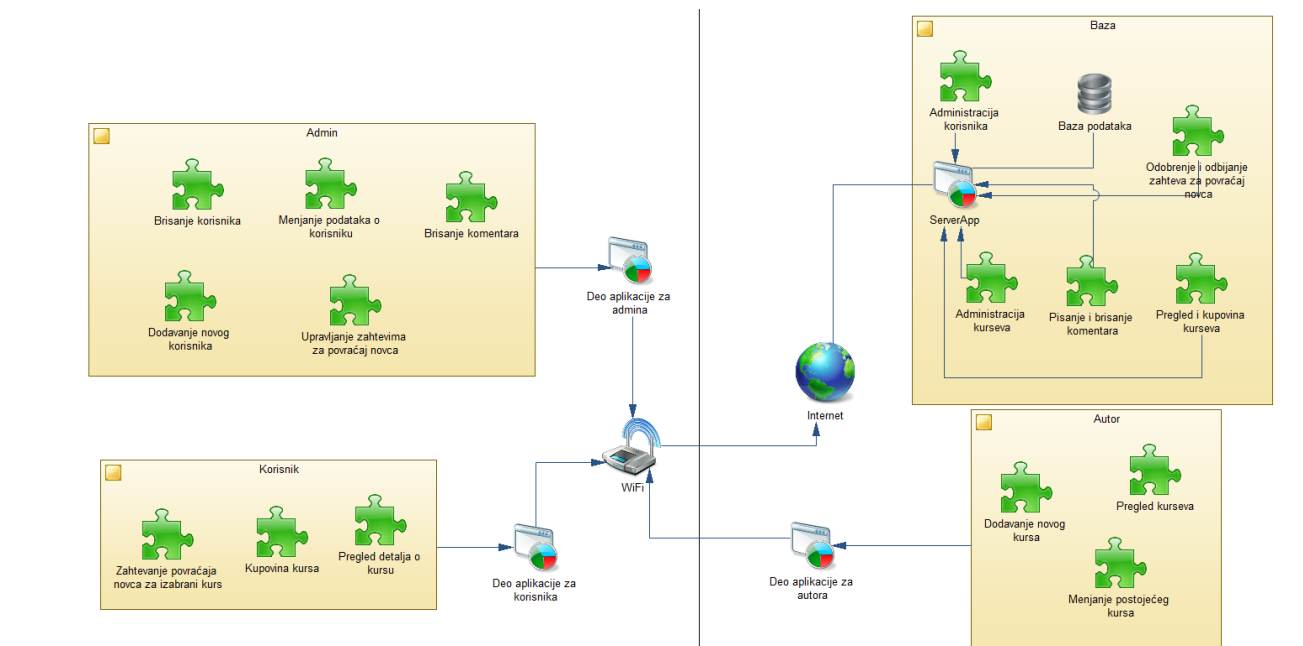
U sledećem delu vidimo klasni dijagram koji zapravo predstavlja servisno-orientisanu arhitekturu sistema koja se može podeliti na kontrolere, servise (i implementacije servisa) i entitete kako bi se omogućila potpuna funkcionalnost. Svaka klasa predstavlja neki deo sistema bio to kontroler ili entitet i ima svoju ulogu. Grafički tj. objektni jezik za modelovanje je UML (Unified Modeling Language) koji služi da se preko grafičkih simbola pravi apstraktni model sistema poznat kao UML model. Na ovoj slici možemo videti kako ove klase komuniciraju međusobno i možemo primetiti veze između njih. Postoje JpaController i JpaRepository interfejsi koji predstavljaju kako u real-time funkcionišu arhitektura Spring Boot aplikacije. Pored klasnog dijagrama imamo i pregled arhitekture sistema, u ovom slučaju primenjujemo arhitekturu klijent-server.

