

BELGRADE
METROPOLITAN
UNIVERSITY

Rent A Car

IT355 Veb sistemi 2

Projektna
dokumentacija

Student: Bogdan Trajković 4405

Profesor: Jovana Jović

Asistent: Tamara Vukadinović

Sadržaj:

Uvod.....
Opis problema
Funkcije i zahtevi
Modeli.....
Kompozicija
Zaključak
Literatura

Uvod

U okviru ovog dokumenta biće detaljno detaljno objašnjeni principi funkcionisanja sistema za iznajmljivanje automobila. Oni će biti objašnjeni kroz niz dijagrama i tabela. Softver korišćen u izradi je PowerDesigner, a programski jezik u kome se sistem razvijao je povezan sa Thymeleaf-om.

Svrha

Glavna svrha ovog projekta je da pruži korisnicima mogućnost da na lak, brz i jednostavan način, kroz nekoliko koraka, rezervišu i izvrše plaćanje auta koji žele da iznajme.

Područje projekta

- (1) Ime sistema je *Rent A Car*
- (2) Ovaj sistem predstavlja osnovne funkcionalnosti iznajmljivanja automobila i manipulacijom podataka.
- (3) Sistem pruža korisnicima mogućnost rezervacija, dok Admin ima mogućnosti dodavanja i izmena.

Opis problema

Rent A Car predstavlja projekat koji simulira sve osnovne funkcionalnosti jedne aplikacije koja je zadužena za rezervisanje i plaćanje kao i izmene, brisanje i dodavanje novih. Pomoći korisničkog naloga, korisnik dodaje rezervaciju i izvršava plaćanje. Ima mogućnost pretrage i sortiranja. Logovanjem Korisnika, sprečene su pregled, izmene i brisanje podataka koje on ne sme da vidi već samo Admin ima tu mogućnost. Svaki korisnik može da pregleda, i izvršava rezervacije samo za svoju listu rezervacija.

Generalni opis sistema

-Sistem podrazumeva aplikaciju za logovanje korisnika putem username-a i password-a. Sama aplikacija je dostupna na svim uređajima preko web aplikacije. Sistem omogućava brz i lak unos željenih kontakata.

1. *Interfejs sistema*: Korisnik sistema je u mogućnosti da se loguje putem naloga i da koristi razlike funkcije u zavisnosti od potrebe

2. *Korisnicki interfejs*: Format ekrana i struktura same aplikacije treba biti takva da je svim korisnicima lako dostupna i vidljiva za korišćenje. Sistem mora da bude korisnički jednostavan i interaktiv. Funkcionalnosti koje sistem obezbeđuje treba adaptirati sve korisnike sistema

3. *Hardverski interfejs*: Sistem ne zahteva nikakvu hardversku podršku, već se nalazi na mreži .

4. *Softverski interfejs*: Thymeleaf I Spring Framework

5. *Komunikacioni interfejs*: Za komunikaciju je korišćena klijent-server arhitektura.

6. *Memorijski zahtevi*: Sistem zahteva minimalnu količinu resursa koji je potreban uređaju za pokretanje veb pretraživača

7. *Operacije*: Sistem zahteva od korisnika da unese jedinstveni username I password pomoću koga korisnik pristupa sistemu.

Funkcije I zahtevi korisnika

Nije potrebno nikakov tehničko predznanje ili iskustvo u upravljanju sistemom za kontakte.

Glavne funkcije uključuju:

- Login/Registracija
- Prikaz liste vozila/registracija/klijenta/transakcija
- Unos, izmene ili brisanje postojećih vozila/registracija/klijenta/transakcija
- Rezervacija vozila

Generalna ograničenja

Obavezno je logovanje putem tačnih podataka, inače nije moguće.

Sledi lista ograničenja koja se odnosi na sistem koji se kreira:

- Korisnik mora imati internet konekciju.
- Sistem mora biti korisnički jednostavan i da prikazuje sve potrebne poruke i poruke greške.
- Sistem ne sadrži interfejs ka nekim drugim sistemima.
- Mora biti u mogudnosti da izvršava više paralelnih operacija odjednom.
- Mora se izbegavati redundatnost podataka i praznih unosa.
- Sistem uvek mora biti adekvatno zaštićen.

Funkcionalni zahtevi

Sledi definisanje funkcionalih zahteva:

1. Korisnik ima pristup sistemu
2. Korisnik kreira svoj nalog
3. Korisnik može da dodaje nove rezervacije
4. Korisnik može da vrši pregled svojih rezervacija
5. Korisnik može da plati rezervaciju
6. Korisnik može da filtrira vozila
7. Admin može da upravlja svim podacima

Korisnik ima pristup sistemu i podrazumeva sledeće elemente:

Uvod - Svrha ovog sistema je da omogudi korisnicima da pristupe sistemu.

Ulaz - Korisnik ima uvid u početni ekran sistema.

Proces - Ulazni podaci se čuvaju u bazi podataka.

Izlaz - Korisnik jasno zna kako da pristupi sistemu.

Korisnik kreira svoj nalog podrazumeva sledeće elemente:

Uvod - Svrha je da omogudi korisnicima da uz pomoć jedinstvenih podataka pristupa sistemu.

Ulaz - Korisnik popunjava polja za kreiranje naloga.

Proces - Procesom verifikacije se potvrđuju validnost podataka

Izlaz - Korisnik uz pomoć naloga može da pristupi sistemu.

Korisnik može da pregleda listu rezervacija podrazumeva sledeće elemente:

Uvod - Svrha je da omogući korisnicima uvid u listu rezervacija.

Ulaz - Korisnik klikom na Moje Rezervacije vrši pregled svih svojih rezervacija.

Izlaz - Korisnik sada ima uvid u dostupne informacije.

