Izrada web aplikacije zasnovane na višeslojnoj arhitekturi pomoću razvojnog okvira Spring

Azinić, Andrija

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

Izrada web aplikacije zasnovane na višeslojnoj arhitekturi pomoću Spring razvojnog okvira

Završni rad

Andrija Azinić

Osijek, 2021

SADRŽAJ

1. UVOD

Razvoj web aplikacija se kao jedan od oblika razvoja programske podrške (engl. *software*) neprestano mijenja i napreduje, a brojne arhitekture su predložene za njihovu izradu. Tradicionalno, web aplikacije su, kao i većina drugih oblika programske podrške, primarno zasnovane na monolitnoj arhitekturi. Iako je ovaj tip arhitektura prevladavao u razvoju web aplikacija od početka Interneta pa sljedećih nekoliko desetljeća, danas je on u značajno manjoj uporabi zbog određenih nedostataka. S druge strane, brojne druge arhitekture su predložene posljednjih godina, koje nastoje nadvladati nedostatke monolitne arhitekture te omogućiti razvoj web aplikacija koje je naknadno lakše nadograđivati te dorađivati. Jedna takva arhitektura je višeslojna arhitektura, u kojoj je web aplikacija razdvojena na nekoliko neovisnih slojeva koji razdvajaju njene osnovne funkcionalnosti na skup ponovno upotrebljivih, odnosno općenitih radnji.

U ovom završnom radu nastoji se izraditi web aplikacija zasnovana na višeslojnoj arhitekturi kako bi se demonstrirali izazovi takvog razvoja, ali naglasili i njene prednosti u odnosu na monolitnu arhitekturu. Studija slučaja kojom će se analizirati izazovi i prednosti višeslojne arhitekture zasniva se na razvoju web aplikacije za vođenje ugostiteljskog objekta. Web aplikacija ima funkcije naručivanja jela iz ponuđenog jelovnika te izdavanja narudžbe sa svim bitnim podacima kao što su naručeno jelo, količina, cijena. Razvijanje ove aplikacije se dijeli na razvoj prednjeg sučelja i pozadinske podrške. Prednje sučelje je dio aplikacije koji je zadužen za prikaz podataka u grafičkom obliku pomoću programskih jezika HTML, CSS, JavaScript. Pozadinska podrška se odnosi na dio web stranice koju korisnik ne vidi, odnosno na dio web aplikacije koji se pokreće prije nego što se web stranica prikaže.

U drugom poglavlju izložene su osnove razvoja web aplikacije te su istaknute prednosti višeslojne arhitekture u odnosu na monolitnu arhitekturu. Treće poglavlje opisuje programsko rješenje odnosno tehnologije, platforme i alate koji su korišteni za razvoj. Konačno je dan osvrt na razvijeno programsko rješenje te su dane smjernice za budući rad.

1.1. Zadatak završnog rada

Cilj je ovog završnog rada opisati višeslojnu arhitekturu za izradu web aplikacija te obrazložiti njene prednosti i nedostatke u odnosu na monolitnu arhitekturu. U praktičnom dijelu rada implementirati će se apstrakcija višeslojne arhitekture koristeći Spring razvojni okvir. Razvijena web usluga komunicirat će s okolinom putem REST arhitekture, a za razvoj korisničkog sučelja upotrijebit će se MVC uzorak dizajna. Metodom studije slučaja, model i funkcionalnosti web aplikacije će se definirati sa svrhom izrade web sjedišta ugostiteljskog objekta.

2. WEB APLIKACIJA ZASNOVANA NA VIŠESLOJNOJ ARHITEKTURI

U ovom poglavlju, prikazane su osnove razvoja web aplikacija s posebnim naglaskom na arhitekture programske podrške na kojima se takav razvoj najčešće zasniva. Prvotno je dan opis sustava World wide web koji omogućuje pristup i dijeljenje web aplikacija. Nakon toga je izložen sažeti opis višeslojne arhitekture na kojoj se zasniva web aplikacija razvijena u ovom radu. Konačno, detaljno je opisana specifikacija arhitekture predloženog programskog rješenja koji je razvijen kako bi se istaknule glavne prednosti, ali i izazovi, korištene višeslojne arhitekture za razvoj web aplikacija.

2.1. Pojam i okolina web aplikacije.

World wide web, poznatiji kao Web, je međusobno povezani sustav javnih web stranica dostupnih putem Interneta. Web daje mogućnost pristupa dokumentima i sadržajima na internetu pomoću hipertekstualnih ili hipermedijskih veza. Razvoj sustava weba započeli su 1989. godine Tim Berners-Lee i njegovi kolege iz laboratorija u okviru međunarodne organizacije CERN. Kreirali su protokol HTTP (engl. *HyperText Transfer Protocol*), koji predstavlja današnji standard za komunikaciju između poslužitelja i klijenata. Njihov tekstualni web preglednik je bio dostupan u siječnju 1992. godine. Hipertekstualni dokument s tekstom i hipervezama napisan je u opisnom jeziku HTML(engl. *HyperText Markup Language*), a dodijeljena mu je mrežna adresa koja se naziva URL (engl. *Uniform Resource Locator*). Hipertekst omogućuje korisniku odabir riječi ili fraze iz teksta i time pristup drugim dokumentima koji sadrže dodatne informacije koji se odnose na tu riječ ili frazu [1].

Web stranica je resurs na sustavu World wide web. Web stranica prikazuje određeni statički sadržaj te daje određene korisne informacije u obliku teksta. Statički sadržaj koji je prikazan na tim stranicama je u pravilu definiran pomoću opisnog jezika HTML i stilskog jezika CSS (engl. *Cascading Sytle Sheets*). Web aplikacija je programska podrška kojoj se također pristupa preko sustava World wide web te se nalazi na poslužiteljskoj strani. Glavne zadaća web stranice su autorizacija i autentikacija, interakcija s korisnikom, odrađivanje poslovne logike i druge [2]. Web aplikaciji je za rad potreban poslužitelj i baza podataka. Web poslužitelj je zadužen za upravljanje

zahtjevima koji su došli od strane klijenta. Baza podataka sadrži sve podatke karakteristične za određenu stranicu. Podaci koji se razmjenjuju između poslužitelja i preglednika su najčešće formatirani nekim od formata podataka poput JSON (engl. *JavaScript Object Notation*) te XML (engl. *Extensible Markup Language*). Format JSON temelji se na programskom jeziku JavaScript, koristi se za razmjenu podataka i neovisan je o programskom jeziku. Format XML je dizajniran za prijenos podataka, a ne za prikaz podataka što je preporuka konzorcija W3C (engl. *World wide web Consortium*). To je format tekstualnih podataka, koristi podršku formata Unicode za različite jezike [3].

Najčešći arhitekturni stil za pružanje standarda za međusobnu komunikaciju računalnih sustava na sustavu WWW jest REST (engl. *Representational State Transfer*). Arhitekturni stil REST zasniva na protokolu HTTP te koristi HTTP operacije za sastavljanje zahtjeva za resursom. API je stil sučelja aplikacijskog sloja koji koristi HTTP zahtjeve za pristup i upotrebu podataka. Zahtjevi se dijele na operacije *GET, PUT, POST i DELETE*, koje se odnose na čitanje, ažuriranje, stvaranje i brisanje operacija koje se tiču resursa.