5.1 Klasni dijagram



Slika 2. - prikaz klasnog dijagrama

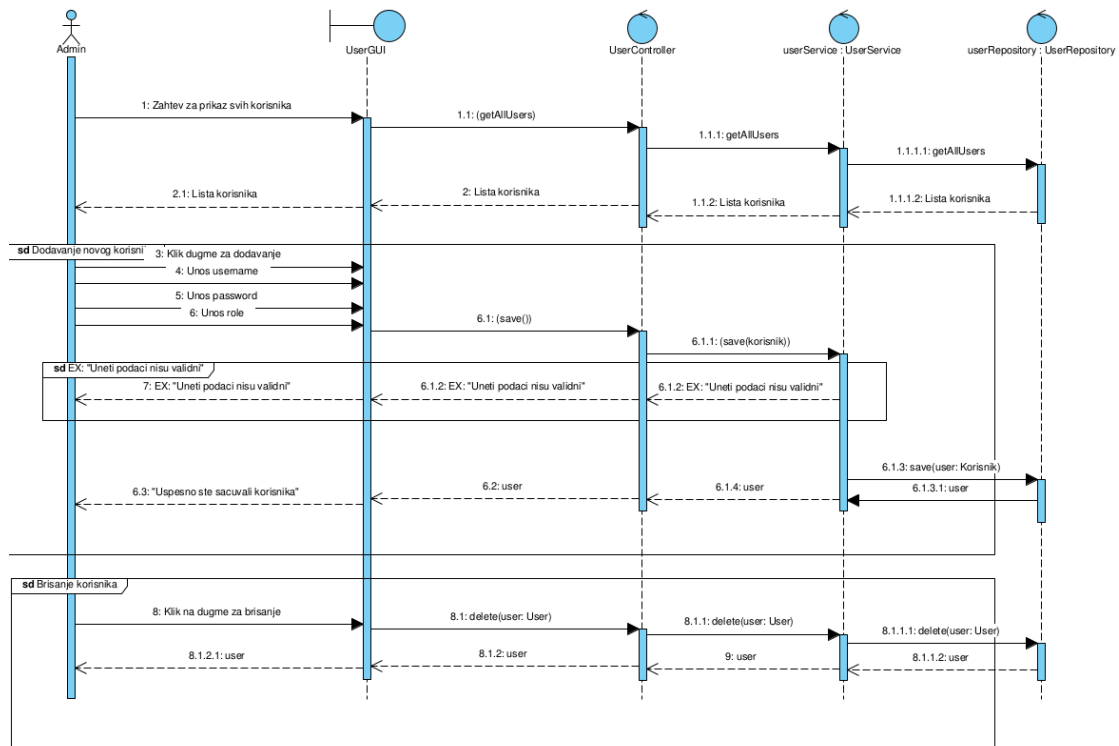
5.2 Arhitektura sistema



Slika 3. - prikaz klijent-server arhitekture sistema

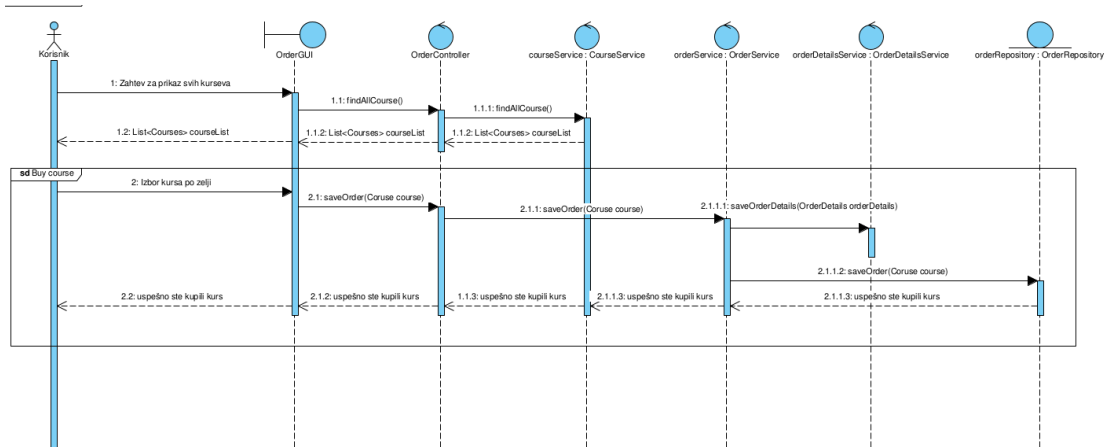
6. Dijagram sekvenci

6.1 Dijagram sekvenci dodavanje korisnika



Slika 4 - prikaz sekvencijalnog dijagrama za dodavanje novih korisnika

6.1 Dijagram sekvenci kupovina kurseva



Slika 5 - prikaz sekvencijalnog dijagrama za kupovinu kurseva

7. Intefejs aplikacije

7.1 Servisi

```
public interface CategoryService {  
  
    1 usage 1 implementation stefan-gg  
    List<Category> getAllCategories();  
  
    1 implementation stefan-gg  
    Category getCategoryById(Integer categoryId);  
  
    1 implementation stefan-gg  
    Category saveCategory(Category category);  
  
    1 implementation stefan-gg  
    Category updateCategory(Category category);  
  
    1 implementation stefan-gg  
    void deleteById(Integer categoryId);  
  
}
```