Korisnik može da pregleda listu vozila i rezervise:

Uvod - Svrha je da omogući korisnicima uvid u listu vozila i rezervisati.

Ulaz - Korisniku se otvara strana sa vozilima i ima opciju rezerviši.

Izlaz - Korisnik unosi datum i rezerviše vozilo

Admin može da doda vozila u listu vozila podrazumeva sledeće elemente:

Uvod - Svrha je da omogući adminu da dodaju novo vozilo u listu.

Ulaz - Admin nakon unosa podataka, čuva te podatke o vozilu.

Izlaz - Admin je uspešno dodao novo vozilo.

Admin može da izbriše vozilo iz liste vozila podrazumeva sledeće elemente:

Uvod - Svrha je da omogući adminu brisanje vozila iz liste.

Ulaz - Admin nakon prikaza liste vozila bira koje želi da obriše, vrši brisanje.

Izlaz - Admin je uspešno obrisao vozilo iz liste.

NEFUNKCIONALNI ZAHTEVI

Performanse su definisane na sledeći način:

Vreme odziva sistema treba biti do 3 sekunde u najboljem slučaju.

Vreme odziva sistema se odnosi na vreme čekanja dok se ne pristupi sistemu, i dok sistem ne povuče podatke iz baze podataka.

Sistem ne treba da prikazuje vreme odziva, broj korisnika sistema, kao ni njihov rast.

Sistem mora da radi u realnom vremenu i da podnese više korisnika odjednom.

Pouzdanost je definisana na sledeći način:

Sistem treba da bude dostupan 24 časa, u svakom trenutku.

Sistem će uvek pružati informacije u realnom vremenu.

Sistem treba da ima visok stepen tolerancije. Na primer, ukoliko korisnik unese pogrešne podatke, sistem ne treba da prestane sa radom, već da prepozna pogrešan unos i korisnika obavesti porukom o grešci

Upotrebljivost je definisana na sledeći način:

Sistem treba da obezbedi lako pristupačan grafički interfejs koji je lak za korišćenje za sve korisnike sistema.

Veb interfejs treba da bude intuitivan i lak za navigaciju. Korisniku meni i opcije treba da budu laki za razumevanje.

Sva obaveštenja o greškama koje proizvodi sistem treba da budu jasne, kratke, smislene i razumljive.

Integritet je definisan na sledeći način:

Samo administrator sistema ima pravo da promeni parametre sistema, kao što je menjanje osjetljivih podataka.

Sistem treba da bude obezbeđen i da koristi enkripciju za zaštitu baze podataka.

Korisnik treba da potvrди autentifikaciju pre nego što dobije pristup bilo kojim detaljima.

Samostalnost je definisana na sledeći način:

Sistem će koristiti minimalni broj drugih sistema, kao što je baza podataka.

Portabilnost je definisana na sledeći način:

Sistemu je moguće pristupiti sa bilo kog računara koji ima stabilnu internet konekciju

KONCEPTUALNI MODEL BAZE PODATAKA

Prezentacija konceptualnog modela baze podataka sistema.

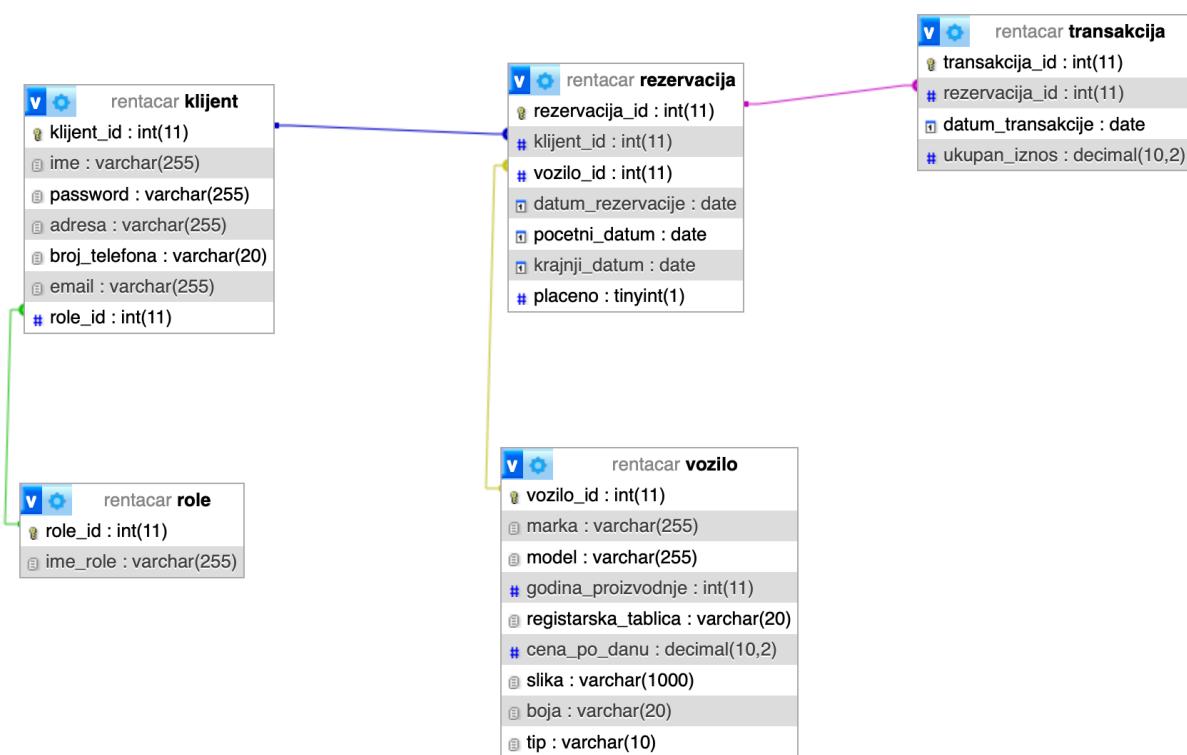
Sistem koristi phpMyAdmin bazu podataka, fleksibilnog sadržaja. Slededom slikom je prikazan konceptualni model baze podataka

-DODATI KONCEPTUALNI MODEL AKO TREBA-

FIZIČKI MODEL BAZE PODATAKA

Prezentacija fizičkog modela baze podataka sistema.