U procesu razvijanja web aplikacija radimo podjelu na prednje sučelje (engl. *frontend*) i pozadinsku podršku (engl. *backend*) aplikacije. Pri tome obje komponente moraju surađivati jedna s drugom kako bi imali funkcionalnu stranicu. Prednje sučelje je dio web aplikacije s kojim je klijent u kontaktu, odnosno to je grafički prikaz svih podataka stranice poput boje i stilove teksta, slike, grafikone i tablice, gumbe, boje i navigacijski izbornik. Opisni jezici korišteni za razvoj prednjeg sučelja su HTML, CSS i JavaScript. Jedan od važnijih dijelova prednjeg sučelja su responzivnost i brzina prikaza stvari na stranici. Responzivnost se odnosi na to da stranica bude prikazana u dobrom formatu na različitim uređajima, koji imaju različite rezolucije. Opisni jezik HTML kombinira hipertekst s opisnim jezikom, pri čemu hipertekst definira vezu između stranica, a opisni jezik se koristi za definiranje tekstualne dokumentacije unutar oznake koja definira strukturu web stranica. CSS omogućuje primjenu stilova na web stranicama i to neovisno o statičkom sadržaju koji čini svaku web stranicu. Programski jezik JavaScript se koristi kako bi stranica postala interaktivna za korisnika. Koristi se za poboljšanje funkcionalnosti web mjesta za pokretanje igara i softvera temeljenog na webu.

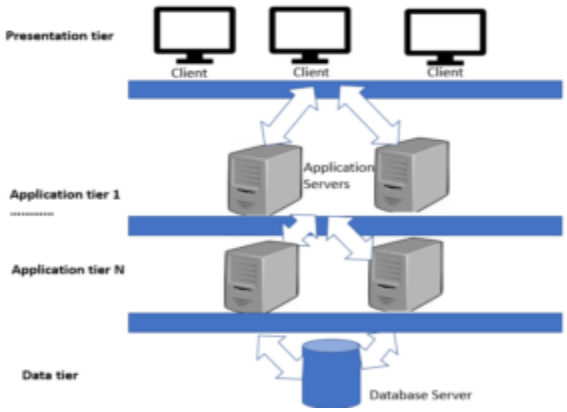
Pozadinska podrška je dio web aplikacije kojeg ne vidimo, a odgovoran je za pohranjivanje podataka i isporuku istih. To je dio programa s kojim klijent ne dolazi u kontakt. Svim

komponentama pozadinske podrške pristupa se preko prednjeg sučelja. U pozadinsku podršku spadaju funkcije, pisanje API poziva sve CRUD (engl. *Create, Read, Update and Delete*) operacije koje su zaslužne za spremanje, brisanje, ažuriranje baze podataka. Neki od najčešćih programskih jezika za razvoj pozadinske podrške su: PHP, C++, Java, Python, JavaScript [4].

Jedan od novijih trendova pri razvoju web aplikacija jest progresivna web aplikacija (*engl. progressive web application,* PWA). Nju odlikuju visoke razine performansi bez obzira na stanje uređaja ili mreže. Neke od prednosti progresivne web aplikacije su: neovisnost povezivanja, push obavijesti, samo ažuriranja, sigurnost. Iako prima puno pažnje, koncept PWA relativno je nov za većinu ljudi [5].

2.2. Višeslojna arhitektura

Višeslojna softverska arhitektura jedna je od najpopularnijih arhitektura za razvoj programske podrške. Sastoji se od više različitih slojeva, od kojih svaki odgovara različitoj usluzi. Budući da je svaki sloj neovisan o drugim slojevima, izmjene na sloju je lakše činiti nego na cjelokupnoj arhitekturi. Glavni slojevi u višeslojnoj arhitekturi su prezentacijski sloj, aplikacijski sloj i podatkovni sloj. Prezentacijski sloj je najviši sloj u aplikaciji, a služi za prezentaciju sadržaja korisniku putem grafičkog sučelja. Da bi se prezentirao sadržaj, neophodno je da ovaj sloj komunicira s ostalim slojevima aplikacije. Aplikacijski sloj je srednja razina arhitekture, a odgovoran je za poslovnu logiku aplikacije. Poslovna logika je skup pravila koji implementiraju funkcionalne zahtjeve aplikacije. Komponente ove razine izvode se na jednom ili više poslužitelja. Podatkovni sloj je najvažniji sloj ove arhitekture jer se bavi pohranom i pronalaženjem podataka aplikacije. Podaci aplikacije se obično pohranjuju na poslužitelju baze podataka, poslužitelju datoteka ili bilo kojem uređaju koji podržava logiku pristupa podacima. To čini razina podataka pružanjem API-ja razini aplikacije. Odredba ovog API-ja osigurava potpunu transparentnost operacija podataka koje se obavljaju u ovoj razini, bez utjecaja na razinu aplikacije. Na primjer, ažuriranja ili nadogradnje sustava u ovoj razini ne utječu na razinu aplikacije ove arhitekture. Višeslojna aplikacija se ne sastoji samo od 3 sloja. Višeslojna arhitektura uvijek ima prezentacijski sloj i sloj podataka, ali na njih se mogu graditi i drugi slojevi, što je prikazano slikom 2.1. Prednosti ove arhitekture su to što olakšava ažuriranje bilo kakvih naslijeđenih sustava. Promjene koje treba izvršiti trebaju biti jednostavnije i manje opsežne nego što bi inače morale biti [6].



Sl. 2.1. Skica višeslojne arhitekture

Dugo vremena je tradicionalna arhitektura programske podrške bila monolitna arhitektura, prema kojoj sve komponente web aplikacije koegzistiraju kao jedan modul. Prednost monolitne aplikacije je jednostavan razvoj jer se sve nalazi u jednom programskom okruženju, a najčešće jedan tim radi na jednoj web aplikaciji. Poslužiteljski sustav je baziran na jednoj aplikaciji. Jedan od glavnih nedostataka takve arhitekture je što svaki dio aplikacije ovisi o nekom drugom dijelu. Drugim riječima, promjena jednog dijela programskog koda utječe na druge komponente, koje je također potrebno prilagoditi da bi web aplikacija radila. Stoga su programeri ograničeni na korištenje samo jednog programskog jezika. Buduća nadogradnja te aplikacije može biti problem, ako klijent dođe s novim zahtjevima koje je potrebno implementirati. Tako aplikacija postaje sve veća i teže se snalaziti u njoj. Još jedan od nedostataka je ako postoji problem s određenim implementiranim dijelom aplikacije koji ne radi. Potrebno je taj dio aplikacije popraviti i aplikaciju cijelu ugasiti za to vrijeme korisnici nemaju pristup aplikaciji.

2.3. Prikaz stanja u području

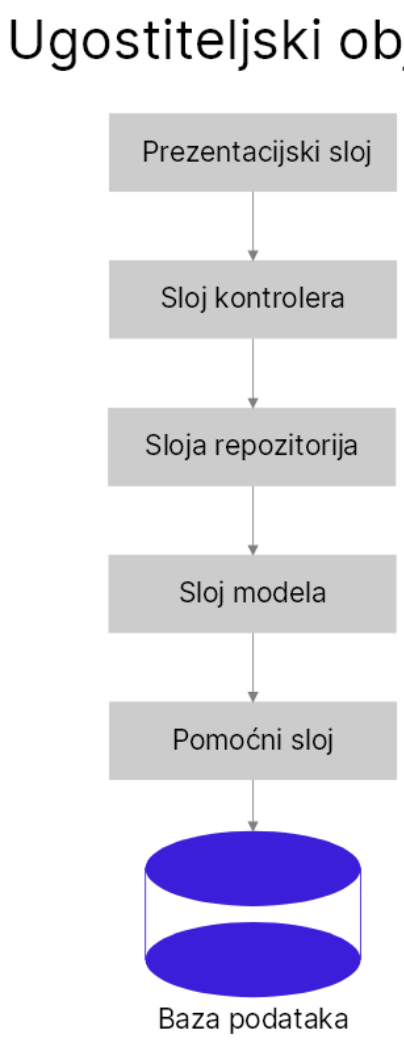
Višeslojna arhitektura u literaturi je korištena za potrebe razvoja programske podrške u brojnim područjima primjerne. Primjerice, Grycuk et al. su u [7] predložili programsko rješenje u obliku višeslojne arhitekture za pronalaženje slika na temelju sadržaja. Koriste višeslojnu arhitekturu prvenstveno radi njene skalabilnosti. Takva arhitektura omogućuje im zamjene bilo komponente bez utjecaja na ostale. Također, Yigui et al. koriste višeslojnu arhitekturu u [8] za razvoj sustava