Slika 6. - CategoryService

```
public interface CommentService {  
  
    1 implementation  👤 Aleksa Cekic  
    List<Comment> getAllComments();  
  
    1 implementation  👤 Aleksa Cekic  
    Comment getCommentById(Integer commentId);  
  
    1 usage  1 implementation  👤 Aleksa Cekic  
    Comment saveComment(Comment comment);  
  
    1 implementation  👤 Aleksa Cekic  
    Comment updateComment(Comment comment);  
  
    1 implementation  👤 Aleksa Cekic  
    void deleteById(Integer commentId);  
  
    2 usages  1 implementation  👤 stefan-gg  
    List<Comment> findAllByCourseId(Integer id);  
}
```

Slika 7. - CommentService

```
public interface CourseService {  
    3 usages 1 implementation 👤 Aleksa Cekic  
    List<Course> getAllCourses();  
  
    2 usages 1 implementation 👤 Aleksa Cekic  
    Course getById(Integer courseId);  
  
    1 usage 1 implementation 👤 Aleksa Cekic  
    Course saveCourse(Course course);  
  
    1 implementation 👤 Aleksa Cekic  
    Course updateCourse(Course course);  
  
    1 implementation 👤 Aleksa Cekic  
    void deleteById(Integer courseId);  
}
```

Slika 8. - CourseService

```
public interface OrderDetailsService {  
  
    List<OrderDetails> getAllOrderDetails();  
  
    OrderDetails getOrderDetailById(Integer orderDetailId);  
  
    OrderDetails saveOrderDetail(OrderDetails orderDetails);  
  
    OrderDetails updateOrderDetail(OrderDetails orderDetails);  
  
    void deleteById(Integer orderDetailId);  
  
    List<OrderDetails> getAllByUserId(Integer userId);  
}
```

Slika 9. - OrderDetailsService

```
public interface OrderService {  
    1 implementation  👤 stefan-gg  
    List<Order> getAllOrders();  
  
    1 implementation  👤 stefan-gg  
    Order getOrderById(Integer orderId);  
  
    1 usage  1 implementation  👤 stefan-gg  
    Order saveOrder(Order order);  
  
    1 implementation  👤 stefan-gg  
    Order updateOrder(Order order);  
  
    1 implementation  👤 stefan-gg  
    void deleteById(Integer orderId);  
}
```

Slika 10. - OrderService

```
public interface ReceiptService {  
  
    1 usage 1 implementation stefan-gg  
    List<Receipt> getAllReceipts();  
  
    1 implementation stefan-gg  
    Receipt getReceiptById(Integer receiptId);  
  
    1 usage 1 implementation stefan-gg  
    Receipt saveReceipt(Receipt receipt);  
  
    1 implementation stefan-gg  
    Receipt updateReceipt(Receipt receipt);  
  
    1 implementation stefan-gg  
    void deleteById(Integer receiptId);  
}
```

Slika 11. - ReceiptService

```
public interface RefundService {  
    1 usage 1 implementation 👤 Aleksa Cekic  
    List<Refund> getAllRefunds();  
  
    1 implementation 👤 Aleksa Cekic  
    Refund getRefundById(Integer refundId);  
  
    1 usage 1 implementation 👤 Aleksa Cekic  
    Refund saveRefund(Refund refund);  
  
    1 implementation 👤 Aleksa Cekic  
    Refund updateRefund(Refund refund);  
  
    1 implementation 👤 Aleksa Cekic  
    void deleteById(Integer refundId);  
}
```

Slika 12. - RefunService


```
public interface RoleService {  
    4 usages 1 implementation 👤 Aleksa Cekic  
    List<Role> getAllRoles();  
  
    1 usage 1 implementation 👤 Aleksa Cekic  
    Role getRoleById(Integer roleId);  
  
    1 implementation 👤 Aleksa Cekic  
    Role saveRole(Role role);  
  
    1 implementation 👤 Aleksa Cekic  
    Role updateRole(Role role);  
  
    1 implementation 👤 Aleksa Cekic  
    void deleteById(Integer roleId);  
}
```

