Univerzitet Singidunum
Tehnički fakultet

Milan Dimitrijević

UPOTREBA SAVREMENIH VEB TEHNOLOGIJA U RAZVOJU APLIKACIJE ZA NARUČIVANJE HRANE

Diplomski rad

Beograd, 2025.

Univerzitet Singidunum
Tehnički fakultet

Studijski program
Softversko i informaciono inženjerstvo

Upotreba savremenih veb tehnologija u razvoju aplikacije za naručivanje hrane

Diplomski rad

Mentor
prof. dr Angelina Njeguš

Student
Milan Dimitrijević 2021/202181

Beograd, 2025. godine

SADRŽAJ

1. Uvod

U ovom radu je prikazan model aplikacije za online naručivanje i dostavu hrane. U oblasti naručivanja hrane, veb aplikacije omogućavaju korisnicima jednostavniji pristup restoranima i njihovoj ponudi, dok vlasnicima pružaju mogućnost efikasnijeg poslovanja i praćenja narudžbina. Cilj ovog rada jeste da se prikaže proces razvoja veb aplikacije za naručivanje hrane, korišćenjem savremenih tehnologija i arhitektonskih obrazaca. Aplikacija slična ovoj na trenutnom tržištu je „*Wolt*“.

2. Veb aplikacije

Veb aplikacija je aplikacija kojoj se pristupa od strane korisnika preko mreže kao što je Internet ili intranet. Termin takođe označava i primenu kompjuterskog softvera koji je kodiran u veb pretraživaču koji podržava programske jezike (kao što su Javaskript, u kombinaciji sa jezikom koji služi za obeležavanje kao što je HTML) i oslanja se na zajedničkom veb pretraživaču da donese izvršnu aplikaciju.

Veb aplikacije su popularne zbog sveprisutnosti veb pretraživača, a pogodnost korišćenja veb pregledača kao klijenta se ponekad zove „tanki/mršavi klijent”. Sposobnost da se ažurira i održava veb aplikacija bez distribucije i instaliranja softvera na hiljade potencijalnih klijentskih računara je ključni razlog za njihovu popularnost. Uobičajene veb aplikacije uključuju slanje pošte, internet trgovinu, aukciju, te mnoge druge funkcije.

Veb aplikacija sadrži tri glavne komponente: veb server, mrežnu konekciju i jedan ili više klijentskih pregledača. Statički veb server distribuira veb strane formatiranih informacija klijentima koji ih zahtevaju. Zahtev se postavlja preko mrežne konekcije i upotrebljava HTTP(S) protokol. U nekim situacijama sadržaj stranice nije statički određen (memorisan u fajlu), već se sklapa dinamički od informacija iz baze podataka i formatira na osnovu niza instrukcija (skripta) koji se čuva u fajlu. Veb server upotrebljava filter stranica da interpretira i izvrši skripte. Veb sajtovi koji primenjuju ovu strategiju nazivaju se dinamički sajtovi.

Tipovi veb aplikacija su:

- Render aplikacije na strani klijenta – većina logike aplikacije se pokreće u veb pretraživaču klijenta
- Render aplikacije na strani servera – većina logike aplikacije se pokreće na serveru
- Višeslojne veb aplikacije – logika aplikacije je podeljena na više slojeva

2.1. Istorija razvoja veb aplikacija

U ranijim računarskim modelima, npr. u klijent-server periodu, zahtevi za aplikacije su deljeni između koda koji je na serveru i koda koji je instaliran na svakom klijentskom računaru. Drugim rečima, aplikacija ima svoj klijentski program, a to je njegov korisnički interfejs i mora da se posebno instalira na ličnom računaru svakog korisnika. Nadogradnja koda na serverskoj strani će obično zahtevati i nadogradnju koda na klijentskoj strani, dajući na taj način povoljniju cenu i smanjujući produktivnost.

Nasuprot tome, veb aplikacije koriste veb dokumente napisane u nekom od standardnih formata kao što su HTML i JavaScript, koji su podržani od strane različitih veb pretraživača. Veb aplikacija se može smatrati kao specifična varijanta klijent-server softvera, gde se klijentski softver preuzima na mašini prilikom posete odgovarajuće veb stranice, primenom standardnih postupaka kao što je HTTP protokol. Tokom sesije, veb pretraživač tumači i prikazuje stranice, te deluje kao univerzalni klijent za bilo koju veb aplikaciju.

