

*Jesenji semestar, 2022/23*

*PREDMET: SE321 OBEZBEĐENJE KVALITETA, TESTIRANJE I ODRŽAVANJE SOFTVERA*

***PROJEKTNI ZADATAK***

Tema: Sistem za praćenje rada restorana/kafića

Ime i prezime: **Đorđe Ristić**

Broj indeksa: **4365**

Profesor: **Jovana Jović**

Asistent: **Tamara Vukadinović**

Sadržaj

[**1.** **Opis softverskog sistema** 2](#_Toc137082758)

[**2.** **Slučajevi korišćenja** 3](#_Toc137082759)

[**3.** **Nefunkcionalni zahtevi** 4](#_Toc137082760)

[**4.** **Funkcionalni zahtevi** 5](#_Toc137082761)

[**5. Arhitektura sistema** 7](#_Toc137082762)

[**6. Implementacija** 8](#_Toc137082763)

[**7. Testiranje** 15](#_Toc137082764)

[**8. Zaključak** 18](#_Toc137082765)

**[9.](#_Toc137082766)****[Reference](#_Toc137082766)** [18](#_Toc137082766)

1. **Opis softverskog sistema**

Ova dokumentacija opisuje funkcionalnosti i zahteve za POS (Point of Sale) sistem namenjen restoranima. Sistem omogućava korisnicima, koji mogu biti kako obični korisnici tako i administratori, da upravljaju menijem, kreiraju nove jelovnike, ažuriraju postojeće, sortiraju jelovnik po ceni i imenu. Takođe, korisnicima se omogućava jednostavno dodavanje stavki iz menija u narudžbinu klikom na sto. Svaki sto pamti svoju narudžbinu sve dok se ne izvrši naplata. Naplata se vrši pritiskom na dugme "Naplati", pri čemu se stavke sa stola brišu, a narudžbina se pamti u bazi podataka.

Administrator ima privilegije da dodaje nove radnike, pregleda postojeće radnike iz baze i ažurira njihove podatke.

Ovaj dokument će definisati funkcionalne zahteve za projekat, koji moraju biti ispunjeni u skladu sa standardima koji se primenjuju pri kreiranju ovih zahteva.

Tehnologije korišćene za kreiranje sistema :

Backend: *Spring Boot MVC*

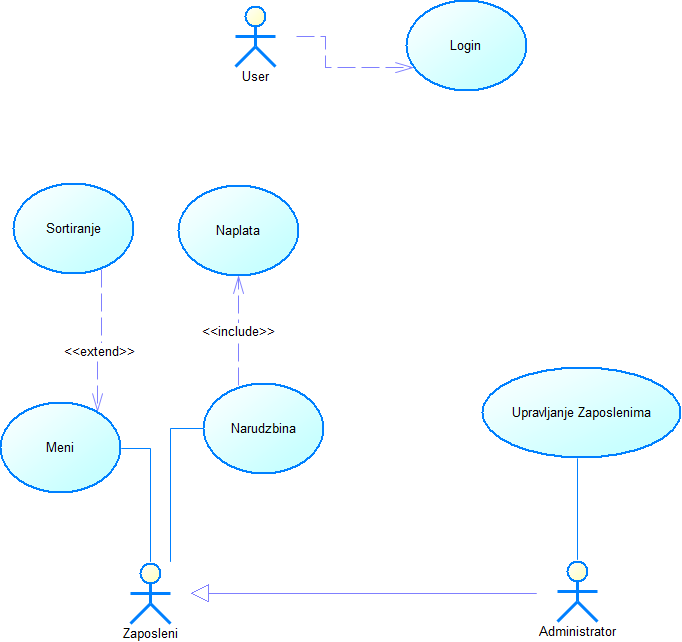
Frontend: *Thymeleaf*

DataBase: *MySQL*

1. **Slučajevi korišćenja**

Use Case dijagram predstavlja vizuelni prikaz akcija koje korisnici sistema mogu izvršiti koristeći funkcionalnosti softvera. Akteri, odnosno korisnici sistema, identifikovani su na dijagramu, dok su slučajevi korišćenja predstavljeni kao elipsaste ikone sa nazivima. Ovaj dijagram pruža pregled interakcija između korisnika i sistema, identifikujući osnovne funkcionalnosti koje su dostupne korisnicima i kako se korisnici mogu koristiti tim funkcionalnostima u svojim aktivnostima.