za praćenje akademske izvrsnosti. Pri tome se oslanjaju na uporabu klasičnog troslojnog uzorka višeslojne arhitekture, koji sadrži sloj pristupa podacima, sloj poslovne logike te prezentacijski sloj. Nadalje, Lina u [9] rabi višeslojnu arhitekturu za izradu aplikacije za e-učenje. Pri tome, aplikaciju dijeli na prezentacijski sloj, pomoćni sloj, sloj kontrolera te sloj modela. Za izradu ove aplikacije koristi razvojne okvire Spring, Struts i Hibernate. Konačno, M. Bertocco u [10] zasnivaju razvoj sustava za raspodijeljeno prikupljanje podataka na višeslojnoj arhitekturi koja omogućava skaliranje u slučaju velikog broja izvora podataka. Višeslojna arhitektura je u literaturi korištena i u brojnim drugim područjima primjene, a većina takvih radova naglašava njenu skalabilnost i jednostavnu integraciju s ostalom programskom podrškom kao ključne prednosti prilikom odabira takve arhitekture za razvoj web aplikacija.

2.4. Specifikacija arhitekture predloženog programskog rješenja

Višeslojna arhitektura može biti osnova za razvoj web aplikacija u raznim područjima primjene. Kako bi se demonstrirali izazovi prilikom njezina uspostavljanja, kao i prednosti koje ona donosi, ova arhitektura je korištena kao osnova za razvoj web aplikacija za potrebe ugostiteljskog objekta. U nastavku je dan opis specifikacija te arhitekture, čija je shema prikazana na slici 2.2.

Web aplikacija za potrebe vođenja ugostiteljskog objekta se sastoji od više zasebnih slojeva koji su neovisni jedan o drugom. Slojevi se međusobno pozivaju pomoću ovisnosti koje su karakteristične za module razvijene u programskom jeziku Java.



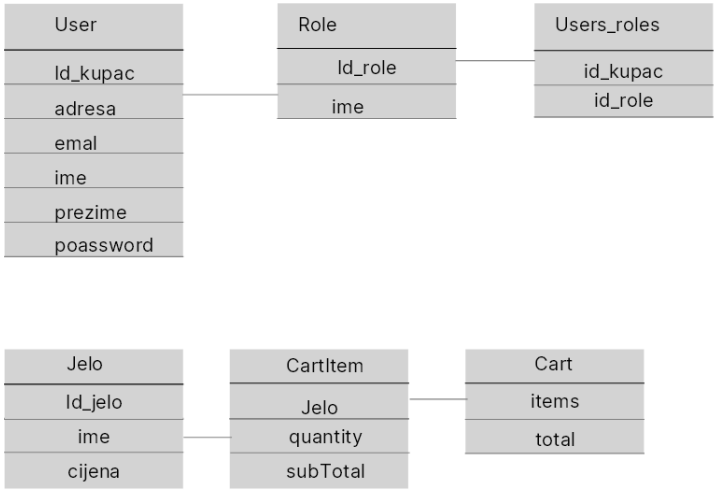
Sl. 2.2. Višeslojna arhitektura ugostiteljskog objekta

Prema konceptualnoj podjeli, arhitektura programskog rješenja se sastoji od pet dijelova, odnosno slojeva (prezentacijski sloj, sloj kontrolera, sloj repozitorija, sloj modela i pomoćni sloj). Korisnik web aplikacije pristupa aplikaciji putem prezentacijskog sloja. Prezentacijski sloj se prvenstveno sastoji od HTML dokumenata u kojem se nalazi sadržaj koji se prikazuje na prednjem sučelju te CSS dokumenata koji služe za stiliziranje tog sadržaja. S obzirom na to da je namjena web aplikacije vođenje ugostiteljskog objekta, sučelja koja se prikazuju korisniku omogućuju mu prikaz glavne stranice objekta, prikaz jelovnika, stvaranje narudžbe te autentifikaciju. HTML dokumenti u kojima je kodiran naveden sadržaj su redom dokumenti Home, Jelovnik, Narudžba, Login/Register.

Pri razvoju korisničkog sučelja koristiti će se obrazac MVC (engl. *Model-View-Controller*). Obrazac MVC koristi se za implementaciju korisničkih sučelja, podataka i upravljačke logike. Naglasak je na podjeli između poslovne logike, prikaza i sloja kontrolera. Sloj kontrolera je jedan od glavnih dijelova MVC-a, pri čemu sloj kontrolera zaprima zahtjeve prema aplikaciji i upravljaju tim zahtjevima pomoću metoda. Sloj Kontrolera ne pohranjuje i ne sadrži podatke već procesira zahtjeve. Akcijske metode unutar klase sloja kontrolera primaju zahtjeve i pozivaju se pomoću zasebnih URL-a. Sloj Kontrolera djeluje kao sučelje između sloj modela i prezentacijskog sloja. Sučelja aplikacije poput Jelovnik, Narudžba, Login/Register koriste svoje zasebne slojeve kontrolera koji služe za obradu zahtjeva korisnika ovisno o HTML dokumentu na kojem se korisnik nalazi i URL-u kojem korisnik pristupa.

Za logiku pristupa podacima u web aplikaciji zaslužan je sloj repozitorija, koji se koristi za (*Create, Read, Update and Delete*) operacije. Operacije poput punjenja, ažuriranja, brisanja podataka u bazi implementirane su pomoću okvira za objektno-relacijsko mapiranje Hibernate. CRUD operacije su bitne za rad web aplikacije, jer omogućavaju lakše rukovanje s bazom podataka i manipuliranje podacima.

Model je jedan od ključnih elemenata web aplikacije, jer su u njemu definirane sve klase s karakterističnim atributima za tu klasu, odnosno model sadrži podatke aplikacije. Definirane klase su Cart, CartItem, Jelo, Role, User, Users_roles, što je prikazano na slici 2.3. Svaki navedeni objekt koristi svojoj svrsi i potrebni su za rad aplikacije i spremanje podataka.



Sl. 2.3. Shema baze podataka

Povezivanje s bazom podataka i definirane funkcije za rad s bazom podataka se nalaze u pomoćnom sloju. Definirane funkcije služe za razne mogućnosti kao što su brisanje jela, dodavanje novog kupca putem registracije, autoriziranje kupca kroz Login.

3. PROGRAMSKO RJEŠENJE: WEB SJEDIŠTE UGOSTITELJSKOG OBJEKTA

U ovom poglavlju prikazan je opis platformi tehnologija i alata, zatim razvoj web aplikacije, a potom analiza web aplikacije. Prvenstveno je opisan razvoji okvir Spring pomoću kojeg razvijamo web aplikaciju, te ostale tehnologe i alati poput okvira za objektno-relacijsko mapiranje Hibernate, relacijske baza podataka MySQL i alata za izradu predložaka Thymeleaf. Nakon toga je detaljno opisan razvoj web aplikacije i komponente višeslojne arhitekture. Konačno, opisane su dvije funkcionalnosti aplikacije s kojima se služi korisnik kao bi se registrirao i napravio narudžbu.

3.1. Opis platformi tehnologija i alata

Prilikom razvoja web aplikacije koriste se razne tehnologije, s posebnim fokusom na razvojni okvir Spring koji omogućuje razvoj web aplikacija prema obrascu MVC. Osim toga, koriste se i okvir za objektno-relacijsko mapiranje Hibernate, baza podataka te alat za izradu predložaka Thymeleaf.