U početku veba, svaka veb stranica dostavljala se klijentu kao statički dokument. Redosled stranica može da obezbedi interaktivno iskustvo ukoliko je korisniku unos vraćen preko elemenata veb formulara ugrađenog u HTML stranici.

Godine 1995. Netskejp (*Netscape*) je uveo skript koji se izvršava na strani klijenta, poznat kao JavaScript. Dakle, umesto slanja podataka na server da bi generisali celu stranicu, ugrađeni skriptovi sa preuzete stranice mogu da obavljaju različite zadatke, kao što su validacija unosa ili prikazivanje/skrivanje delova stranica.

Godine 1996, makromedija uvodi Fleš (*Flash*) — vektorsku grafiku koja se može dodati pretraživaču kao dodatak od plagin (*Plug-in*), kako bi se omogućio prikaz animacije na veb stranici. To omogućava upotrebu skript-jezika za izvršavanje programa na strani klijenta bez potrebe za komunikaciju sa serverom.

Godine 1999, koncept „veb aplikacija” je uveden u jeziku Java, kao *Servlet Specification* verzija 2.2. U to vreme i JavaScript i XML su već bili razvijeni, ali Ajax još uvek nije postojao
.

Godine 2005, nastaje termin Ajax zajedno sa aplikacijama kao što su Gmail, a klijentske strane ponovo bivaju povezane. Skriptovi počinju da komuniciraju sa serverom tako što će skladištiti/preuzimati podatke bez preuzimanja cele veb stranice.

Godine 2011, HTML5 biva završen, sa grafičkim i multimedijalnim mogućnostima bez potrebe za dodacima na strani klijenta. HTML5 je obogatio semantički sadržaj dokumenata. VebGL API je otvorio put naprednoj 3D grafici baziranoj na HTML5 i JavaScript jeziku.

Veb razvoj ili razvoj internet softvera (*veb development*) predstavlja različite poslove koji se obavljaju prilikom razvoja veb sajta ili veb aplikacije za internet ili intranet. Nivoi složenosti su različiti i mogu varirati od razvoja najjednostavnije statičke veb stranice, pa sve do izrade najsloženijih internet aplikacija za elektronsko poslovanje.

Pojam veb razvoj obuhvata procese kao što su veb dizajn, izrada i razvoj veb sadržaja, programiranje, razvoj baza podataka, konfigurisanje servera i mreže, upravljanje sadržajem, razvoj elektronskog poslovanja, marketing. U slučaju veće organizacije ili biznisa, timovi za veb razvoj mogu brojati stotine ljudi (veb developera). Malim organizacijama su za ovakve poslove dovoljne samo jedna ili dve osobe.

Izrada sajtova (*Website Making*) se vrši pomoću nekoliko različitih programskih jezika: HTML, XHTML, CSS, Flash, PHP, Java Script, JQuery itd. Osnova svakog veb sajta je HTML i svi pretraživači vide isključivo podatke napisane u tom programskom jeziku. Svi ostali programski jezici se koriste u kombinaciji sa HTML-om, za modernizovanje i kreiranje naprednih funkcija sajta.

Prvi sajtovi su izrađeni samo pomoću programskog jezika HTML. Vremenom su počeli da se razvijaju alati koji će pomagati programerima da što lakše ispišu kod i naprave sajt. Prvi takav alat je razvio Adobe i zvao se Dreamweaver. Ovaj i slični softveri su se koristili sve do uvođenja W3C standarda, koji je pomerio granice. CSS polako ali sigurno postaje neizostavan deo svakog modernijeg veb sajta i tako je ostalo do današnjih dana.

Novo doba u izradi veb sajtova je nastupilo 2003. kada se pojavio WordPress CMS koji se do danas veoma razvio i nastavlja da se razvija. WordPress je zasnovan na PHP programskom jeziku i napravljen je tako, da se prilagođava korisnicima, a ne programerima. Svi najmoderniji sajtovi se zasnivaju na izradi pomoću nekog CMS-a. Najpoznatiji CMS-ovi su WordPress, Joomla, Drupal, Magma.

Posebno interesantni načini izrade sajtova su Drag & Drop sistemi za izradu sajtova. Ovi sistemi su maksimalno prilagođeni korisnicima. Ideja je da korisnici sami mogu da naprave za sebe veb sajt bez poznavanja bilo kog programskog jezika. Najpoznatiji dragendrop sistemi su Weebly i Wix.