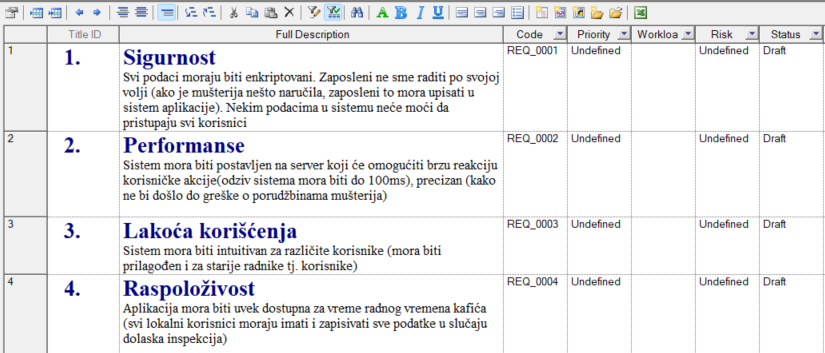
Kroz Use Case dijagram, može se jasno videti koje korisničke akcije su podržane u sistemu, kao što su pregled menija, otvaranje narudžbine, dodavanje jela u narudžbinu, izvršavanje naplate, dodavanje novih jela i upravljanje radnicima. Ove akcije se prikazuju kao slučajevi korišćenja, koji su povezani sa odgovarajućim korisnicima sistema. Dijagram takođe ilustruje interakcije između korisnika i sistema, pružajući osnovnu sliku o toku aktivnosti i funkcionalnostima koje korisnici mogu očekivati.

**

*Slika 1: Use Case Diagram sistema*

1. **Nefunkcionalni zahtevi**

Nefunkcionalni zahtevi su zahtevi koji definišu karakteristike i ograničenja sistema, a ne fokusiraju se direktno na specifične funkcionalnosti. Oni se odnose na performanse, bezbednost, pouzdanost, upotrebljivost, skalabilnost i druge aspekte softverskog sistema. Evo nekoliko primera nefunkcionalnih zahteva:



*Slika 3: Nefunkcionalni zahtevi sistema*

1. **Funkcionalni zahtevi**

Funkcionalni zahtevi opisuju specifične funkcionalnosti i akcije koje softverski sistem treba da pruži korisnicima. Ovi zahtevi definišu očekivano ponašanje sistema i identifikuju ključne funkcionalnosti koje treba implementirati. U narednoj slici su opisani funkcionalni zahtevi za POS sistem restorana:



*Slika 4: Funkcionalni zahtevi sistema*

1. **Arhitektura sistema**

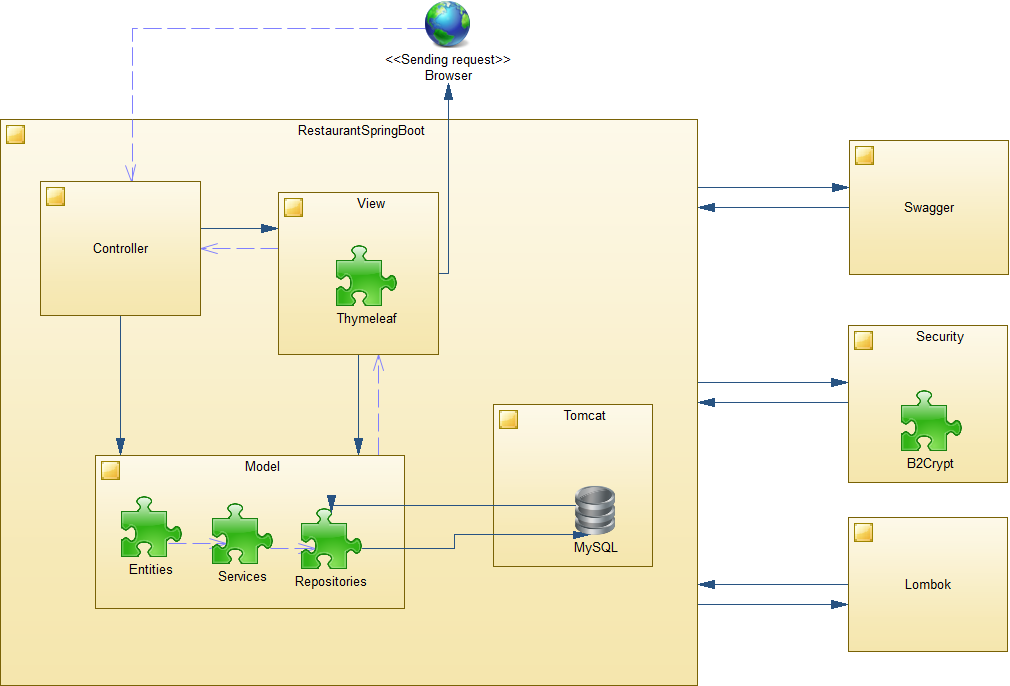
U Spring Boot projektu za sistem za upravljanje restoranom, primenjena je arhitektura Model-View-Controller (MVC). MVC je popularna arhitektura koja omogućava jasnu razdeljenost odgovornosti i organizaciju koda. U ovom kontekstu, MVC arhitektura se koristi za efikasno upravljanje poslovnim logikama, korisničkim interfejsom i podacima.

Model predstavlja sloj koji se bavi poslovnim logikama i podacima. U našem slučaju, model bi uključivao klase koje opisuju objekte kao što su jela, narudžbine, radnici i druge relevantne entitete. Model takođe sadrži sloj pristupa podacima (Data Access Layer) koji omogućava komunikaciju sa bazom podataka i izvršavanje operacija poput čitanja, pisanja i ažuriranja podataka.

View predstavlja sloj koji se bavi prikazom korisničkog interfejsa. Ovde se koristi Thymeleaf tehnologija koja uključuje HTML, CSS I može se koristiti JavaScript kako bi se kreirao vizuelni prikaz podataka korisnicima. U slučaju sistema za upravljanje restoranom, view bi se odnosio na prikaz menija, narudžbina, formi za unos podataka i ostalih interakcija sa korisnicima.

Controller je sloj koji upravlja komunikacijom između Modela i View-a. On obrađuje zahteve korisnika, izvršava odgovarajuće poslovne logike i ažurira Model ili View prema potrebi. Kontroleri u Spring Boot projektu mogu koristiti anotacije kako bi definisali rute, metode za obradu zahteva i komunikaciju sa servisima za pristup podacima.

Kombinacija ovih slojeva - Modela, View-a i Controllera - omogućava modularnost, reusabilnost i olakšava održavanje aplikacije. MVC arhitektura omogućava odvajanje odgovornosti, olakšava timski rad, testiranje i pruža fleksibilnost za buduće promene. U Spring Boot projektu za sistem za upravljanje restoranom, primena MVC arhitekture omogućava organizovanu i skalabilnu razvojnu sredinu za razvoj i održavanje softverskog sistema.