Slika 13. - RoleService

```

public interface UserService {
    3 usages 1 implementation 👤 Aleksa Cekic
    List<User> getAllUsers();

    1 usage 1 implementation 👤 Aleksa Cekic
    User getUserById(Integer userId);

    1 implementation 👤 Aleksa Cekic
    User getUserByUsernameAndPassword(String username, String password);

    5 usages 1 implementation 👤 stefan-gg
    User getUserByUsername(String username);

    1 implementation 👤 Aleksa Cekic
    User getUserByEmail(String email);

    4 usages 1 implementation 👤 Aleksa Cekic
    User saveUser(User user);

    1 usage 1 implementation 👤 Aleksa Cekic
    User updateUser(User user);

    1 implementation 👤 Aleksa Cekic
    void deleteById(Integer userId);
}

```

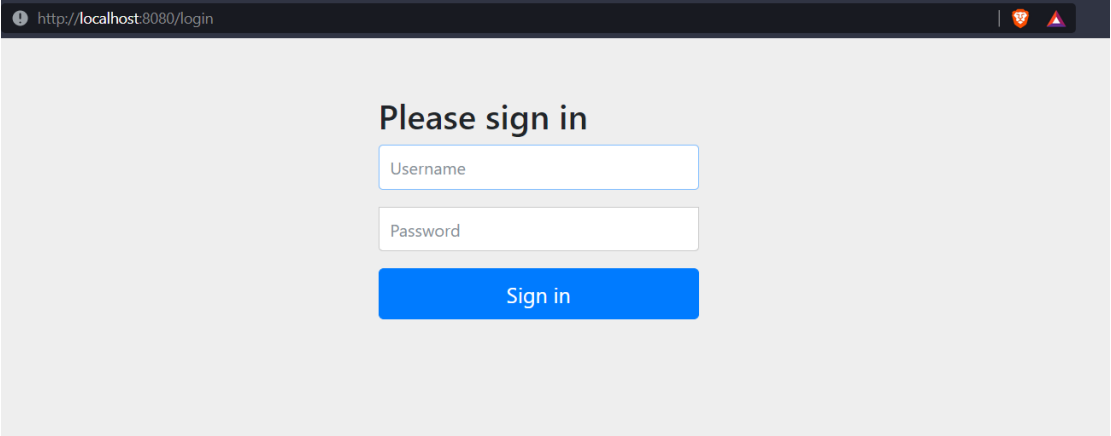
Slika 14. - UserService

Na slikama (od 8. do 13. slike) se vidi prikaz servisa koji sadrže osnovne operacija ažuriranja, getById, deleteById, getAll, čuvanja novog objekta u bazi, kao i još neke dodatne operacije ukoliko je bilo potrebe za tim. Na slici 6. se može videti metoda findAllByCourseId koja je morala da se naknadno doda u repozitorijum. Na slici 8. se može videti metoda getAllByUserId koja je isto dodata u repozitorijum. Na slici 13. se mogu videti dodatne metode getUserByEmail i getUserByUsernameAndPassword koje su takođe dodate u repozitorijum.

7.1 Korisnički interfejs

Dizajn korisničkog interfejsa je rađen korišćenjem Bootstrap-a i CSS-a.

U daljem delu predstavimo korisnički interfejs aplikacije.



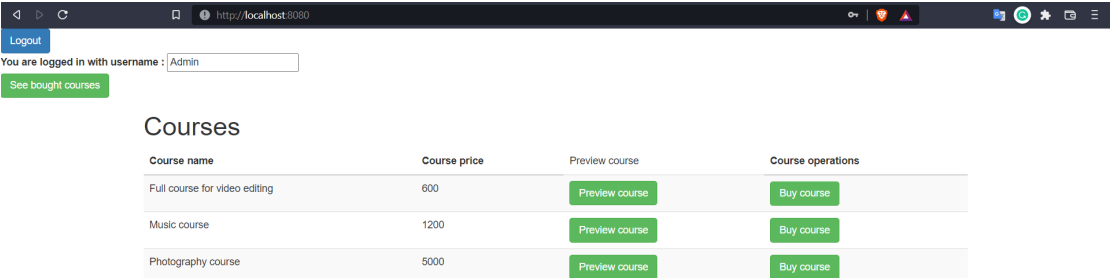
A screenshot of a web browser showing a login page. The address bar displays 'http://localhost:8080/login'. The page has a light gray background. In the center, there is a form titled 'Please sign in'. It contains two input fields: 'Username' and 'Password', both with light blue borders. Below these fields is a blue button with the text 'Sign in' in white.