Sistem koristi phpMyAdmin bazu podataka, fleksibilnog sadržaja. Slededom slikom je prikazan fizički model baze podataka.

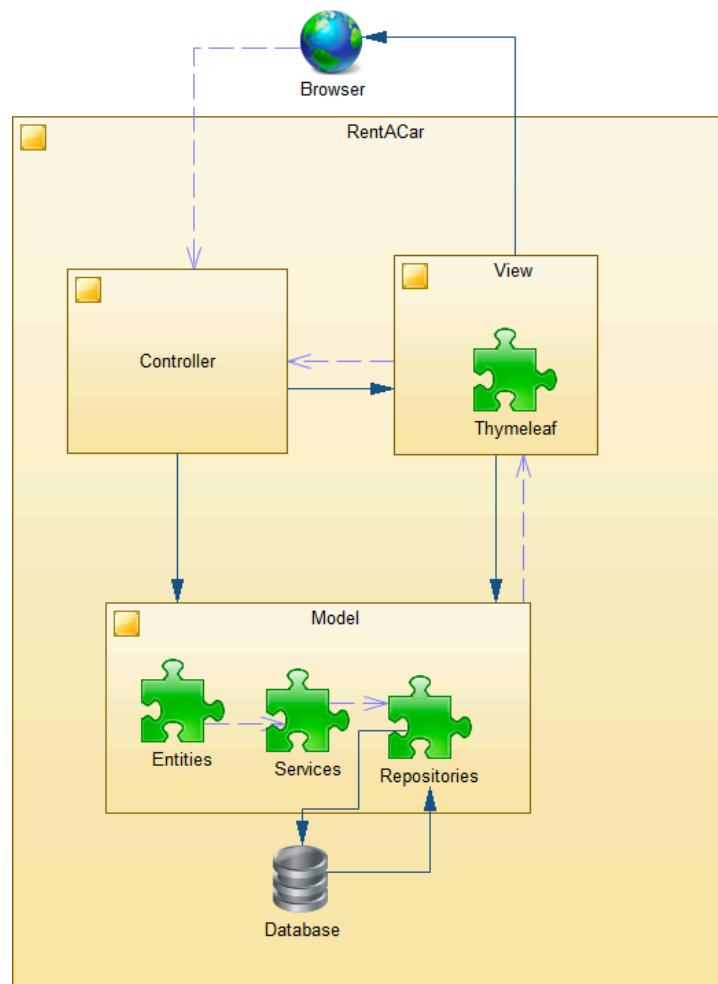


DIJAGRAM ARHITEKTURE

Dijagram arhitekture predstavlja arhitekturu jednog veb sistema.

Pored osnovnih delova za funkcionisanje jednog veb sistema, tu je baza podataka koja predstavlja phpMyAdmin bazu podataka sa osnovnim mogućnostima manipulacije nad podacima u bazi podataka (čitanje, upisivanje, ažuriranje i brisanje).

Sledećom slikom je prikazan dijagram arhitekture.



U ovom slučaju primenjuje se troslojna arhitektura. Troslojna arhitektura je takva arhitektura kod koje je potrebno uvesti još jedan sloj, srednji sloj, u kome se nalazi aplikativni server.

Troslojna arhitektura sastoji se iz tri sloja:

1. Korisnički sloj (User interface),
2. Srednji sloj, sloj upravljanja procesima (Middle Tier Server),
2. Sloj podataka, sloj upravljanja podacima (Data access).

Klijent/server sistem sa troslojnom arhitekturom predstavlja sistem sa tri nezavisna podsistema:

1. Podsistem za interakciju sa korisnikom (implementira funkcije korisničkog interfejsa)
2. Podsistem za implementaciju osnovnih funkcija sistema (Takozvana „poslovna logika“), što u ovom slučaju predstavlja aplikacioni sloj
3. Podsistem za rukovanje bazom podataka, pri čemu se pre svega misli na fizičko smeštanje podataka (upravljanje bazom podataka), Logika aplikacije nalazi se u aplikacionom sloju, i njegova namena je da izvršava programski kod koji implementira logiku aplikacije.

Ovako organizovan sistem je jednostavniji za održavanje jer je moćude nezavisno razvijati korisnički interfejs i logiku aplikacije.

KOMPOZICIJA SISTEMA

Sam sistem se sastoji iz dve osnovne komponente: backend i frontend.

Backend – predstavlja logiku sistema

Frontend – predstavlja korisnički interfejs sistem

IMPLEMENTACIJA I STRUKTURA

Postoji nekoliko grupa implementacionog koda.