2.2. Savremene tehnologije za razvoj veb aplikacija

Danas se razvoj veb aplikacija oslanja na čitav ekosistem savremenih tehnologija koje zavise od tipa aplikacije, ali uopšteno, ključne tehnologije i trendovi su:

- Frontend programiranje (programiranje na strani korisnika)
- Backend programiranje (programiranje na strani servera)
- Baze podataka
- Bezbednost

2.2.1. Frontend programiranje

Front-end veb razvoj je postupak kreiranja HTML, CSS i JavaSkript koda za veb sajt ili veb aplikaciju koji je vidljiv korisniku i sa kojim ima direktan kontakt. Osnovni izazov koji se javlja kod front-end veb razvoja je da se alati i tehnike koji se koriste za njegovo kreiranje konstantno razvijaju tako da programeri moraju stalno da budu upoznati na koji način se ova oblast razvija.

Cilj dizajna veb sajta je da se obezbedi da korisnici koji njemu pristupe vide informacije u formatu koji je lako čitljiv i pre svega relevantan. To se još više komplikuje činjenicom da korisnici danas koriste različite uređaje sa različitim veličinama ekrana i rezolucijom, što primorava dizajnere da sve ove pojedinosti uzimaju u obzir prilikom dizajna sajta. Oni moraju da obezbede da se sajtovi pravilno prikazuju u različitim veb pregledačima, operativnim sistemima i uređajima, što zahteva pažljivo planiranje od strane programera.

Postoji mnogo dostupnih alata koji se mogu koristiti za front-end razvoj sajtova, pa je razumevanje toga koji su alati najbolje uklapaju za specifične potrebe, veoma bitno kako bi se napravila razlika između razvoja sajtova koji se lako hakuju i onih dobro dizajniranih, skalabilnih sajtova.

HTML (*HyperText Markup Language*) - jezik za označavanje hiperteksta. HTML je standardizovani jezik koji se koristi pri strukturiranju tekstova, medija i ugrađenih objekata u veb stranice i elektronsku poštu, HTML je opisni jezik specijalno namenjen opisu veb stranica, pomoću koga se jednostavno mogu odvojiti elementi kao što su naslovi, paragrafi, citati i sl. HTML nije programski jezik, njime ne možemo izvršavati nikakve zadatke, čak ni najjednostavniju operaciju sabiranja ili oduzimanja. On služi samo za opis naših hipertekstualnih dokumenata. HTML datoteke su zapravo obične tekstualne datoteke sa ekstenzijom *.html.*

CSS (*Cascading Style Sheets*) je jezik formatiranja pomoću kog se definiše izgled elemenata veb stranice. Tri osnovne karakteristike CSS jezika su mogućnost za definisanje klasa

za izgled, boje i fontove. Ovi elementi omogućavaju pristupačniji i fleksibilniji sadržaj kao i kontrolu veb dizajnera nad određenom grupom HTML elemenata u sadržaju. CSS se, dakle, koristi za definisanje opštih pravila o tome kako se elementi ponašaju unutar veb stranica i kako izgledaju, gde se nalaze, njihove veličine, transparentnost itd.

JavaScript je kompaktan i objektivno-orijentisan skriptni jezik za razvoj internet aplikacija, po sistemu klijent-server. Programski kod se upisuje direktno u HTML stranicu i omogućava izradu dinamičkih veb stranica. Jezgro jezika JavaScript podržava brojeve, znakovne nizove i logičke vrednosti kao osnovne tipove podataka. Najčešće se koristi u pretraživačima veba (*Web browsers*), pa se jezgro opšte namene proširuje objektima koji omogućavaju skriptovima interakciju sa korisnikom, upravljanje pretraživačem veba i izmene sadržaja dokumenta koji se pojavljuje unutar prozora čitača. Pošto se JavaScript interpretira umesto da se prevodi, često se smatra da je to jezik za skriptovanje, a ne pravi programski jezik.

Frontend framework je alat koji koji olakšava pravljenje korisničkog interfejsa (UI) i interakcije u pretraživaču. Umesto da se sve piše od nule, framework nudi gotove komponente, strukture koda, organizaciju fajlova, reactive UI, optimizaciju performansi. Glavni savremeni framework-ovi su **React, Angular, Vue.js, Svetle, SolidJS, Bootstrap.** Bootstrap je najpopularniji CSS okvir za razvoj prilagodljivih i mobilnih veb sajtova. Bootstrap 4 je najnovija verzija Bootstrap-a.

Aplikacije koje se sastoje samo iz frontenda se zovu statičke veb aplikacije. Sadržaj statičke veb aplikacije se ne menja, ona ostaje onako kako je napisana u frontend kodu.