 *Slika 5: Prikaz MVC arhitekture korišćene za ovaj sistem*

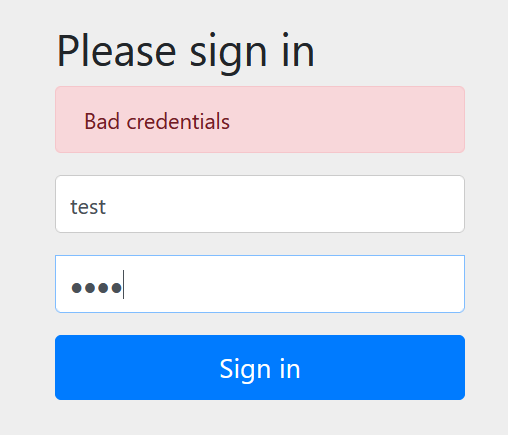
1. **Implementacija**

Za implementaciju sistema za upravljanje restoranom odabran je programski jezik Java, a kao backend framework korišćen je Spring Boot. Arhitektura MVC (Model-View-Controller) primenjena je kako bi se postigla jasna organizacija i odvajanje odgovornosti. Za deo aplikacije koji se odnosi na korisnički interfejs odnosno frontend, korišćen je Thymeleaf, alat za generisanje dinamičkog HTML sadržaja. Primena MVC arhitekture omogućava logičku podelu sistema na tri glavne komponente: Model, koji se bavi poslovnom logikom i obradom podataka; View, koji je zadužen za prikaz korisničkog interfejsa; i Controller, koji upravlja komunikacijom između Modela i View-a. Ceo projekat je smešten na javnom repozitorijumu na GitHubu.

GitHub link: <https://github.com/Djolean/Restautant_POS_System_JavaSpringBoot>



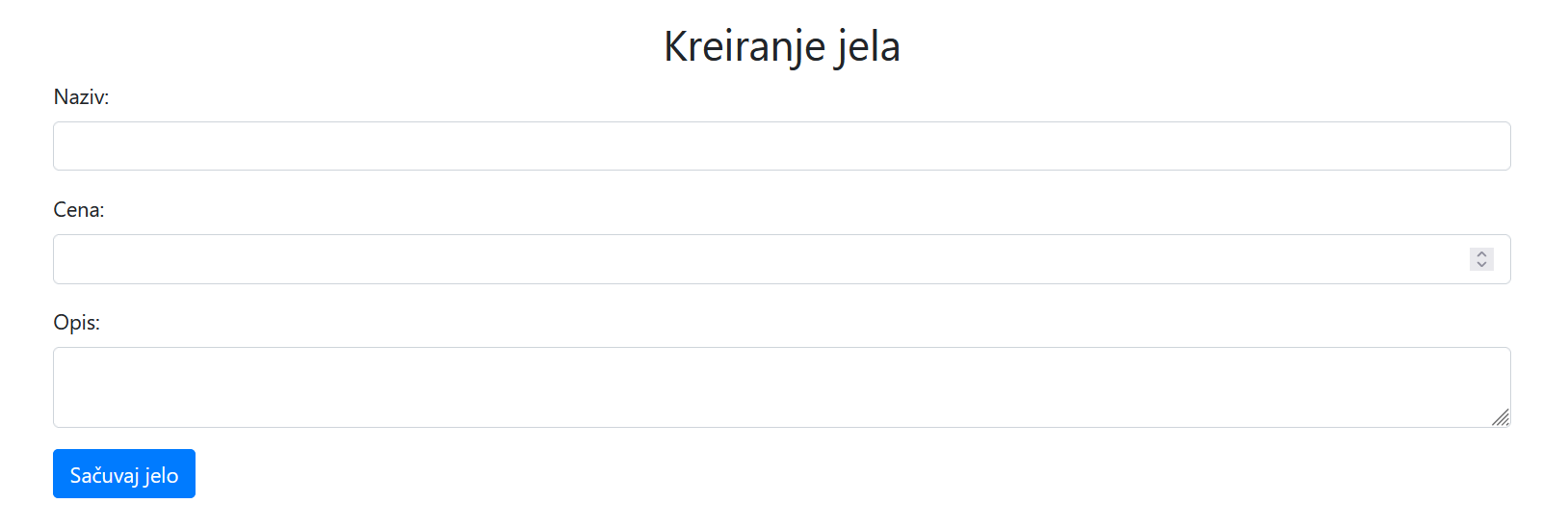
Prva I početna stranica projekta je login, korisnik se prvo mora logovati kako bi nastavio interakciju sa aplikacijom.