Slika 15. - prikaz login forme



A screenshot of a web browser showing a registration page. The page has a light gray background. At the top, there is a blue header bar with the text 'Register' in white. Below the header, there is a form with four input fields: 'Username', 'Email', 'Password', and 'Role'. The 'Role' field has a dropdown menu with 'ROLE_USER' selected. At the bottom of the form is a blue button with the text 'Submit' in white.

Slika 16. - prikaz register forme



A screenshot of a web browser showing a user dashboard. The address bar displays 'http://localhost:8080'. At the top left, there is a 'Logout' button. Below it, a message says 'You are logged in with username : Admin' next to a text input field. Below that is a green button labeled 'See bought courses'. The main content area is titled 'Courses' and contains a table with the following data:

Course name	Course price	Preview course	Course operations
Full course for video editing	600	Preview course	Buy course
Music course	1200	Preview course	Buy course
Photography course	5000	Preview course	Buy course

Slika 17. - prikaz početne stranice za običnog korisnika

Logout Go back

Category

Video editing

Course name

Full course for video editing

Course description

Best video editing course. Nice

Price

600

Buy course

Add your comment :

Course name

Add comment

Stefan't

Best courseeee

Admin

Nice course

Slika 18. - prikaz detalja o kursu

Logout

Courses

Course name	Course description	Preview course	Course operations
Photography course	Photography course!	Refund reason	Request a refund
Photography course	Photography course!	Refund reason	Request a refund
Photography course	Photography course!	Refund reason	Request a refund
Photography course	Photography course!	Refund reason	Request a refund

Slika 19. - prikaz svih kupljenih kurseva

Logout

Date	Price	Category Name	Course Name	Course Description
2022-06-20	600	Video editing	Full course for video editing	Best video editing course. Nice

Buy

Slika 20. - kupovina kursa

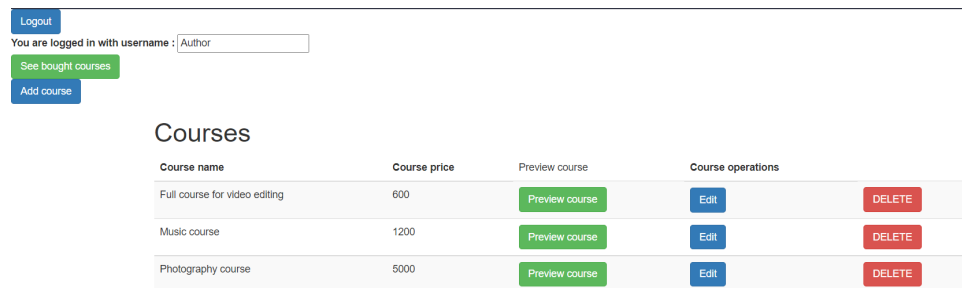
Your purchase is successful !

Date 2022-06-20 Price 600

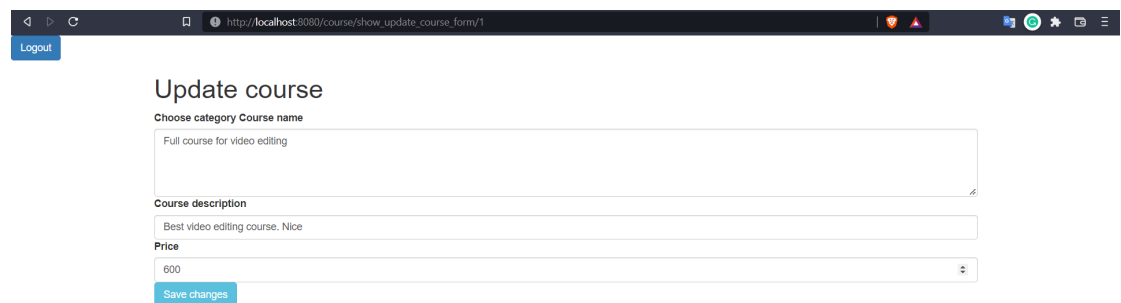
Slika 11. - prikaz poruke nakon što je pritisnuto dugme Buy na slici br. 10.