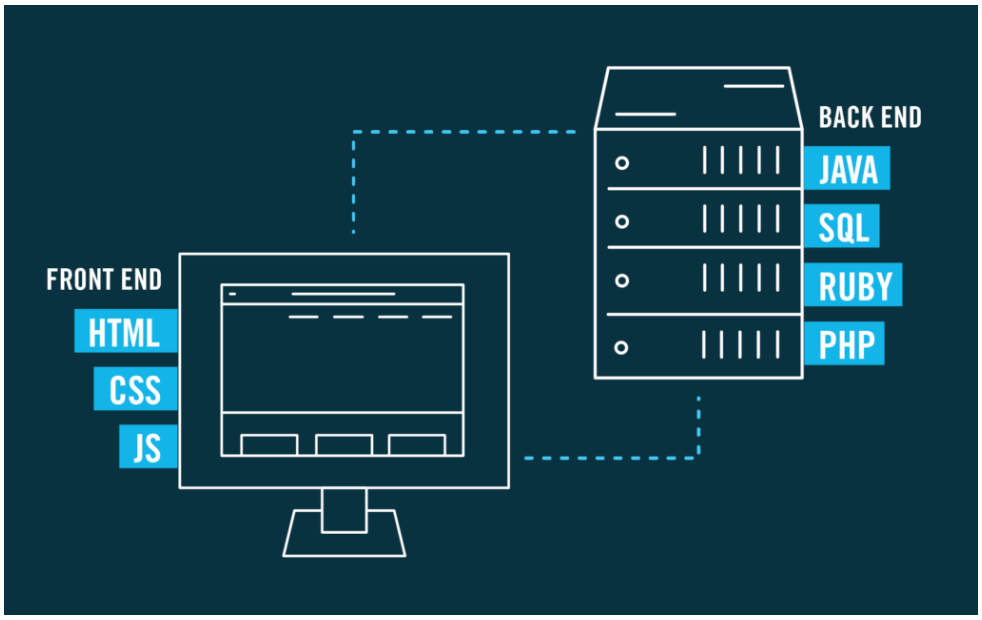
2.2.2. Backend programiranje

Backend je deo veb sajta koji nije vidljiv posetiocima sajta. Odgovoran je za skladištenje, čuvanje i organizovanje podataka i osiguravanju da sve na strani korisnika funkcioniše. Backend komunicira sa frontend-om, tako što šalje i prima podatke koje se prikazuju na veb sajtu. Kada popunjavate kontakt formu, upišete veb adresu, ili kupujete online, pretraživač šalje zahtev backend-u , koji se vraća kao informacija u obliku frontend koda koji browser može da prikaže.

Backend se sastoji iz baze podataka, serverske logike, API-ja, servera i osigurava podatke od napada. Dodatkom backend-a na već postojeći frontend, veb aplikacija prelazi iz statičke u dinamičnu. Sadržaj dinamične veb aplikacije se može menjati u zavisnosti od toga šta se nalazi u bazi podataka.

Backend tehnologije obuhvataju programski jezici (Java, Python, PHP, Node.js), okvire kao što su Spring za Javu, Django za Python, Twig za PHP ili Express za Node.js, sisteme za upravljanje bazama podataka (MySQL), kao i rad sa API-jima i serverskim okruženjem.

API je akronim za *Application Programming Interface* koji omogućava komunikaciju između komunikaciju između frontend-a i backend-a. Backend developer pravi servise kojima frontend aplikacija pristupa preko API-ja da bi dobila potrebne podatke. Svaki put kada se koristi neka aplikacija koristi se API. Korišćenjem API-ja programeri mogu integrisati već postojaće funkcije i podatke iz drugih servisa u svoje aplikacije, umesto da sve grade od nule. Na primer, aplikacija za vremensku prognozu koristi API za preuzimanje podataka o prognozi iz meteorološke službe.



Slika 1. Dinamična veb aplikacija

2.2.3. Baze podataka

Baza podataka je kolekcija podataka organizovanih za brzo pretraživanje i pristup, koja zajedno sa sistemom za administraciju, organizovanje i memorisanje tih podataka, čini sistem baze podataka. Iz ugla korisnika, podaci su na neki logički način povezani. Oni predstavljaju neke aspekte realnog sveta.

Korisnici pristupaju bazi podataka prvenstveno preko upita. Korišćenjem ključnih reči i svrstavanjem komandi korisnici mogu brzo da pronađu, preurede, grupišu i odaberu oblast u mnogim zapisima koje treba vratiti ili pomoću kojih treba sastaviti izveštaje o naročitoj grupi podataka u skladu s pravilima dotičnog sistema vođenja baze podataka.