Prikaz login stranice prilikom greške pri unošenju podataka.



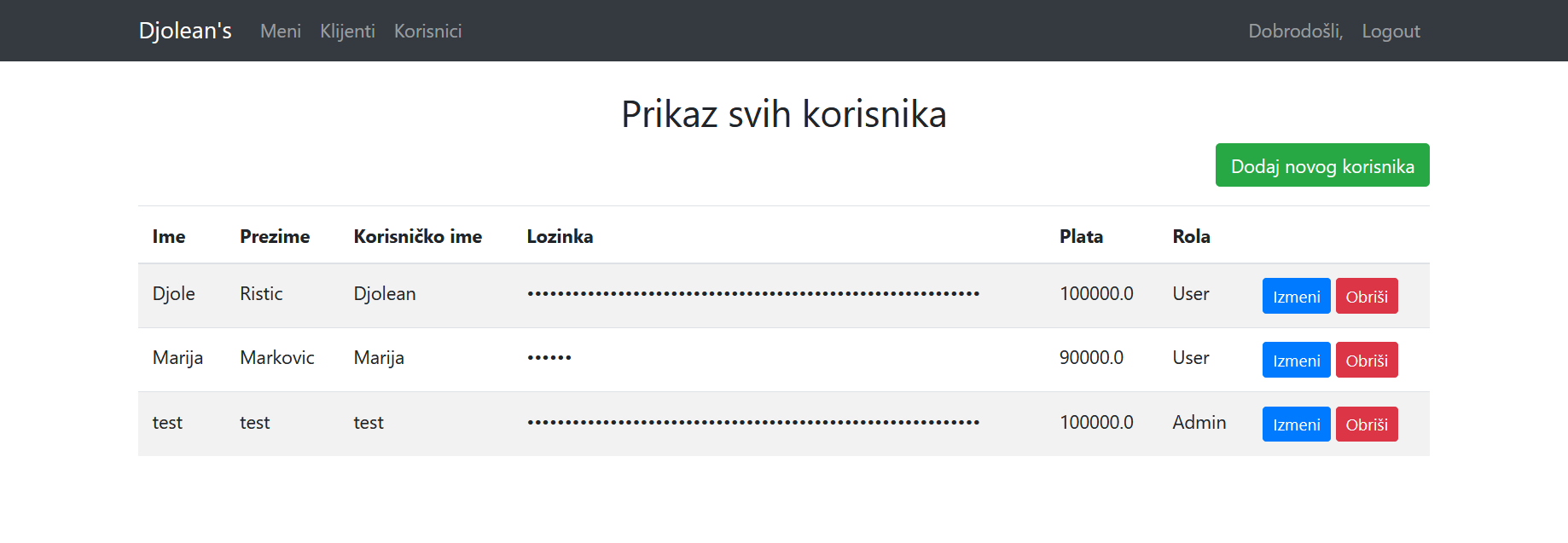
Prva stranica nakon uspešnog logovanja korisnika je stranica koja prikazuje meni gde se nalaze informacije o jelima iz menija.



Stranica za kreiranje novog jela.



Stranica za ažuriranje postojećeg jela iz baze podataka.