Course name	Course description	Preview course	Course operations
Photography course	Photography course!	I dont want this course anymore!	Request a refund

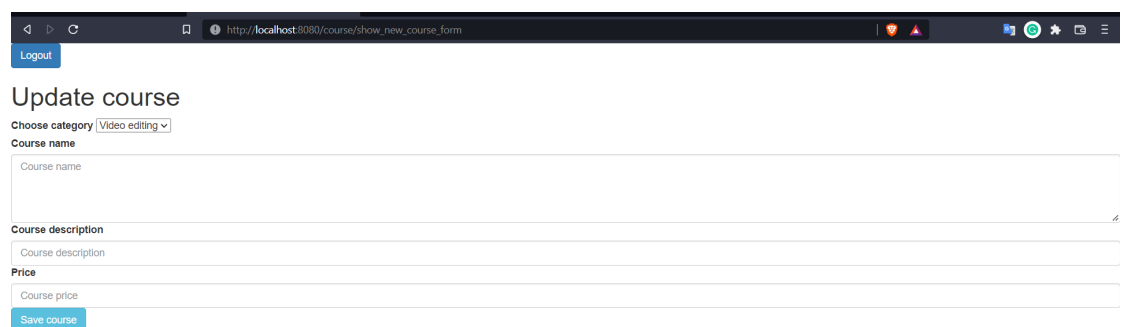
Slika 21. - popunjavanje forme za povraćaj novca
Niš, 2022



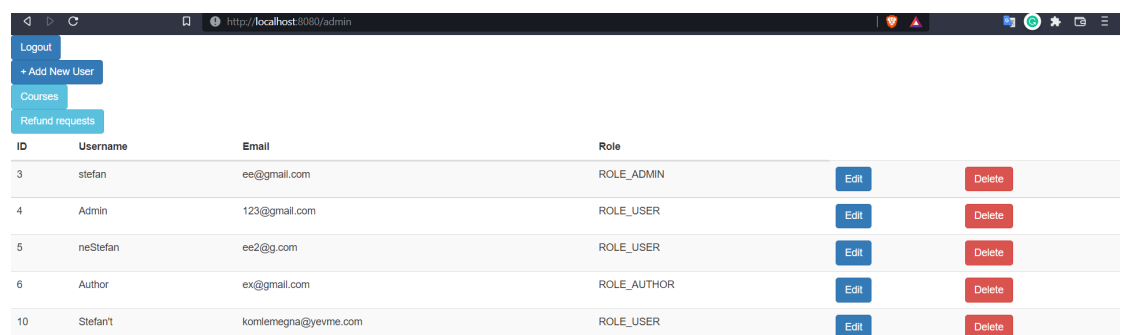
Slika 22. - prikaz početne stranice za autora



Slika 23. - prikaz stranice za ažuriranje postojećeg kursa



Slika 24. - prikaz stranice za dodavanje novog kursa



Slika 25. - prikaz početne stranice za admina

Logout

Edit user

stefan Password ee@gmail.com ROLE_ADMIN Save

Slika 17. - prikaz stranice za ažuriranje korisnika

Logout

Save user

User name Password Email ROLE_USER Save

Slika 26. - prikaz stranice za kreiranje novog korisnika

Logout

Course name	Course operations
Full course for video editing	See comments for this course
Music course	See comments for this course
Photography course	See comments for this course

Slika 27. - prikaz stranice sa kursevima

Logout

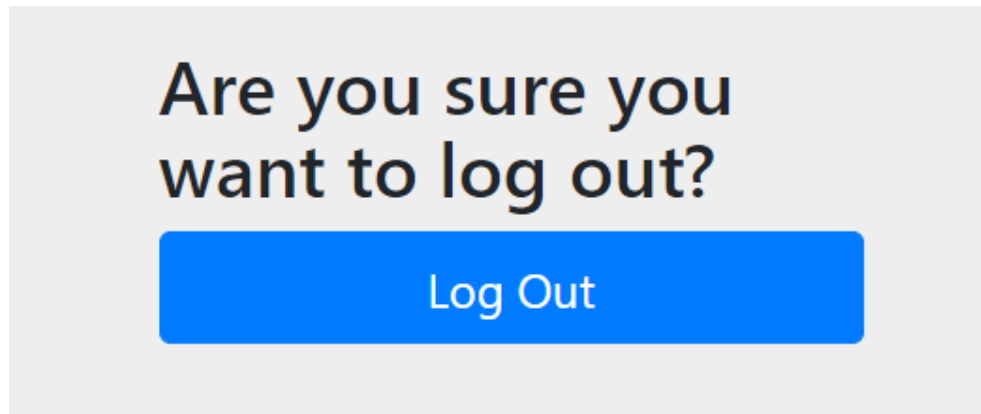
User	Comment	Operatins
Stefan't	Best courseee	Delete comment
Admin	Nice course	Delete comment