Sistem za upravljanje bazama podataka (*database management system*, DBMS) je računarski program namenjena rukovođenju bazom podataka, velikim skupom strukturiranih podataka, i izvršavanju operacija na podacima, koje zahtevaju mnogobrojni

korisnici. Tipične upotrebe ovih sistema uključuju računovodstvo ili sisteme za podršku mušterijama. U najpoznatije programe iz ove oblasti spadaju: Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker.

Postoje različite vrste baza podataka, zavisno od toga na koji način su podaci interno organizovani. Tako se razlikuju hijerarhijske, mrežne, relacionalne, objektno-orijentisane, objektno-relacione, prilagođene za VEB, XML i multimedijske baze podataka.

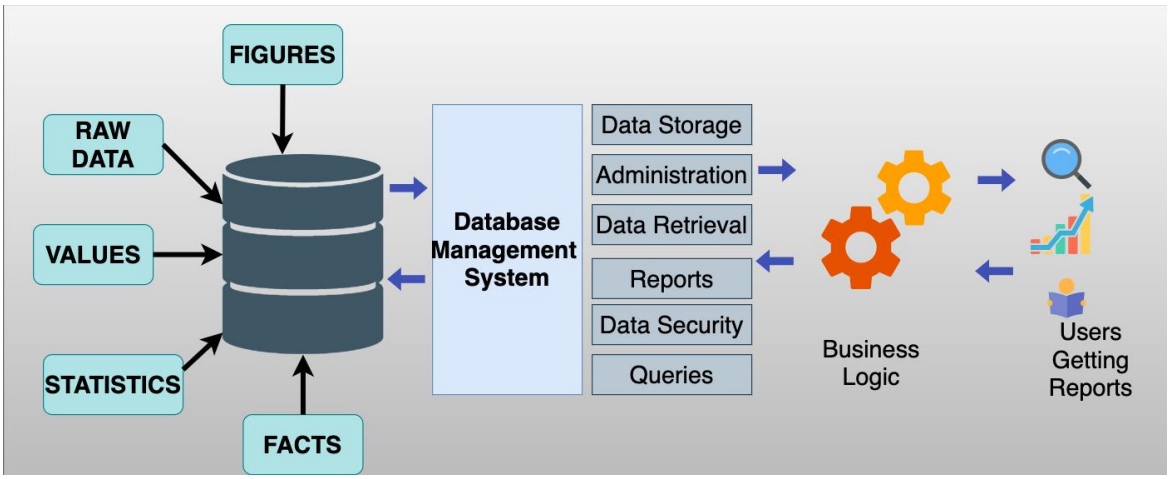
Podaci su predstavljeni na uniformni način (npr. u relacionim bazama podataka podaci su organizovani u tabelama), što olakšava pristup i korišćenje od strane eksternih programa. Tako jednu bazu podataka može koristiti niz različitih programa, pisanih u različitim programskim jezicima.

Izvan sveta profesionalne informacione tehnologije, termin baza podataka se često koristi da se odnosi na bilo koju kolekciju povezanih podataka jer zahtevi za veličinu i korišćenje obično zahtevaju korišćenje sistema za upravljanje bazom podataka.

Postojeći DBMS-ovi pružaju različite funkcije koje omogućavaju upravljanje bazom podataka i njenim podacima koji se mogu klasifikovati u četiri glavne funkcionalne grupe:

- Definicija podataka – kreiranje, modifikacija i uklanjanje definicija koje definišu organizaciju podataka.
- Ažuriranje – umetanje, modifikacija i brisanje stvarnih podataka.
- Preuzimanje – pružanje informacija u obliku koji se može direktno koristiti ili za dalju obradu od strane drugih aplikacija. Preuzeti podaci mogu biti dostupni u formi koja je u osnovi ista kao što je uskladištena u bazi podataka ili u novom obliku dobijenom izmenom ili kombinovanjem postojećih podataka iz baze podataka.
- Administracija – registrovanje i nadgledanje korisnika, sprovođenje bezbednosti podataka, praćenje performansi, održavanje integriteta podataka, bavljenje kontrolom istovremenosti i oporavak informacija koje su oštećene nekim događajem kao što je neočekivani sistemski kvar.

Baza podataka i njen DBMS su u skladu sa principima određenog modela baze podataka. Sistem baze podataka se odnosi zajedno na model baze podataka, sistem za upravljanje bazom podataka i bazu podataka.