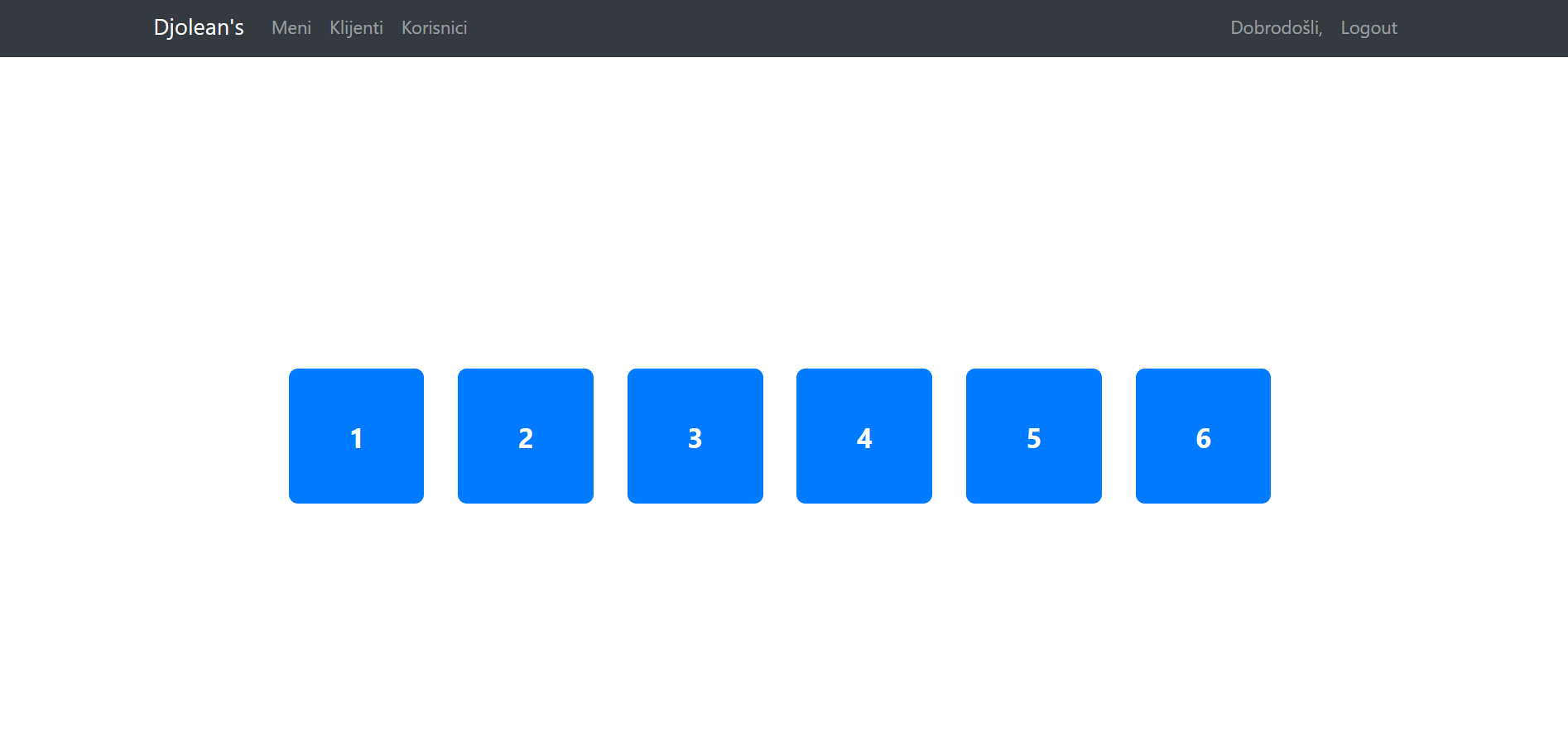
Stranica za prikaz podataka o svim korisnici koju može videte samo korisnik sa rolom “Admin”.



Stranica za kreiranje novog radnika kojoj takođe ima pristup samo korinsik sa rolom “Admin”



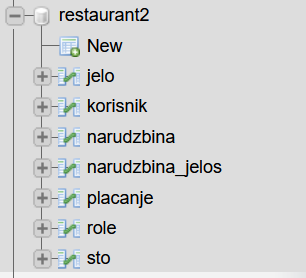
Stranica za izmenu podataka nekog radnika, kojoj takođe ima pristup samo korisnik sa rolom “Admin”.