Slika 28. - prikaz komentara za izabrani kurs

Logout

User username	Refund comment	Course name	Refund operations
Admin	I dont want this course anymore!	Photography course	Decline refund Accept refund

Slika 29. - prikaz stranice sa svim zahtevima za povraćaj novca



Slika 30. - prikaz stranice za logout

8. Tesitranje

Što se tiče testiranja softvera, koristili smo javinu JUnit 5 biblioteku radi jediničnog testiranja aplikacije. Pored javine JUnit 5 iskoristili smo takođe Mockito biblioteku koja nam pomaže u testiranju RESTful servisa. U ovom slučaju, istestirali smo jedinke user dela aplikacije.

```
@ExtendWith(MockitoExtension.class)
@SpringBootTest
class UserServiceImplTest {
    @MockBean
    private UserRepository userRepository;
    @Autowired
    private UserService userService;
    @Autowired
    private RoleService roleService;

    @Test
    void getAllUsersTest() {
        List<User> userList = userService.getAllUsers();
        when(userRepository.findAll()).thenReturn(userList);
        Integer expectedCount = userList.size();
        assertEquals(expectedCount, userRepository.findAll().size());
    }

    @Test
    void saveUserTest() {
        User user = new User();
        Role role = new Role();
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        role.setName("ROLE_TEST");
        user.setUsername("Test");
        user.setPassword(passwordEncoder.encode("test"));
        user.setEmail("test@test.com");
        user.setRole(role);

        when(userRepository.save(user)).thenReturn(user);
        System.out.println(user);
        assertEquals(user, userService.saveUser(user));
    }

    @Test
    void updateUserTest() {
        User user = new User();
        Role role = new Role();
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
    }
```

Slika 31. - prikaz koda iz test klase (1. deo)


```
@Test
void updateUserTest() {
    User user = new User();
    Role role = new Role();
    BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
    role.setName("ROLE_TEST");
    user.setUsername("Updated Test");
    user.setPassword(passwordEncoder.encode("test"));
    user.setEmail("test@test.com");
    user.setRole(role);
    when(userRepository.save(user)).thenReturn(user);

    User actualUser = userService.saveUser(user);
    System.out.println(user);
    assertEquals(user, actualUser);
    assertEquals(user.getUsername(), actualUser.getUsername());
    assertEquals(user.getPassword(), actualUser.getPassword());
    assertEquals(user.getRole().getName(), actualUser.getRole().getName());
}

@Test
void getUserByUsernameTest() {
    String userName = "stefan";

    User user = new User();
    Role role = new Role();
    role.setId(1);
    role.setName("ROLE_ADMIN");
    user.setId(13);
    user.setUsername("stefan");
    user.setPassword("$2a$12$LP/ahNreRJ9LGKENYmHvwu2CdHrSGEI1v2i5E49kddvuSWnQCvxTy");
    user.setEmail("stefan@gmail.com");
    user.setRole(role);

    when(userRepository.findByUsername(userName)).thenReturn(user);

    User actualUser = userRepository.findByUsername(userName);

    assertEquals(user, actualUser);
    assertEquals(userName, actualUser.getUsername());
}
```