Prezentacija implementacionog koda vrši se kroz nekoliko prezentacionih faza:

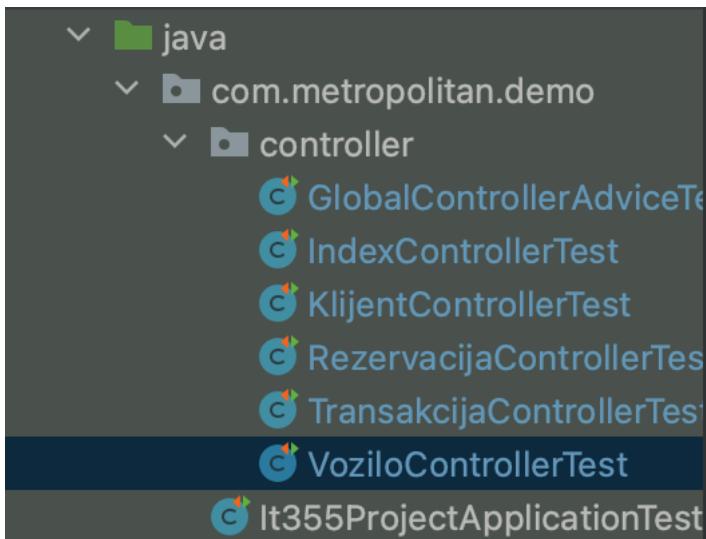
API - controller klasa.

Podaci - modelska klasa.

Repozitorijumi - pretraga po nekom parametru iz baze podataka. Upravo kroz ove korake bude prikazan specifičan Java kod generisan primenom Spring frejmvorka

TESTIRANJE

Za Testiranje koristimo JUnit testove. Testirani su kontroleri i sve metode u njima.
Prikazani su paketi i testovi koji su uspešno završeni.



The screenshot shows an IDE interface with the code editor open to a file named 'RezervacijaControllerTest.java'. The code is a JUnit test for a 'RezervacijaController' class. The editor shows imports for java.util.List, org.junit.jupiter.api.Assertions, and org.mockito.Mockito. The test class has several methods annotated with @Mock and @InjectMocks. The bottom of the screen shows the test results: 'Tests passed: 5 of 5 tests – 268ms'. The log pane displays the names and execution times of the five passed tests.

```
import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;
import static org.mockito.Mockito.times;

no usages ▾ trajkovicbogdan *
@ExtendWith(MockitoExtension.class)
class RezervacijaControllerTest {

    4 usages
    @Mock
    private RezervacijaService rezervacijaService;
    5 usages
    @InjectMocks
    private RezervacijaController rezervacijaController;
    2 usages
    @Mock
    private VoziloService voziloService;

    @Test
    void getAllRezervacije() {
        List<Rezervacija> rezervacije = Arrays.asList(
                new Rezervacija("Vozilo 1", "Klijent 1", "Transakcija 1"),
                new Rezervacija("Vozilo 2", "Klijent 2", "Transakcija 2")
        );
        when(rezervacijaService.getAllRezervacije()).thenReturn(rezervacije);
        List<Rezervacija> rezervacijeRetorno = rezervacijaController.getAllRezervacije();
        assertEquals(rezervacije, rezervacijeRetorno);
    }

    @Test
    void addRezervacija() {
        Rezervacija rezervacija = new Rezervacija("Vozilo 1", "Klijent 1", "Transakcija 1");
        rezervacijaController.addRezervacija(rezervacija);
        verify(rezervacijaService, times(1)).addRezervacija(rezervacija);
    }

    @Test
    void addRezervacija_withErrors() {
        Rezervacija rezervacija = new Rezervacija("Vozilo 1", "Klijent 1", "Transakcija 1");
        rezervacija.setVozilo(null);
        rezervacija.setKlijent(null);
        rezervacija.setTransakcija(null);
        assertThrows(IllegalArgumentException.class, () -> rezervacijaController.addRezervacija(rezervacija));
    }

    @Test
    void showCreateRezervacijaForm() {
        Map<String, String> formValues = new HashMap<>();
        Map<String, String> expectedFormValues = new HashMap<>();
        assertEquals(formValues, expectedFormValues);
    }

    @Test
    void getMyPurchasesPage() {
        Map<String, String> formValues = new HashMap<>();
        Map<String, String> expectedFormValues = new HashMap<>();
        assertEquals(formValues, expectedFormValues);
    }
}
```

Run: RezervacijaControllerTest

Tests passed: 5 of 5 tests – 268ms

Process finished with exit code 0

it355PZ | src | test | java | com | metropolitan | demo | controller | IndexControllerTest

Project Commit Pull Requests Database Notifications

IndexControllerTest.java VoziloRepository.java VoziloServiceImpl.java VoziloControllerTest.java KlijentControllerTest.java GlobalControllerAdviceTest.java IndexControllerTest.java

12 import java.util.ArrayList;
13 import java.util.List;
14
15
16 import static org.junit.jupiter.api.Assertions.*;
17 import static org.mockito.Mockito.when;
18 import static org.mockito.Mockito.verify;
19
20 no usages * trajkovicbogdan *
21 @ExtendWith(MockitoExtension.class)
22 class IndexControllerTest {
23
24 3 usages
25 @Mock
26 private VoziloService voziloService;
27
28 2 usages
29 @Mock
30 private VoziloRepository voziloRepository;