3.1.1. Razvojni okvir Spring

Spring razvojni okvir je otvorenog koda (engl. *open source*) koji pruža podršku za razvoj Java aplikacija. Namjena Spring razvojnog okvira je to što se koristi za izradu Java aplikacija, a postoje i ekstenzije koje se koriste za izradu web aplikacija na Java EE (*engl. Java Enterprise Edition*) platformi. Spring omogućuje jednostavno razvijanje web aplikacija, pri čemu nije potrebno koristiti XML konfiguraciju u Springu. Također ga karakterizira lakoća razvoja web aplikacija. Osim toga, pruža fleksibilan način konfiguriranja Java Beans klasa koje enkapsuliraju jedan ili više objekata u jedan standardizirani objekt (bean) [7]. Omogućuje stvaranje samostalnih aplikacija, pri čemu se ne oslanja na vanjski poslužitelj već koristi svoj ugrađeni web poslužitelj Tomcat ili Netty. U ovom završnom radu koristio se Spring Boot za konfiguriranje aplikacije zasnovane na razvojnom okviru Spring. Spring Boot je u osnovi proširenje Spring okvira, koji eliminira standardne konfiguracije potrebne za postavljanje Spring aplikacije [11].

3.1.2. Okvir za objektno-relacijsko mapiranje Hibernate

Konkretno, Hibernate ORM (engl. *Object-Relational Mapping*) se bavi postojanošću podataka, što se odnosi na relacijske baze podataka RDBMS (engl. *Relational database management system*). Hibernate omogućuje mapiranje Java klasa u tablice baze podataka koristeći XML datoteke, bez pisanja koda. Pruža jednostavne API-je za pohranu i dohvaćanje Java objekata izravno u i iz baze podataka. Ako dođe do promjena u bazi podataka potrebno je samo izmijeniti XML datoteku [12].

3.1.3. Relacijska baza podataka MySQL

MySQL je sustav upravljanja relacijskim bazama podataka RDBMS. Baza podataka je strukturirana zbirka podataka koja može sadržavati različite tipove podataka. Relacijska baza podataka prikuplja podatke te ih organizira prema relacijskom modelu. MySQL baza podataka sadrži zapise u više zasebnih tablica, za razliku od jednog sveobuhvatnog spremišta. To omogućuje relacijskoj bazi podataka bolje ažuriranje informacija, dohvaćanje podataka ili složenije radnje. MySQL je sastavni dio mnogih popularnih aplikacija zbog otvorenog koda, stabilnosti i bogatih značajki [12].

3.1.4. Alat za izradu predložaka Thymeleaf

Thymeleaf je alat za obradu i stvaranje sadržaja kreiranog pomoću opisnog jezika HTML, stilskog jezika CSS te programskog jezika Javascript. Najčešće se koristi za generiranje HTML pogleda. Thymeleaf je najčešće HTML stranica s Thymeleaf sintaksom koja se koristi za povezivanje sloja kontrolera s pogledom [14].

3.2. Razvoj Web aplikacije

U poglavlju razvoj web aplikacije za potrebe ugostiteljskog objekta, izloženi su detaljniji opisi pojedinih komponenata višeslojne arhitekture, počevši od baze podataka pa sve do prezentacijskog sloja. Svako poglavlje opisuje zasebnu komponentu višeslojne arhitekture koje sadrži sliku s detaljnim opisom slike.

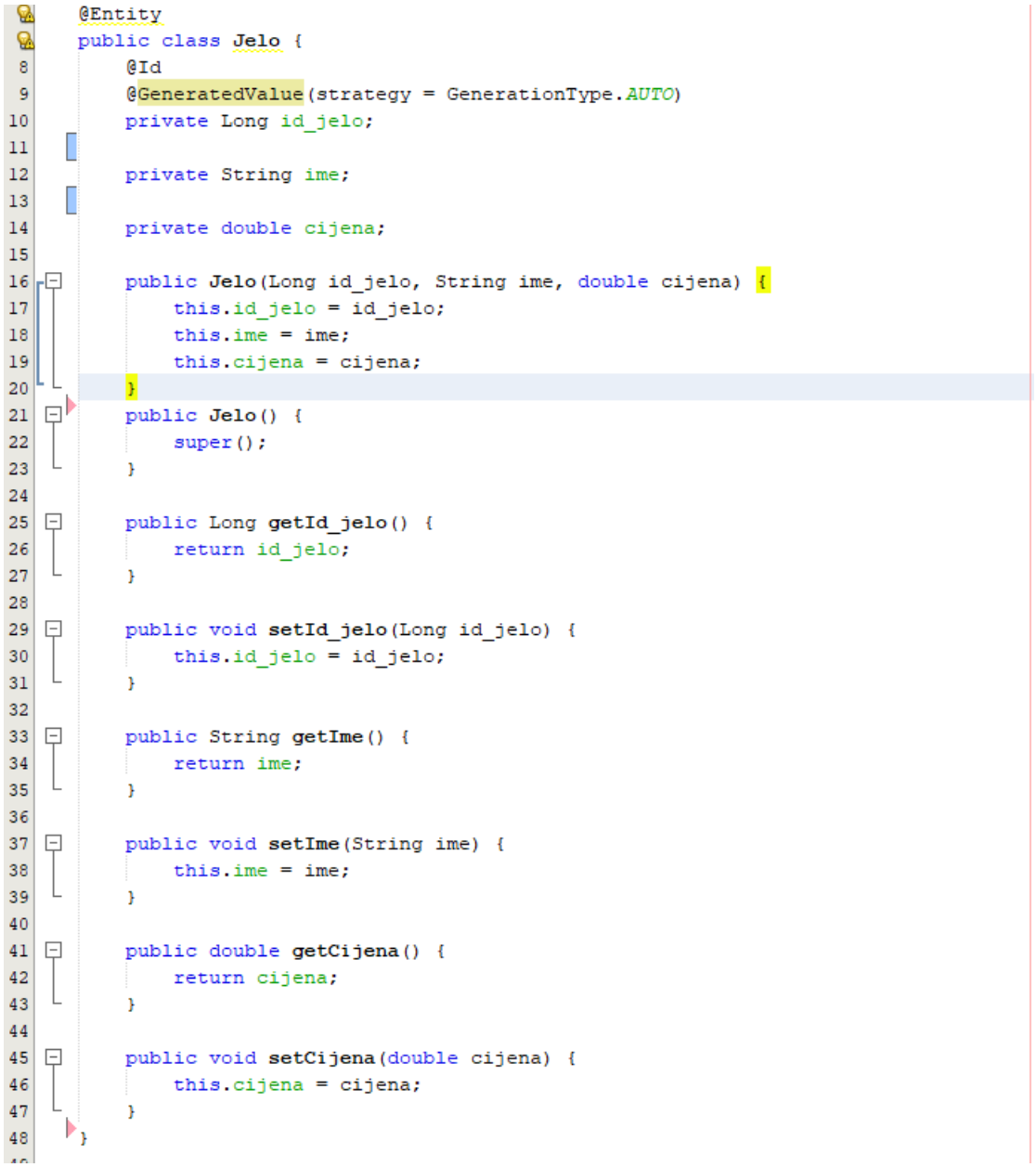
3.2.1. Baza podataka

U web aplikaciji korištena je MySql baza podataka, u bazi se nalaze 4 tablice: role, user, user roles, jelo, kupac. Pomoću tehnologije Hibernate omogućeno je spajanje na poslužitelj MariaDB server. Hibernate se nalazi u JPA (*Java Persistence API*). JPA specifikacija eksplicitno definira objektno-relacijsko mapiranje. JPA standardizira važan zadatak objektno-relacijskog mapiranja korištenjem napomena ili XML-a za mapiranje objekata u jednu ili više tablica baze podataka.