Slika 32. - prikaz koda iz test klase (2. deo)

```
@Test
void getUserByEmailTest() {
    String email = "user@gmail.com";

    User user = new User();
    Role role = roleService.getRoleById( roleId: 3);
    user.setId(15);
    user.setUsername("user");
    user.setPassword("$2a$12$r-fybhogfgRtBqj/63WksIezoUwHY1a1iWIS5.QDm8k6CbRM1bCSwC");
    user.setEmail("user@gmail.com");
    user.setRole(role);

    when(userRepository.findById(email)).thenReturn(user);

    User actualUser = userService.getUserByEmail(email);

    assertEquals(user, actualUser);
    assertEquals(email, actualUser.getEmail());
}

@Test
void getUserByIdTest() {
    Integer id = 14;
    User user = new User();
    Role role = new Role();
    role.setId(1);
    role.setName("ROLE_ADMIN");
    user.setId(14);
    user.setUsername("aleksa");
    user.setPassword("$2a$12$ZdZ046FRbw0r4cv86vUte0f8e0H0ZK2MSWVNdIXaf8thgPz4YWgF2");
    user.setEmail("aleksa@gmail.com");
    user.setRole(role);

    when(userRepository.findById(id)).thenReturn(Optional.of(user));

    User actualUser = userService.getUserById(id);
    assertEquals(user, actualUser);
    assertEquals(id, actualUser.getId());
}
```

Slika 33. - prikaz koda iz test klase (3. deo)

```

@Test
void getUserByIdTest() {
    Integer id = 14;
    User user = new User();
    Role role = new Role();
    role.setId(1);
    role.setName("ROLE_ADMIN");
    user.setId(14);
    user.setUsername("aleksa");
    user.setPassword("$2a$12$ZdZ046FRbw0r4cv86vUte0f8e0H0ZK2MSWVNdIXaf8thgPz4YWgF2");
    user.setEmail("aleksa@gmail.com");
    user.setRole(role);

    when(userRepository.findById(id)).thenReturn(Optional.of(user));

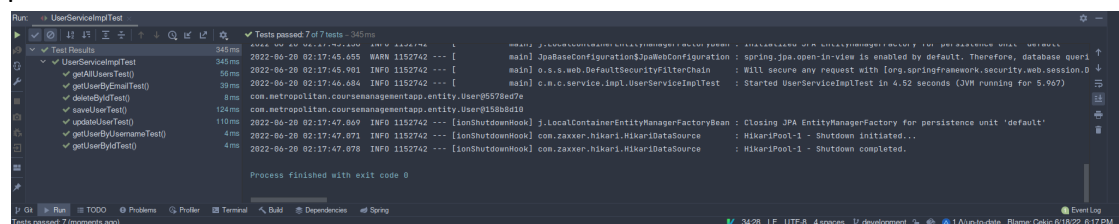
    User actualUser = userService.getUserById(id);
    assertEquals(user, actualUser);
    assertEquals(id, actualUser.getId());
}

@Test
void deleteByIdTest() {
    userService.deleteById(15);
    verify(userRepository, times(wantedNumberOfInvocations: 1)).deleteById(15);
}

```

Slika 34. - prikaz koda iz test klase (4. deo)

Nakon pokretanja testova možemo primetiti da su svi testovi prošli bez problema.



Slika 35. - prikaz rezultata kada se pokrenu testovi

9. Zaključak

Prilikom izrade ovog projektnog zadatka upoznali smo se detaljno sa Spring MVC i Spring Boot frameworkom. Pored toga imali smo priliku da radimo sa thymeleaf-om za prikaz podataka na lep način korišćenjem bootstrapa i css-a. Iskustvo koje smo stekli u izradi ovog projektnog zadatka pored samog ovog projektnog zadatka jeste rad u grupi i rad sa GitHub-om.

10. Literatura

[1]

A. Boroumand, "Check for logged in user with Thymeleaf and Spring security 4 - code by Amir," *Code by Amir*, 14-Mar-2017. [Online].

Available:

<https://www.codebyamir.com/blog/check-for-logged-in-user-with-thymeleaf-and-spring-security-4>. [Accessed: 19-Jun-2022].

[2]

"Thymeleaf page how check user logged in or not," *Stack Overflow*.

[Online]. Available:

<https://stackoverflow.com/questions/64071890/thymeleaf-page-how-check-user-logged-in-or-not>. [Accessed: 19-Jun-2022].

[3]

"Thymeleaf," *Thymeleaf.org*. [Online]. Available:

<https://www.thymeleaf.org/>. [Accessed: 19-Jun-2022].

[4]

"Login - LAMS :: Learning activity management system,"

Metropolitan.ac.rs:8080. [Online]. Available:

<http://lams.metropolitan.ac.rs:8080/lams/index.do>. [Accessed: 19-Jun-2022].

[5]

M. Otto and J. Thornton, "Introduction," *Getbootstrap.com*. [Online].

Available: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>.

[Accessed: 19-Jun-2022].