Slika 2. Baza podataka

Relacione baze podataka

SQL (*Structured Query Language*) je relacioni upitni jezik (ANSI i ISO standard). Relacije se kreiraju jednom naredbom i odmah su dostupne, što ga čini jednostavnim za korišćenje. Uniforman je, jer se svi podaci i rezultati operacija prikazuju u vidu tabele i omogućava interaktivno i klasično programiranje. Sve do verzije SQL:1999 ovaj jezik je bio neproceduralan, odnosno njime se specificiralo ŠTA, a ne i KAKO nešto treba uraditi.

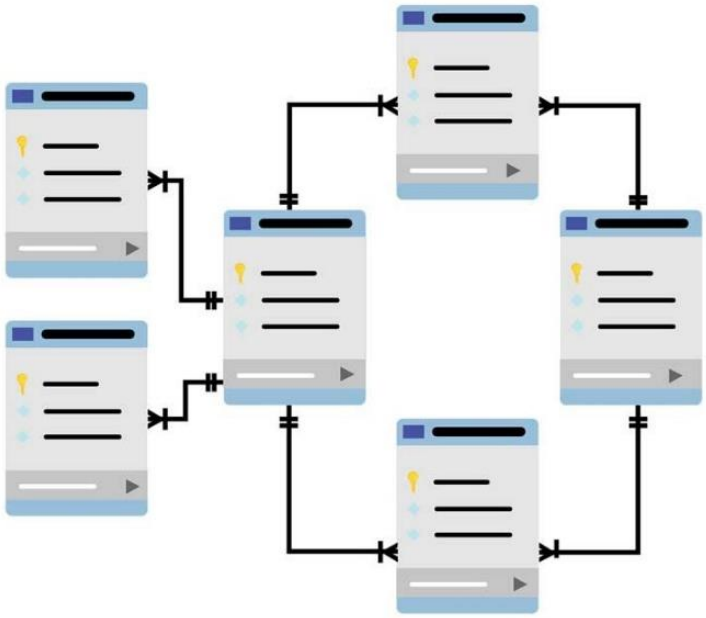
SQL se koristi za pristup i manipulaciju sistemima za upravljanje bazom podataka (DBMS), što najčešće podrazumeva čitanje i izmenu podataka u različitim bazama podataka. SQL je najviše korićeni programski jezik za baze podataka. Za njega kažemo da je deklarativni jezik jer se njegovim rečenicama prvenstveno iskazuje šta se želi učiniti, prepuštajući DBMS-u da odredi način na koji će se to izvesti. SQL obuhvata unos podataka, upite, ažuriranje i brisanje, šeme kreiranja i menjanja, kao i podatke za kontrolu pristupa.

Tvorac SQL-a je Donald Čamberlin, a nastao je u IBM-ovoj istraživačkoj laboratoriji u San Hozeu, Kalifornija 1974. godine, dakle na istom mestu gde je E. F. Kod 1970. definisao osnovne koncepte relacionog modela podataka. Jezik se u početku zvao SEQUEL (*Structured English Query Language*) i predstavljao je programski interfejs (API) za System R, prototipski sistem za upravljanje bazom podataka koji se razvijao kao deo istraživačkog projekta pod istim nazivom.

Pojava komercijalnih relacionih sistema uvećala je značaj i ubrzala proces standardizacije relacionog upitnog jezika. Prva etapa tog procesa završila se 1986. godine usvajanjem SQL-a kao standardnog relacionog upitnog jezika. Ta prva verzija SQL standarda je poznata pod nazivom SQL-86. Njom su standardizovane osnovne karakteristike SQL-a kao deklarativnog relacionog upitnog jezika. Međutim, mnoge bitne karakteristike jezika ostale su nestandardizovane. To je

dovelo do revizija standarda, koji je usvojen 1989. godine i kojom su standardizovane karakteristike koje se odnose na očuvanje integriteta baze podataka i povezivanje sa klasičnim programskim jezicima. Ta verzija SQL standarda poznata je pod nazivom SQL-89. 1992. godine usvojena je sledeća bitna revizija standarda, poznata pod nazivom SQL-92 ili SQL-2, kojom je SQL zaokružen kao programski jezik, a obim standarda uvećan šest puta u odnosu na polaznu verziju. Naredna verzija SQL standarda usvojena je 1999. godine.