3.2.2. Sloj modela

Prema uzorku dizajna MVC svi modeli predstavljaju klase u programskom jeziku Java. Primjer jednog takvog sloja modela prikazan je na slici 3. U modelu Jelo definirani su karakteristični atributi Id_jelo, ime i cijena. Klasa Jelo označena je sa *@Entity*, što pokazuje da je JPA entitet, odnosno da entitet ili klasa Jelo ima pridruženu tablicu u relacijskoj bazi podataka. Svojstvo id_jelo objekta Jelo označeno je sa @Id tako da ga JPA prepoznaje kao ID objekta. Svojstvo id_jelo također je označeno sa *@GeneratedValue* kako bi se naznačilo da se ID treba automatski generirati. Druga dva svojstva, ime i cijena, ostaju bez napomena. Pretpostavlja se da su mapirani u stupce koji dijele ista imena kao i sama svojstva.

Konstruktor u Javi je posebna metoda koja se koristi za inicijalizaciju objekata. Konstruktor se poziva kada se kreira objekt klase. Može se koristiti za postavljanje početnih vrijednosti za atribute objekta: id_jelo, ime i cijena. Za svaki od atributa su kreirane funkcije za dohvaćanje (*engl. getter*) te funkcije za postavljanje (*engl. setter*) radi lakšeg pristupanja podacima van klase.



```java
@Entity
public class Jelo {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id_jelo;

    private String ime;

    private double cijena;

    public Jelo(Long id_jelo, String ime, double cijena) {
        this.id_jelo = id_jelo;
        this.ime = ime;
        this.cijena = cijena;
    }
    public Jelo() {
        super();
    }

    public Long getId_jelo() {
        return id_jelo;
    }

    public void setId_jelo(Long id_jelo) {
        this.id_jelo = id_jelo;
    }

    public String getIme() {
        return ime;
    }

    public void setIme(String ime) {
        this.ime = ime;
    }

    public double getCijena() {
        return cijena;
    }

    public void setCijena(double cijena) {
        this.cijena = cijena;
    }
}
```

Sl. 3. Primjer implementacije sloja modela Jelo

3.2.3. Sloj repozitorija

Sloj repozitorija zadužen je za sve CRUD operacije koje je potrebno vršiti na modelu. JeloRepository nasljeđuje JpaRepository (JPA) koji prima kao argumente Model (Jelo) i tip ID-a (*Long*). JpaRepository je JPA ekstenzija za repozitorij koja sadrži već postojeće CRUD operacije u razvojnom okruženju Spring. JPA se sastoji od dva dijela: podsustava za mapiranje klasa u

relacijske tablice, kao i *EntityManager* API za pristup objektima, definiranje i izvršavanje upita i još mnogo toga.

```
9   public interface JeloRepository extends JpaRepository<Jelo, Long>{
10
11  }
```

Sl. 3.1. Primjer sučelja JeloRepository

3.2.4. Pomoćni sloj

Pomoćni sloj služi za definiranje potrebnih funkcija za rad s bazom podataka. U slučaju za model Jelo potrebno je iz liste Jelo povući sve potrebne podatke o Jelima id_jela, ime i cijenu. Funkcija koja to radi zove se *getAllJelo()*, prikazana u odlomku 3.2.4 Implementacija pomoćnog sloja.

```
7   @Component
    public interface JeloService {
9
        List<Jelo> getAllJelo();
1
2
3   }
```

Sl. 3.2. Primjer sučelja JeloService

3.2.5. Implementacija pomoćnog sloja

Implementacija pomoćnog sloja Jelo koristi inicijalizaciju jeloRepository kako bi pristupili svim potrebnim operacijama s kojim povlačimo podatke iz baze podataka. Funkcija *getAllJelo()* je naslijeđena funkcija od JeloService (pomoćnog sloja) u kojoj je definirano da vraća sva jela koja se nalaze u bazi podataka pomoću sučelja *JeloRepository* koji sadrži funkciju *findAll()* funkciju od strane JPA, kao što je prikazano na slici 3.3.

```java
@Service
public class JeloServiceImpl implements JeloService {

	private JeloRepository jeloRepository;



	public JeloServiceImpl(JeloRepository jeloRepository) {
		super();
		this.jeloRepository = jeloRepository;
	}

	@Override
	public List<Jelo> getAllJelo() {
		return jeloRepository.findAll();
	}
}
```

Sl. 3.3. Primjer implementacije JeloService

3.2.6. Sloj kontrolera

Kontroler je odgovoran za logiku aplikacije i djeluje kako koordinator između sloja modela i pogleda. Kontroler prima ulaz od strane korisnika putem pogleda i obrađuje podatke pomoću sloja modela i vraća ih na pogled. Primjerice, kontroler jelovnik ima tri funkcije, funkciju koja omogućava otvaranje početne stranice web aplikacije, login funkciju koja omogućava autorizaciju i prijavu na web aplikaciju i funkcija koja povlači sva postojana jela iz baze i prikazuje ih na stranci jelovnik. Svaki od kontrolera ima anotaciju *@RequestMapping* koja mapira HTTP zahtjeve na *handler* metode od MVC-a i REST kontrolera, slika 3.4. *Handler* metode omogućuju pristup parametrima metode, povratnim vrijednostima metode i bilješkama metode.

```
15
16    @Controller
17    public class JelovnikController {
18
19            private JeloService jeloService;
20
21    public JelovnikController(JeloService jeloService) {
22                super();
23                this.jeloService = jeloService;
24        }
25        @RequestMapping(value = "/home")
26        public String openindex(){
27            return "home";
28        }
29           @RequestMapping(value = "/login")
30        public String openLogin(){
31            return "login";
32        }
33        @GetMapping("/jelovnik")
34        public String listJelo(Model model) {
35                model.addAttribute("jela", jeloService.getAllJelo());
36                return "jelovnik";
37        }
38    }
```

Sl. 3.4. Primjer implementacije sloja kontrolera Jelovnik

3.2.7. Prezentacijski sloj

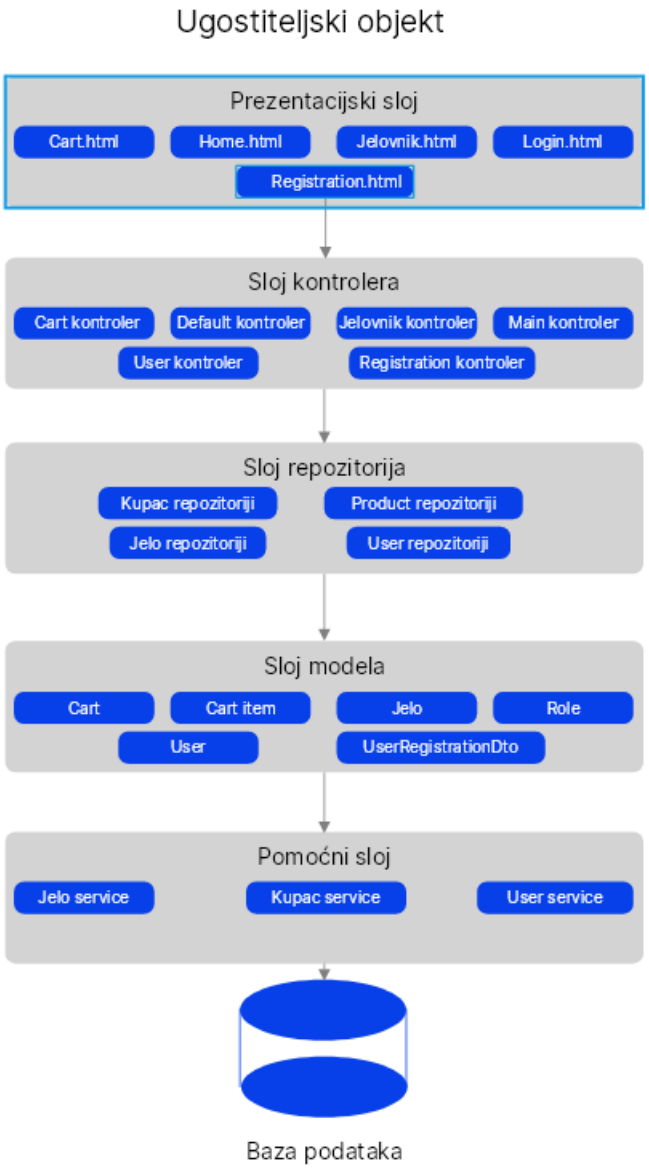
HTML stranica jelovnik prikazuje sve potrebne podatke o jelima i mogućnost naručivanja jela. Za prikaz podatka iz baze koristi se razvojni okvir Thymeleaf pomoću kojeg možemo pristupati potrebnim slojevima kontrolera i nakon toga prikazati potrebne podatke. HTML stranca jelovnik prikazuje tablicu s ponuđenim jelima i podatke o jelima (ime i cijenu), te gumb koji služi za naručivanje jela. Na slici 3.5 je vidljiva HTML datoteka jelovnika. Linija 50 prikazuje sva jela koja se nalaze u listi jela pomoću atributa th:each koji je dio Thymeleafa. Također atributi th:text na linijama 51, 52 pružaju ispis pojedinog imena i cijena za svako jelo. Atribut th:action se postavlja na formu kako bi točno odredili URL na koji se prosljeđuje forma, dok th:value nam omogućava pristup ID-u jela. Također se koristi POST metoda koja nam omogućava slanje podataka na poslužitelj za stvaranje i ažuriranje.