Run: IndexControllerTest

Tests passed: 6 of 6 tests - 246 ms

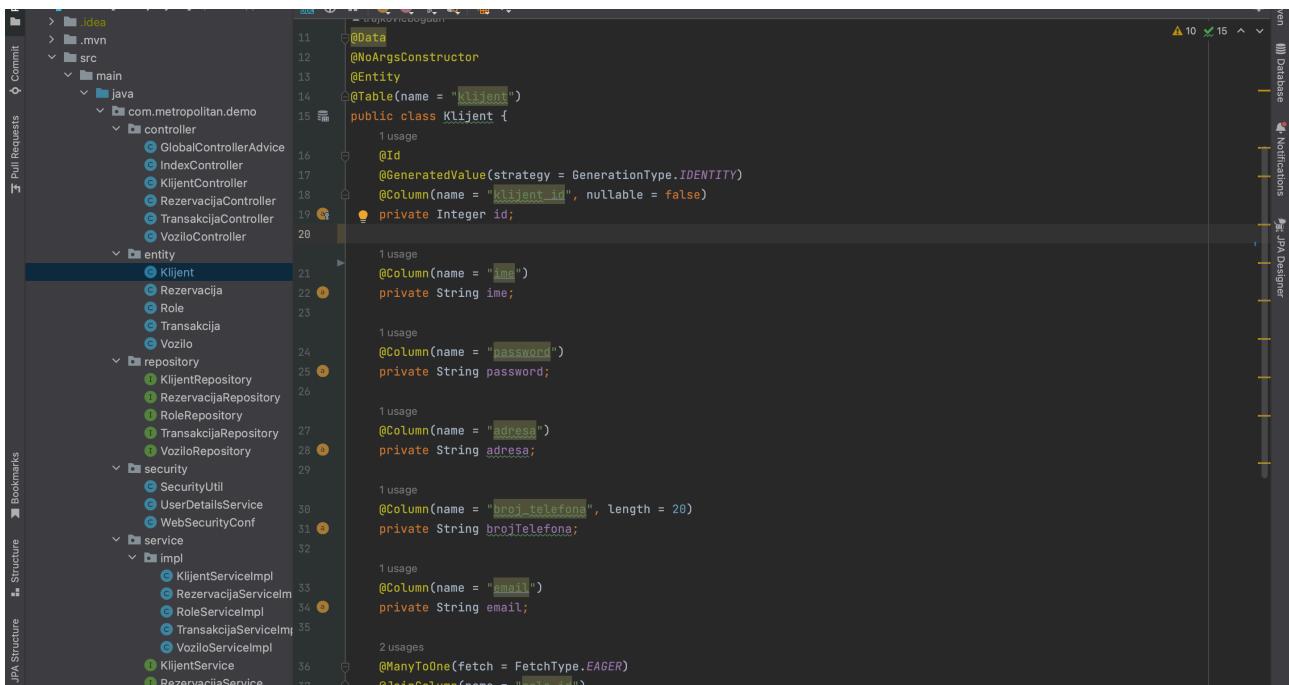
IndexControllerTest (com.metron... 246 ms /Users/bogdantrajkovic/Library/Java/JavaVirtualMachines/correcto-17.0.6/Contents/Home/bin/java ...
pronadjiVozilaPoBoji() 242 ms
pronadjiVozilaPoMarci() 1ms
prikaaziDetaljeVozila() 1ms
sortirajPoCeni()
openAdminPanel()
showLoginForm()