Iako su početne verzije SQL-a bile prilično jednostavne, bliske korisniku i u velikoj meri deklarativne za SQL:1999 se može reći da je kompleksan, proceduralno/deklarativan jezik. U njega su uključeni koncepti objektne tehnologije, mehanizam trigera, rekurzija i proceduralna proširenja. Da bi se povećala funkcionalnost jezika, u SQL:1999 standardu uvedena je proceduralna nadgradnja SQL-a, koju uglavnom čine upravljačke strukture slične upravljačkim strukturama klasičnih programskih jezika. SQL-1999 standard definiše više načina korišćenja SQL-a. Dva osnovna načina su direktno (interaktivno) korišćenje SQL-a i povezivanje SQL-a sa klasičnim programskim jezicima („ugrađeni“ SQL).



Slika 3. Primer SQL baze podataka

3. Razvoj veb aplikacije za onlajn naručivanje i dostavu hrane

Cilj aplikacije za onlajn naručivanje i dostavu hrane je da omogući sledeće funkcionalnosti korisnicima:

- Prijava i odjava na aplikaciju
- Promena ličnih podataka
- Pretraga hrane
- Naručivanje hrane
- Plaćanje
- Unos podataka za dostavu
- Pregled istorije naručivanja

S obzirom, da je ova aplikacija osmišljena tako da se izvršava kao sajt, njenim funkcionalnostima moze pristupiti svaki korisnik.

3.1. Tehnologije neophodne za rad aplikacije

Za razvoj ove veb aplikacije korišćene su sledeće tehnologije i sistemi.

Appache HTTP server (*Xampp*) - veoma zastupljen širom sveta, veliki broj korisnika ga koristi gotovo za sve aplikacije. Open-source je i dostupan za platforme kao što su Windows, Unix i Linux.

Xampp paket pored Apache veb servera sadrži i MariaDB bazu podataka, a takođe i prevodioce za skripte pisane u PHP i Perl programskim jezicima. U okviru Xampp-a je dostupan i MySQL sistem za upravljanje relacionim bazama podataka i njihovo čuvanje.

XML (*Xtensible Markup Language*) predstavlja podatke za opis podataka tj. sintaksu, u tekstualnom formatu. On je kreiran sa namerom da bude jednostavan za učenje, jeftin, brz i optimizovan za internet. XML je tu da opiše strukturu, integriše protokole između aplikacija, da razmenjuje podatke. XML je skup pravila koja omogućavaju kreiranje tekstualnog formata koji opisuje strukturu podataka.

PHP (*Hypertext Preprocessor*) je reflektivni programski jezik prvobitno dizajniran za pravljenje dinamičkih veb stranica. Primarni način izvršavanja programa napisanih u PHP jeziku je u okviru obrade zahteva ka veb serveru, što podrazumeva postojanje PHP interpretera/izvršioca unutar veb servera. PHP obrađuje zahteve uz pomoć Apache servera. Kada korisnik pošalje zahtev

za određenim resursom, zahtev će preuzeti Apache server. PHP će započeti komunikaciju sa bazom podataka i iščitaće potrebne podatke. Nakon toga, uspostaviće konekciju sa folderima kako bi preuzeo potrebnu strukturu stranice i CSS stilove. Kada preuzme sve što je potrebno, počeće sa obradom i vratiće korisniku željeni prikaz, bez PHP sintakse u izvornom kodu, pošto se sve izvršava na strani servera.

UML (*Unified Modeling Language*) predstavlja skup grafičkih notacija zasnovanih na jedinstvenom metamodelu. Pod grafičkom notacijom podrazumijeva se skup grafičkih elemenata koji definišu sintaksu jezika. Metamodel je dijagram koji opisuje koncepte jezika za modelovanje. UML je široko prihvaćen u praksi, između ostalog, i zbog toga što se ni grafička notacija, ni metamodel ne moraju strogo primenjivati.

Klijent/server proces, gde klijent ili jedan njegov deo koji zahteva uslugu od drugog procesa (bilo na lokalnom ili na udaljenom računaru), a server čeka zahteve drugih procesa i isporučuje odgovor na njih.

Veb server je softver za obradu klijentskih zahteva koji podržava različite protokole, npr. HTTP i HTTPS.

Twig je šablonski sistem za PHP programski jezik. Sintaksa potiče od Džindž šablonskog sistema i Đango veb frejmova. Twig je proizvod otvorenog koda. Simfoni 2 PHP framework dolazi sa podrškom za Twig kao njegovim podrazumevanim šablonskim sistemom.