Sl. 3.5. Primjer implementacije sučelja Jelovnik

3.3. Analiza programskog rješenja višeslojne arhitekture

U ovom poglavlju je izložen cjelokupni opis predloženog programskog rješenja. Prema uzoru na standardni model višeslojne arhitekture web aplikacija je podijeljena na slojeve: prezentacijski sloj, sloj kontrolera, sloj repozitorija, sloj modela i pomoćni sloj. Svaki od slojeva sadrži svoje pripadajuće klase s kojima sloj raspolaže. Prezentacijski sloj se prvenstveno sastoji od HTML dokumenata u kojem se nalazi sadržaj koji se prikazuje na prednjem sučelju. S obzirom na to da je namjena web aplikacije vođenje ugostiteljskog objekta, sučelja koja se prikazuju korisniku omogućuju mu prikaz glavne stranice objekta, prikaz jelovnika, stvaranje narudžbe te autentifikacija. Nadalje, kontroleri djeluju kao sučelje između sloja modela te prezentacijskog sloja, zaprimajući zahtjeve prema aplikaciji i upravljajući tim zahtjevima pomoću metoda. Akcijske metode unutar klase kontrolera primaju zahtjeve i pozivaju se pomoću zasebnih URL-a. Za logiku pristupa podacima u web aplikaciji zaslužan je sloj repozitorija, koji se koristi za CRUD operacije. Povrh toga, model je jedan od ključnih elemenata web aplikacije jer su u njemu definirane sve klase s karakterističnim atributima za tu aplikaciju. Definirane klase su Cart, CartItem, Jelo, Role, User i UserRegistrationDto. Povezivanje s bazom podataka te definirane funkcije za rad s bazom podataka se nalaze u pomoćnom sloju. One omogućuju razne mogućnosti

poput brisanja jela, dodavanja novog kupca putem registracije te autoriziranje kupca kroz sučelje Login. Slika 3.6 pobliže opisuje višeslojnu arhitekturu ugostiteljskog objekta i pripadajuće klase.



Sl. 3.6. Višeslojna arhitektura ugostiteljskog objekta i pripadajuće klase

3.4. Analiza korištenja web aplikacije

Nakon razvoja aplikacije za potrebe vođenja ugostiteljskog objekta, prikazane su dvije njene osnovne funkcionalnosti. Kao prvo, dan je primjer funkcionalnosti stvaranja narudžbe koji omogućuje odabir željenog jela te dodavanje istog u narudžbu. Kao drugo, izložen je primjer

registracije korisnika koji se zasniva na popunjavanju osobnih podataka te autentikaciji korisnika kao bi pristupio web aplikaciji za potrebe vođenja ugostiteljskog objekta.

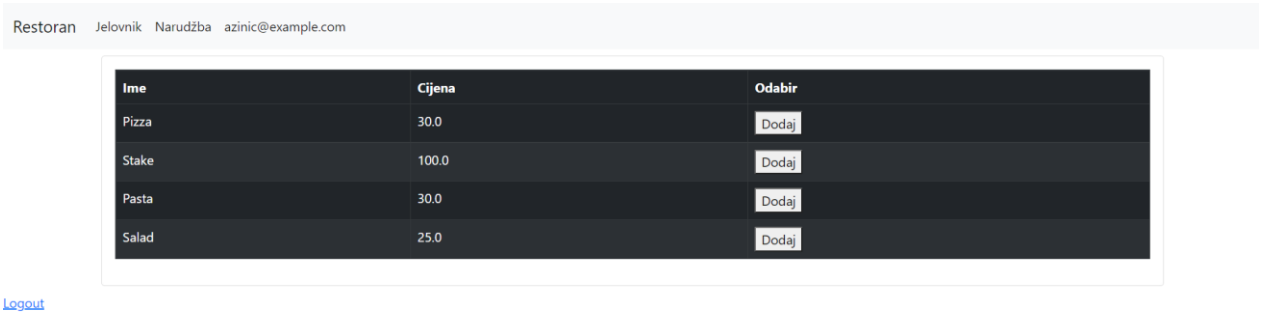
3.4.1. Primjer stvaranja narudžbe

Primjer stvaranja narudžbe je opisan slikom 3.7 gdje se prikazuje red izvođenja funkcija pri kreiranju narudžba. Kako bi kreirali narudžbu potrebo je otvoriti HTML stranicu jelovnik na kojoj se nalazi sav popis jela te kliknuti na gumb dodaj koji poziva Cart kontroler. Cart kontroler sadrži funkciju pod nazivom add, koja omogućuje dodavanje odabranog jela pomoću ID-a jela, slika 3.8. Nakon dodavanja jela aplikacija nas preusmjerava na HTML stranicu narudžba gdje možemo vidjeti detalje narudžbe, ukupnu cijenu i opciju brisanja jela iz narudžbe. Sav postupak kreiranja opisan je slikama 3.9 i 3.10.

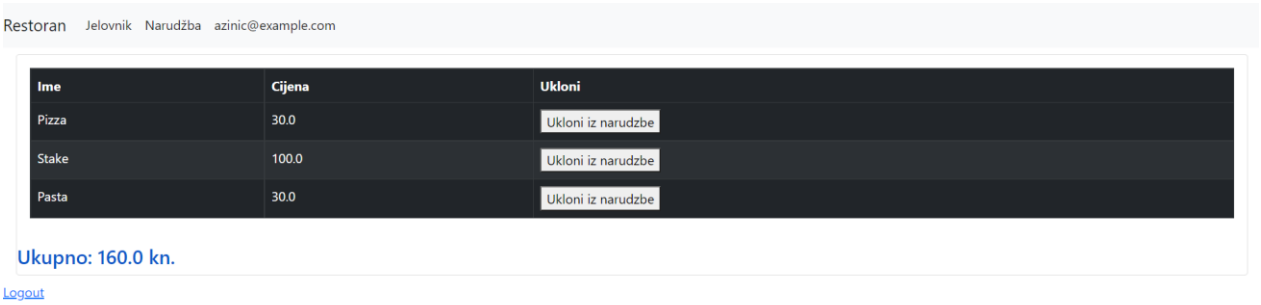
Sl. 3.7. Dijagram slijeda narudžbe

```
@RequestMapping("/add")
public String add(HttpSession session, @RequestParam("id_jelo") Jelo jelo,
                  @RequestParam(value = "qty", required = false, defaultValue = "1") int qty){
    Cart cart = cartManager.getCart(session);
    cart.addItem(jelo, qty);
    return "redirect:/cart";
}
```

Sl. 3.8. Funkcija add u Cart kontroleru



Sl. 3.9. HTML sučelje Jelovnik



Sl. 3.10. HTML sučelje narudžba

3.4.2. Primjer registracije korisnika

Slika 3.11 prikazuje red izvođenja funkcija pri kreiranju novog korisnika putem registracije. Korisnik prvo otvara login HTML stranicu na kojoj se nalazi gumb registracija, te se na pritisak gumba otvara HTML stranica registracije gdje korisnik popunjava osobne podatke i nakon toga klika gumb registracija. Gumb registracija poziva *User registration* kontroler koji u sebi ima definiranu funkciju *registerUserAccount* što se vidi prikazano na slici 3.12 koja sprema novog korisnika u bazu podataka što je prikazano slikom 3.13 te preusmjerava na login HTML stranicu putem koje se korisnik ulogira u web aplikaciju te dolazi na početnu HTML stranicu. Postupak kreiranja korisnika ilustriran je slikama 3.14, 3.15, 3.16.