Process finished with exit code 0

Tests passed: 6

ENTITETI APLIKACIJE

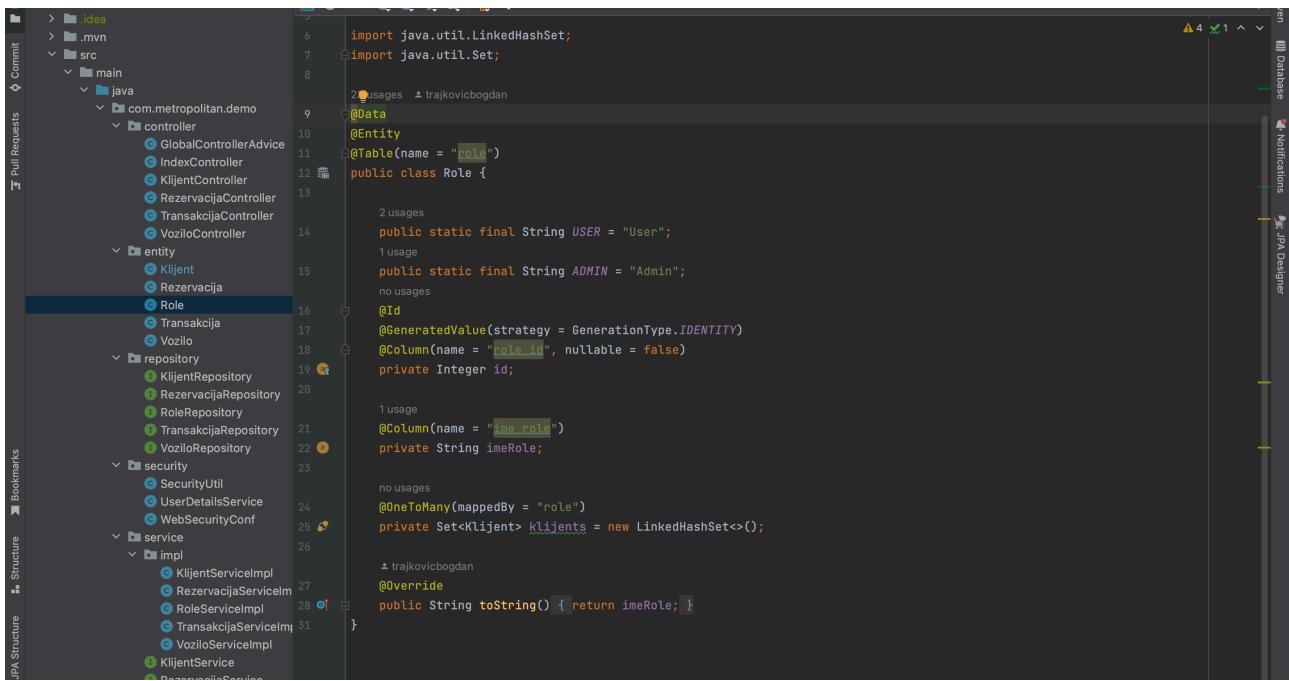
Klijent



The screenshot shows the Java code for the `Klijent` entity. The code defines the `Klijent` class with annotations for the table name (`@Table(name = "klijent")`) and primary key (`@Id @GeneratedValue(strategy = GenerationType.IDENTITY)`). It includes fields for `ime` and `password`, and a many-to-one relationship to `Vozilo`. The code is part of a larger project structure with controllers, repositories, and services.

```
11  * @Data
12  * @NoArgsConstructor
13  * @Entity
14  * @Table(name = "klijent")
15  * public class Klijent {
16      /**
17      * @Id
18      * @GeneratedValue(strategy = GenerationType.IDENTITY)
19      * @Column(name = "klijent_id", nullable = false)
20      */
21      private Integer id;
22
23      /**
24      * @Column(name = "ime")
25      */
26      private String ime;
27
28      /**
29      * @Column(name = "password")
30      */
31      private String password;
32
33      /**
34      * @Column(name = "adresa")
35      */
36      private String adresa;
37
38      /**
39      * @Column(name = "broj_telefona", length = 20)
40      */
41      private String brojTelefona;
42
43      /**
44      * @Column(name = "email")
45      */
46      private String email;
47
48      /**
49      * @ManyToMany(fetch = FetchType.EAGER)
50      * @JoinColumns({
51      *     @JoinColumn(name = "role_id", referencedColumnName = "role_id"),
52      *     @JoinColumn(name = "klijent_id", referencedColumnName = "klijent_id")
53      * })
54      */
55      Set<Role> role;
56  }
```

Role



The screenshot shows the Java code for the `Role` entity. The code defines the `Role` class with annotations for the table name (`@Table(name = "role")`) and primary key (`@Id @GeneratedValue(strategy = GenerationType.IDENTITY)`). It includes fields for `imeRole` and a one-to-many relationship to `Klijent`. The code is part of a larger project structure with controllers, repositories, and services.

```
6  import java.util.LinkedHashSet;
7  import java.util.Set;
8
9  /**
10  * @Data
11  * @Entity
12  * @Table(name = "role")
13  * public class Role {
14
15      /**
16      * public static final String USER = "User";
17      */
18
19      /**
20      * public static final String ADMIN = "Admin";
21      */
22
23      /**
24      * @Id
25      * @GeneratedValue(strategy = GenerationType.IDENTITY)
26      * @Column(name = "role_id", nullable = false)
27      */
28      private Integer id;
29
30      /**
31      * @Column(name = "ime_role")
32      */
33      private String imeRole;
34
35      /**
36      * @OneToOne(mappedBy = "role")
37      */
38      private Set<Klijent> klijents = new LinkedHashSet<>();
39
40      /**
41      * @Override
42      * public String toString() { return imeRole; }
43      */
44  }
```

Rezervacija

The screenshot shows the IntelliJ IDEA interface with the code editor open. The file is `Rezervacija.java` located in the `com.metropolitan.demo.entity` package. The code defines a JPA entity with annotations like `@Data`, `@Entity`, and `@Table`. It has a primary key `id` and a many-to-one relationship to `Klijent` and `Vozilo`. It also contains fields for `datumRezervacije` and `pocetniDatum`.

```
11  @Data
12  @Entity
13  @Table(name = "rezervacija")
14  public class Rezervacija {
15      no usages
16      @Id
17      @GeneratedValue(strategy = GenerationType.IDENTITY)
18      @Column(name = "rezervacija_id", nullable = false)
19      private Integer id;
20
21      1 usage
22      @ManyToOne(fetch = FetchType.EAGER)
23      @JoinColumn(name = "klijent_id")
24      private Klijent klijent;
25
26      1 usage
27      @ManyToOne(fetch = FetchType.EAGER)
28      @JoinColumn(name = "vozilo_id")
29      private Vozilo vozilo;
30
31      no usages
32      @DateTimeFormat(pattern = "yyyy-MM-dd")
33      @Column(name = "datum_rezervacije")
34      private LocalDate datumRezervacije;
35
36      no usages
37      @DateTimeFormat(pattern = "yyyy-MM-dd")
38      @Column(name = "pocetni_datum")
39      private LocalDate pocetniDatum;
```

Vozilo

The screenshot shows the IntelliJ IDEA interface with the code editor open. The file is `Vozilo.java` located in the `com.metropolitan.demo.entity` package. The code defines a JPA entity with annotations like `@Data`, `@Entity`, and `@Table`. It has a primary key `id` and columns for `marka`, `model`, `godinaProizvodnje`, `registarskaTablica`, `cenaPoDanu`, and `slika`.

```
11  @Data
12  @Entity
13  @Table(name = "vozilo")
14  public class Vozilo {
15      no usages
16      @Id
17      @GeneratedValue(strategy = GenerationType.IDENTITY)
18      @Column(name = "vozilo_id", nullable = false)
19      private Integer id;
20
21      1 usage
22      @Column(name = "marka")
23      private String marka;
24
25      1 usage
26      @Column(name = "model")
27      private String model;
28
29      1 usage
30      @Column(name = "godina_proizvodnje")
31      private Integer godinaProizvodnje;
32
33      1 usage
34      @Column(name = "registarska_tablica", length = 20)
35      private String registarskaTablica;
36
37      1 usage
38      @Column(name = "cena_po_danu", precision = 10, scale = 2)
39      private BigDecimal cenaPoDanu;
40
41      1 usage
42      @Column(name = "slika", length = 1000)
43      private String slika;
```