Klikom na stranicu “Klijenti” otvara se stranica gde postoji prikaz stolova u restoranu. Klikom na neki od stolova sledi sledeća stranica:



Na ovoj stranici se nalazi prikaz nove narudžbine koaj je inicijalno prazna sa leve strane, a sa desne strane se nalazi ponuda iz menija. Klikom na dugme “Dodaj” iz menija, korisnik ubacuje željenu stavku u narudžbinu sa leve strane. Takođe korisnik ima mogućnost klikom na dugme “Delete” da obriše željenu stavku iz narudžbine. Klikom na dugme “Naplati” se ova tabela prazni, a podaci narudžbine se čuvaju u bazi podataka kao I podaci o ukupnoj ceni.



Struktura baze podataka.

1. **Testiranje**

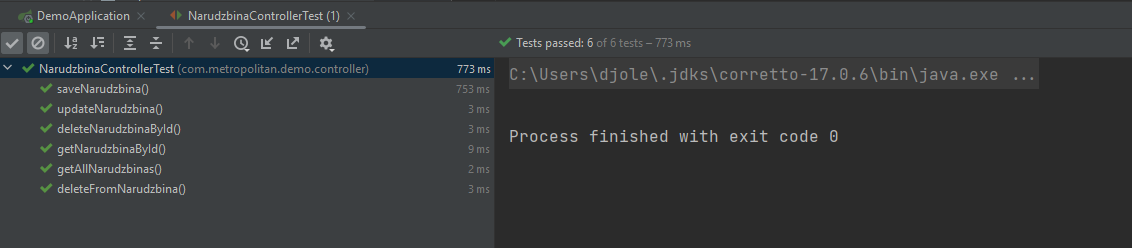
Za prethodni projekat smo se odlučili da kombinujemo 50% black box i 50% white box testiranja kako bismo postigli što potpuniju i efikasniju kontrolu kvaliteta.

Black box testiranje je obuhvatilo testiranje sistema bez poznavanja unutrašnjih detalja i strukture koda. Fokusirali smo se na testiranje funkcionalnosti i mogućnosti sistema kao krajnji korisnici. Ovo nam je omogućilo da identifikujemo bilo kakve probleme ili nedostatke u korisničkom iskustvu i funkcionalnosti.

S druge strane, white box testiranje je obuhvatilo detaljno proučavanje i testiranje internih komponenti sistema. Koristili smo tehniku provere i analizu izvornog koda, kao i vršenje testova jedinica i integracije. Ovo nam je pomoglo da identifikujemo greške u logici, performansama i pouzdanosti sistema.







1. **Zaključak**

Tokom realizacije ovog projekta, demonstrirana su stečena znanja iz predmeta "Web Sistemi 1 - IT255" i "Web Sistemi 2 - IT355". Projekat je predstavljao sveobuhvatan prikaz gradiva obrađenog tokom predavanja i vežbi u ovim kursevima. Postoji još mnogo mogućnosti za poboljšanje projekta kao što su dodavanje crvene boje stola ukoliko je zauzet I podela menija na odvojene sekcije sa pićem, sa hranom itd..

1. **Reference**

[1] IT255 Web Sistemi 1, nastavni materijali za e-učenje, dr Vladimir Milićević, Univerzitet Metropolitan, 2022.

[2] IT355 Web Sistemi 2, nastavni materijali za e-učenje, Jovana Jović, Univerzitet Metropolitan, 2023

[3] Spring Boot -Thymeleaf tutorial, *TUTORIALSPOINT,*

<https://www.tutorialspoint.com/spring_boot/spring_boot_thymeleaf.htm>

[4] Swagger API, *GitHub Repository,*

<https://github.com/swagger-api/swagger-ui>

[5] Spring Documentation, *Spring*,

<https://docs.spring.io/spring-security/reference/servlet/integrations/mvc.html>