Sl. 3.11. Dijagram slijeda registracije

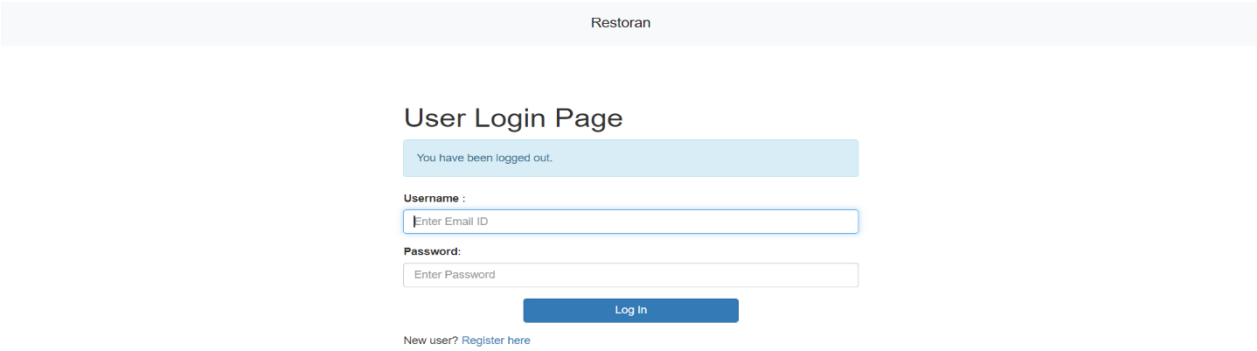
```
@PostMapping
public String registerUserAccount(@ModelAttribute("user") UserRegistrationDto registrationDto) {

        var user =  userService.save(registrationDto);
        if(user == null){
            return "registration";
        }
       return "redirect:/login";
}
```

Sl. 3.12. Funkcija registerUserAccount

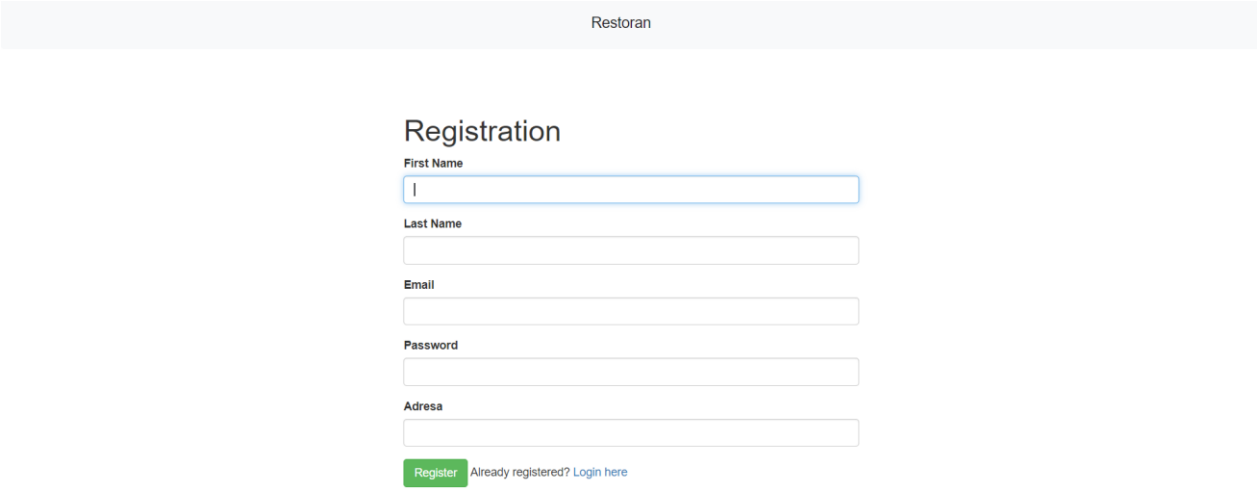
Grid

	id_kupac	address	email	first_name	last_name	password
1	1	Vukovarska 23	azo@example.com	Azinić	Andrija	$2a$10$uXZXv
2	2	Gundulićeva 55	azinic@example.com	Azinić	Andrija	$2a$10$p4yMs.

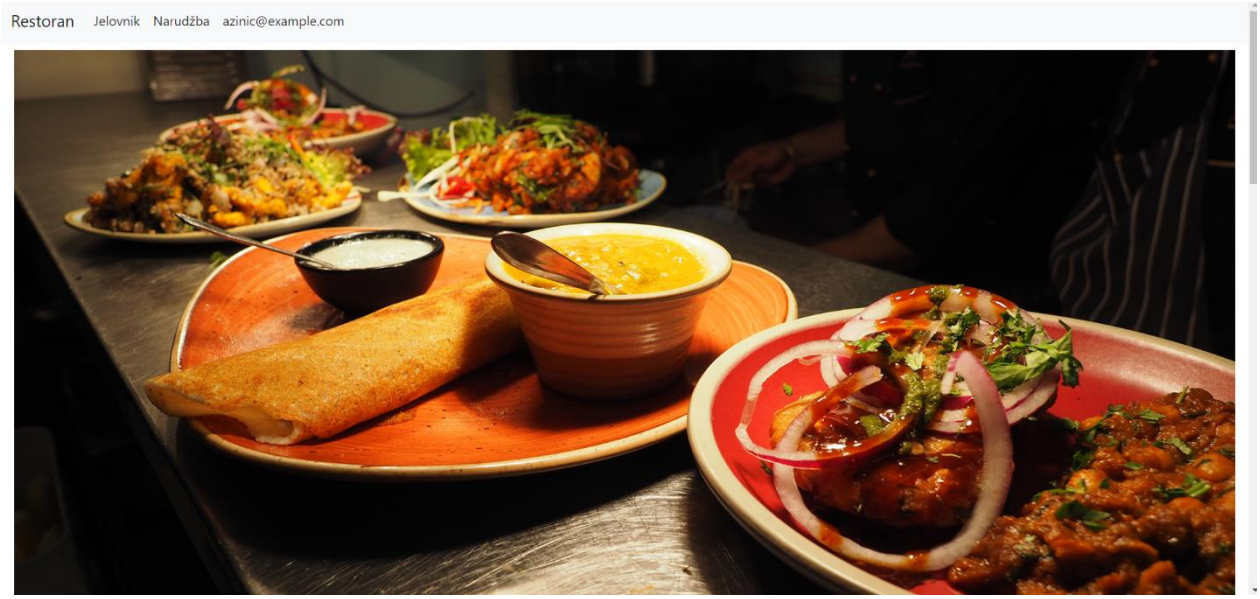
Sl. 3.13. Prikaz korisnika u bazi podataka



Sl. 3.14. HTML sučelje login



Sl. 3.15. HTML sučelje registracija



Sl. 3.16. HTML sučelje početne stranice

4. ZAKLJUČAK

Tema ovog završnog rada je implementacija višeslojne arhitekture pomoću razvojnog okruženja Spring, što je i realizirano studijom slučaja u obliku web aplikacije za vođenje ugostiteljskog objekta. Razvijena web aplikacija komunicira s okolinom putem arhitekturnog stila REST, a za razvoj korisničkog sučelja koristi se uzorak dizajna MVC. Web aplikacija je podijeljena na prednje sučelje koje je implementirano pomoću predloška Thymeleaf, opisnog jezika HTML te stilskog jezika CSS u kombinaciji s pozadinskom podrškom. Pozadinska podrška podijeljena je na sloj kontrolera, sloj repozitorija, sloj modela i pomoćni sloj. Web aplikacija koristi višeslojnu arhitekturu, što znači da buduće izmjene ili dodavanje novog sadržaja ne predstavljaju zahtjevan zadatak. Ovaj rad je rezultirao uspješnom implementacijom višeslojne arhitekture u obliku web aplikacije ugostiteljskog objekta te primjenom arhitekturnog stila REST i uzorka dizajna MVC.