Transakcija

The screenshot shows the IntelliJ IDEA interface with the code editor open to the file `Transakcija.java`. The code defines a JPA entity named `Transakcija` with attributes `id`, `rezervacija`, and `ukupan_iznos`. The code editor includes syntax highlighting, code completion, and navigation tools. The project structure on the left shows the package `com.metropolitan.demo.entity` containing the `Transakcija` class.

```
1 package com.metropolitan.demo.entity;
2
3 import javax.persistence.*;
4 import lombok.Data;
5
6 import java.math.BigDecimal;
7 import java.time.LocalDate;
8
9 import static org.junit.Assert.*;
10
11 @Data
12 @Entity
13 @Table(name = "transakcija")
14 public class Transakcija {
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     @Column(name = "transakcija_id", nullable = false)
18     private Integer id;
19
20     @ManyToOne(fetch = FetchType.EAGER)
21     @JoinColumn(name = "rezervacija_id")
22     private Rezervacija rezervacija;
23
24     @Column(name = "datum_transakcije")
25     private LocalDate datumTransakcije;
26
27     @Column(name = "ukupan_iznos", precision = 10, scale = 2)
28     private BigDecimal ukupanIznos;
29 }
```

Stranice sa funkcionalnostima sistema

The screenshot shows a web browser window for 'localhost' displaying the 'Rent a Car' website. The header includes a logo, navigation links for 'Početna', 'Kontakt', and 'Login', and search functions. The main content area is titled 'RENT A CAR' and features four car models in a grid:

- Toyota Corolla** (2020) - Red hatchback - Price: 50.00€ - Registration Plate: NS-123-AA - Buttons: Rezerviši, Detaljnije
- Volkswagen Golf** (2018) - White hatchback - Price: 60.00€ - Registration Plate: BG-456-BB - Buttons: Rezerviši, Detaljnije
- Ford Focus** (2019) - Blue hatchback - Price: 55.00€ - Registration Plate: BG-789-CC - Buttons: Rezerviši, Detaljnije
- BMW 520d** (2017) - Black sedan - Price: 90.00€ - Registration Plate: NI-123-CC - Buttons: Rezerviši, Detaljnije

Below the grid, there are links for 'O nama', 'Kategorije', 'Korisni linkovi', and 'Preplatite se na naš'.

Početna strana pre logovanja ili registracije bilo kog korisnika.

Login

Username:

Password:

Don't have an account? [Register](#) Back home [Home](#)

Login strana na kojoj se može korisnik ulogovati kao User ili kao Admin, ukoliko nema nalog može se registrovati.

Register

Name Password

Adresa

Broj Telefona

Email

Nakon uspešnog logovanja ili registracije korisnik ima sledeće mogućnosti

The screenshot shows a web browser window for 'localhost / localhost / rentacar | phpMyAdmin 5.2.1'. The title bar says 'localhost' and the tab says 'Rent a Car'. The main content area displays four car models in cards:

- Toyota Corolla** (2020) - Red - Price: 50.00€ - Registration Plate: NS-123-AA - Buttons: Rezervisi, Detaljnije
- Ford Focus** (2019) - Blue - Price: 55.00€ - Registration Plate: BG-789-CC - Buttons: Rezervisi, Detaljnije
- Volkswagen Golf** (2018) - White - Price: 60.00€ - Registration Plate: BG-456-BB - Buttons: Rezervisi, Detaljnije
- BMW 520d** (2017) - Black - Price: 90.00€ - Registration Plate: NI-123-CC - Buttons: Rezervisi, Detaljnije

At the top right, there is a search bar with a dropdown menu titled 'Sve boje' (All colors) containing 'crvena' (red), 'plava' (blue), 'bela' (white), and 'crna' (black). There is also a 'Sortiraj po ceni' (Sort by price) button.

Korisnik može sortirati vozila, kao i vršiti pretragu po ceni ili po boji auta.

Osim toga može pregledati i detaljniji prikaz vozila sa više informacija i većom slikom.

The screenshot shows a web browser window with the address bar displaying 'localhost'. The main content area is titled 'Detalji vozila' (Car Details) and features a white Volkswagen Golf hatchback. Below the image, the car's name is displayed as 'Volkswagen Golf'. To the right of the car, several descriptive text labels are listed:

- Boja: bela
- Tip: benzin
- Godište: 2018
- Cena po danu: 60.00 €
- Registratne tablice: BG-456-BB

Klikom na dugme Rezerviši od njega se traži da unese početni i krajnji datum kako bi rezervisao vozilo.

The screenshot shows a web browser window with the following details:

- Address Bar:** localhost
- Page Title:** Rezervacija
- Content Area:** A form titled "Rezervisi" containing two input fields:
 - Pocetni Datum: 05/06/2023
 - Krajnji Datum: 05/06/2023
- Buttons:** A blue button labeled "Rezervisi" at the bottom of the form.

localhost / localhost / rentacar | phpMyAdmin 5.2.1

Add Vozilo

Navbar Klijenti Vozila Rezervacije Transakcije Login

Dodaj Vozilo

Marka	Model
<input type="text"/>	<input type="text"/>
Godina Proizvodnje	
<input type="text"/>	
Registarska Tablica	
<input type="text"/>	
Cena Po Danu	
<input type="text"/>	
Slika	
<input type="text"/>	<input type="text"/>
Boja	Tip Motora
<input type="text"/>	<input type="text"/>
Add Vozilo	

Korinsik u meniju ima opciju Moje Rezervacije gde dobija prikaz samo njegovih rezervacija i može da izvrši neku rezervaciju odnosno da je plati ili da je poništi.

localhost / localhost / rentacar | phpMyAdmin 5.2.1

Rezervacije

Lista rezervacija

Marka vozila	Model vozila	Godina proizvodnje	Registarska tablica	Cena po danu	Slika vozila	Datum rezervacije	Početni datum	Krajnji datum	Plati	Poništi
BMW	BMW 520d	2017	NI-123-CC	90.00		2023-05-30	2023-05-31	2023-06-02	Plati:	
Volkswagen	Golf	2018	BG-456-BB	60.00		2023-05-31	2023-07-01	2023-05-03	Plati:	
Toyota	Corolla	2020	NS-123-AA	50.00		2023-05-31	2023-05-31	2023-06-02		
Toyota	Corolla	2020	NS-123-AA	50.00		2023-06-01	2023-06-10	2023-06-11		
Ford	Focus	2019	BG-789-CC	55.00		2023-06-04	2023-06-10	2023-06-11		

Ukoliko se Admin uloguje sa svojim podacima, dobija prikaz Admin panela sa sledećim mogućnostima:

Dobrodošli u Admin Panel

Upravljaljajte vašim veb sajtom lako pomoću naših moćnih alata.

Vozila	Rezervacije	Transakcije	Klijenti
Upravljaljajte informacijama o vozilima. Idi na Vozila	Pregledajte i upravljaljajte rezervacijama. Idi na Rezervacije	Prikaz i upravljanje klijentskim transakcijama. Idi na Transakcije	Upravljanje informacijama o klijentima. Idi na Klijente

Admin može pregledati sva vozila, rezervacije, transakcije i klijente i izvršavati izmene, brisanja i dodavanja novih.

Dostupna vozila

Sortiraj po ceni

Slika	Marka	Model	Godina Proizvodnje	Cena Po Danu	Registraske Tablice	Boja	Tip Motora	Edit	Delete
	Toyota	Corolla	2020	50.00€	NS-123-AA	crvena	dizel		
	Volkswagen	Golf	2018	60.00€	BG-456-BB	bela	benzin		
	Ford	Focus	2019	55.00€	BG-789-CC	plava	benzin		
	BMW	520d	2017	90.00€	NI-123-CC	crna	dizel		

Dodavanje novog vozila i ažuriranje postojećeg.

localhost / localhost / rentacar | phpMyAdmin 5.2.1

Update Vozilo

Navbar Klijenti Vozila Rezervacije Transakcije Login

Update Vozilo

Marka	Model
Volkswagen	Golf
Godina Proizvodnje	
2018	
Registarska Tablica	
BG-456-BB	
Cena Po Danu	
60.00	
Slika	
https://www.greencap.com/wp-content/uploads/VW-Golf	
Boja	Tip Motora
Golf	Golf

Update Vozilo

localhost / localhost / rentacar | phpMyAdmin 5.2.1

Rezervacije

Navbar Home

Rezervacije

Lista rezervacija

E-mail	Marka vozila	Model vozila	Godina proizvodnje	Registarska tablica	Cena po danu	Slika vozila	Datum rezervacije	Početni datum	Krajnji datum	Edit	Delete
mail.com	BMW	520d	2017	NI-123-CC	90.00		2023-05-29	2023-05-30	2023-05-31	 	
mail.com	Toyota	Corolla	2020	NS-123-AA	50.00		2023-05-29	2023-05-30	2023-06-02	 	
gmail.com	BMW	520d	2017	NI-123-CC	90.00		2023-05-30	2023-05-31	2023-06-02	 	
mail.com	Volkswagen	Golf	2018	BG-456-BB	60.00		2023-05-31	2023-06-02	2023-05-07	 	

Prikaz svih rezervacija i mogućnost za Edit i Delete.

Admin ima i mogućnost prikaza svih transakcija koje su izvršene.

localhost / localhost / rentacar | phpMyAdmin 5.2.1

Prikaz transakcija

Navbar Home

Search

Search

Prikaz transakcija

ID	Rezervacije	Datum Transakcije	Ukupan Iznos
4	15	2023-05-31	100.00
5	15	2023-06-01	100.00
9	16	2023-06-01	50.00
10	16	2023-06-01	50.00
11	15	2023-06-01	100.00
12	17	2023-06-04	55.00
13	18	2023-06-04	120.00

Kao posledje admin ima mogućnost da vidi korisnike sistema i da ažurira njihove podatke ili da ukloni korisnika.

Ime	Password	Adresa	Broj Telefona	Email	Role	Edit	Delete
Test	\$2a\$10\$MDOyC4zWpVbzrNxQocw1r.EaaV0KROsyAiQOR5IdxOSu10ySc1KbO	Nis	23143242	test@gmail.com	User		
Bogdan	\$2a\$10\$XAKISk4T2c1pxIX9A/sYluEZHspecS4hUeStvluP8H8AEjPPufvAq	Nis	12345675	bogdan@gmail.com	User		
Admin	\$2a\$10\$MDOyC4zWpVbzrNxQocw1r.EaaV0KROsyAiQOR5IdxOSu10ySc1KbO	Nis	123123	admin@admin.com	Admin		
Pera	\$2a\$10\$rDMg1u/C2kyUEU2ldt4vkOsIXe4ybkh3dECvmcOya/b0P5vDGLTQi.	Bg	123441	pera@gmail.com	User		
Luka	\$2a\$10\$rb8/Nw2/NUCrcTuMnKJ7JOXX7UmkrNBp1Ra0BC57OW7EldnZ/EsMW	Bg	5445764	luka@gmail.com	User		

ZAKLJUČAK

Kao što je već navedeno, tema ovog uspešno realizovanog projekta je bila Rent A Car, aplikacija za iznajmljivanje automobila.

Backend deo radjen je u Spring Frameworku, Frontend je preko Thymeleaf-a. Projekat uspešno ispunjava sve korisničke zahteve i služi kao pokazatelj stečenog znanja za celokupan rad preko semestra. Moguće je vršiti nadogradnje i ubacivanje novijih verzija, kao i osposobljavanje i priprema aplikacije za strano i domaće trжиšte.

Literatura:

- [1] *LAMS*, <http://lams.metropolitan.ac.rs:8080/lams/>
- [2] *Spring MVC Tutorial*, <https://www.javatpoint.com/spring-mvc-tutorial>
- [3] *Tutorial: Thymeleaf + Spring*, <https://www.thymeleaf.org/doc/tutorials/3.0/thymeleafspring.html>