Primarni izazovi pri izradi ove aplikacije su učenje razvojnog okruženja Spring, prilagodba izrade aplikacije u višeslojnoj arhitekturi u odnosu na izradu monolitne aplikacije te podjela paketa na model, pogled i kontroler. Neki od nedostataka postojećeg sučelja su samostalno dodavanje novog jela, brisanje postojećeg jela iz baze podataka te dodavanje popisa prijašnjih narudžbi. Cjelokupan dizajn aplikacije je vrlo jednostavan i nedostaju značajke kao što su informacije o ugostiteljskom objektu. Kod buduće nadogradnje aplikacije u planu je dodavanje novih funkcionalnosti kao što su mogućnost dodavanja novog jela u jelovnik, brisanje starih jela iz jelovnika i dodavanje popisa prijašnjih narudžbi. Također, u budućem radu se planira i poboljšanje dizajna cjelokupne aplikacije te dodavanje novih sadržaja koji pruža informacije o ugostiteljskom objektu.

LITERATURA

[1] *World wide web*, https://www.britannica.com/topic/World-Wide-Web, pristupljeno: 6.11.2021

[2]Web aplikacija, https://digitalya.co/blog/web-application-vs-website/, pristupljeno: 6.11.2021

[3] JSON, XML, https://www.geeksforgeeks.org/difference-between-json-and-xml/ , pristupljeno: 6.11.2021

[4] Prednje sučelje i podazdinska podrška, https://www.geeksforgeeks.org/frontend-vs-backend/, pristupljeno: 6.11.2021

[5] Progresivna aplikacija, https://www.simicart.com/blog/progressive-web-apps-examples/, pristupljeno: 6.11.2021

[6] Višeslojna arhitektura, https://hub.packtpub.com/what-is-multi-layered-software-architecture/, pristupljeno: 12.11.2021

[7] Sun Yigui, Xu Zhenqiang, „*Design Solution of Scientific Research Achievements Management System based on .NET multi-layer architecture* ", *Henan University of Technology, Zhengzhou*, str. 68-69, Henan, 2010, pristupljeno: 6.12.2021

[8] Lan Lina, „ *Personalized e-Learning System Based on Multi-Layer Architecture*", *School of Network Education, Beijing University of Posts and Telecommunications* (BUPT), str. 278, Peking, 2009, pristupljeno: 6.12.2021

[9] Rafał Grycuk, Patryk Najgebauer, Robert Nowicki, Rafał Scherer," *Multilayer Architecture for Content-based Image Retrieval Systems", Institute of Computational Intelligence, Cze¸stochowa University of Technology*, str. 119-120, Armii Krajowej, 2020, pristupljeno: 6.12.2021

[10] M. Bertocco, S. Cappellazzo, A. Flammini, M. Parvis, „*A multi-layer architecture for distributed data acquisition" Dipartimento di Electtronica e Informatica, UniversitA di Padova*, str. 1261, Padova, 2002, pristupljeno: 6.12.2021

[11] Razvojno okruženje Spring https://www.tutorialspoint.com/spring/spring_overview.htm , pristupljeno: 21. 11.2021

[12] Spring Boot, https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm, 21.11.2021

[13] Okvir za objektno-relacijsko mapiranje Hibernate, https://www.tutorialspoint.com/hibernate/hibernate_quick_guide.htm, pristupljeno: 23.11.2021

[14] Relacijska baza podataka MySQL, https://www.talend.com/resources/what-is-mysql/, pristupljeno: 23.10.2021

[15] Alat za izradu predložaka Thymeleaf , https://www.baeldung.com/thymeleaf-in-spring-mvc,

pristupljeno: 23.10.2021

SAŽETAK

Cilj ovog završnog rada je izrada web aplikacije zasnovane na višeslojnoj arhitekturi kako bi se demonstrirali izazovi prilikom takvog razvoja, ali naglasili i njene prednosti u odnosu na monolitnu arhitekturu. Prednosti i izazovi su demonstrirani na razvoju web aplikacije za vođenje ugostiteljskog objekta pomoću razvojnog okvira Spring. Web usluga komunicira s okolinom putem arhitekture REST, a za razvoj korisničkog sučelja upotrebljava uzorak dizajna MVC. U radu je prvo dan opis sustava World wide web, a nakon toga je opisana višeslojna arhitektura. Također je izložen opis specifikacije arhitekture predloženog programskog rješenja koji je razvijen kako bi se istaknule glavne prednosti, ali i izazovi. Opisane su i sve korištene tehnologije, kao i dvije funkcionalnosti web aplikacije. Po uzorku sloja modela višeslojne arhitekture web aplikacija je podijeljena na slojeve: sloj modela, sloj repozitorija, pomoćni sloj, sloj kontrolera i prezentacijski sloj. U konačnici je donesen zaključak o izazovima razvoja i prednostima višeslojne arhitekture.

Ključne riječi: monolitna arhitektura, obrazac dizajna MVC, razvojno okruženje Spring, višeslojna arhitektura, web aplikacija.

ABSTRACT

This aim of the paper is to create web application based on multi-layer architecture in order to demonstrate the challenges of such development, but also emphasize its advantages over monolithic architecture. The advantages and challenges were demonstrated in the development of a web application for running a catering facility using the Spring development framework. The web service communicates with the environment through the REST architecture and uses the MVC design pattern to develop the user interface. The paper first describes the World wide web system, then the multi-layer architecture. The paper also describes the architecture specification of the proposed software solution that was developed to highlight the main advantages and challenges. All the technologies used and the two functionalities of the web application are also described. According to the multilayer architecture model, the web application is divided into; model layer, repository layer, services layer, implementation of a services layer, controller layer and view layer. Ultimately, a conclusion was reached on the challenges of development and the benefits of a multi-layered architecture.

Keywords: monolithic architecture, MVC design pattern, Spring framework, multilayer architecture, web application.

ŽIVOTOPIS

Andrija Azinić je rođen u Đakovu 29. ožujka 1999. godine. Pohađao je Elektrotehničku i prometnu školu Osijek, smjer Elektrotehničar. Stručni studij Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek upisao je 2018. godine, a uz fakultet polazi i završava 2020. godine tečaj za Java programera u školi Edunova. Aktivno se služi engleskim jezikom u govoru i u pismu.

_____

Potpis autora

POPIS UPOTRJEBLJENIH KRATICA

CRUD – Create, Read, Update and Delete

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

JPA – Java Persistence API

JSON – JavaScript Object Notation

Java EE – Java Enterprise Edition

MVC – Model-View-Controller

PWA – Progressive web application

REST – REpresentational State Transfer

URL – Uniform Resource Locator

W3C – World wide web Consortium

XML – Extensible Markup Language

PRILOZI (na CD-u)

Prilog 1. Završni rad u docx formatu

Prilog 2. Programski kod