PDO je skraćenica za *PHP Data Objects*. PDO je konzistentan način pristupa bazama podataka. To znači da programeri mogu mnogo lakše da pišu prenosivi kod. PDO nije sloj apstrakcije kao PearDB. PDO je više nalik sloju pristupa podacima koji koristi unificirani API.

Composer je alat za upravljanje zavisnostima u PHP-u. Omogućava vam da deklarišete biblioteke od kojih vaš projekat zavisi i da njima upravljate (instalirati/ažurirati).

Sesija je način za čuvanje informacija koje će se koristiti na više stranica. Za razliku od kolačića, informacije se ne pohranjuju na računaru korisnika. Varijable sesije sadrže informacije o jednom korisniku i dostupne su svim stranicama u jednoj aplikaciji.

Validacija znači proveriti unos koji je korisnik dostavio. Postoje dve vrste validacije:

1. validacija na strani klijenta, koja se izvodi na veb pretraživačima klijentske mašine;
2. validacija na strani servera - nakon dostavljanja podataka, podaci su poslati serveru i vrši se provera ispravnosti na serverskoj mašini.

Yarn je iniciran od strane Facebook-a, a sada je podržavaju velike kompanije kao što je Google. To je uglavnom zamjena za upravljanje paketima u veb projektima, sa nekoliko nedostajućih karakteristika u npm CLI. Yarn je potpuno kompatibilna sa strukturom npm paket menadžera.

jQuery je "piši manje, radi više" JavaScript biblioteka. Svrha jQuery-a je da olakša korišćenje JavaScript-a. jQuery ima mnogo uobičajenih zadataka koji zahtevaju mnogo linija JavaScript koda za izvršenje, i odlaže ih u metode koje možete pozvati jednom linijom koda. jQuery takođe pojednostavljuje mnogo komplikovanih stvari iz JavaScript-a, kao što su ajax pozivi i DOM manipulacija.

codeguy/upload je paket za rukovanje učitavanjem datoteka sa proširivim strategijama validacije i skladištenja.

JSON (*JavaScript Object Notation)*. JSON je format za skladištenje i prenos podataka, često se koristi kada se podaci šalju sa servera na veb stranicu. JSON je "samoopisujući" i lako razumljiv.

HTTPS (*Hipertext Transfer Protocol Secure)*. To je protokol za osiguravanje komunikacije između dva sistema, pretraživača i veb servera. “S” u HTTPS-u označava “Secure”. To je bezbedna verzija standardnog "protokola za prenos hiperteksta" koji vaš veb pretraživač koristi kada komunicira sa veb lokacijama.

TCP/IP (*Transmission Control Protocol / Internet Protocol*) je skup komunikacijskih protokola koji se koriste za povezivanje mrežnih uređaja na internetu. TCP/IP se može koristiti i kao komunikacioni protokol u privatnoj mreži.

3.2. Dijagram slučaja korišćenja

Dijagram slučajeva korišćenja (*Use case diagram*) prikaz je interakcije korisnika sa sistemom koji pokazuje odnos između korisnika i različitih slučajeva korišćenja u kojima je korisnik uključen. Dijagram slučaja korišćenja može identifikovati različite tipove korisnika sistema i različite slučajeve korišćenja i često će biti propraćen i drugim tipovima dijagrama. Slučajevi korišćenja predstavljeni su krugovima ili elipsama.

Iako se slučaj korišćenja može razbiti u mnogo detalja o svakoj mogućnosti, dijagram slučajeva korišćenja može obezbediti pogled na sistem na višem nivou. Oni obezbeđuju pojednostavljen i grafički prikaz onoga što sistem zaista mora da radi.

Spisak glavnih aktera predstavljenih na dijagramu slučajeva korišćenja, kao i njihove aktivnosti.

Uloge Actora:

- Korisnik
- Administrator

Aktivnosti Korisnika:

- Pretraga
- Izbor rastorana
- Kreiranje porudžbine
- Plaćanje
- Prijava
- Registracija
- Odjava

Aktivnosti Administratora:

- Prijava
- Odjava
- Dodavanje novih proizvoda

Da bi korisnik mogao da naruči, a potom da plati, neophodno je da se prvo da se prijavi na sistem sa ispravnim podacima. Ako korisnik nema već kreiran nalog, neophodno je da se prvo registruje u sistem, pa potom prijavi.

Kako bi se bolje prikazalo funkcionisanje aplikacije u realnim uslovima, u nastavku su predstavljeni primeri scenarija koji se mogu dešavati tokom korišćenja sistema.

Scenario: Prijava na sistem

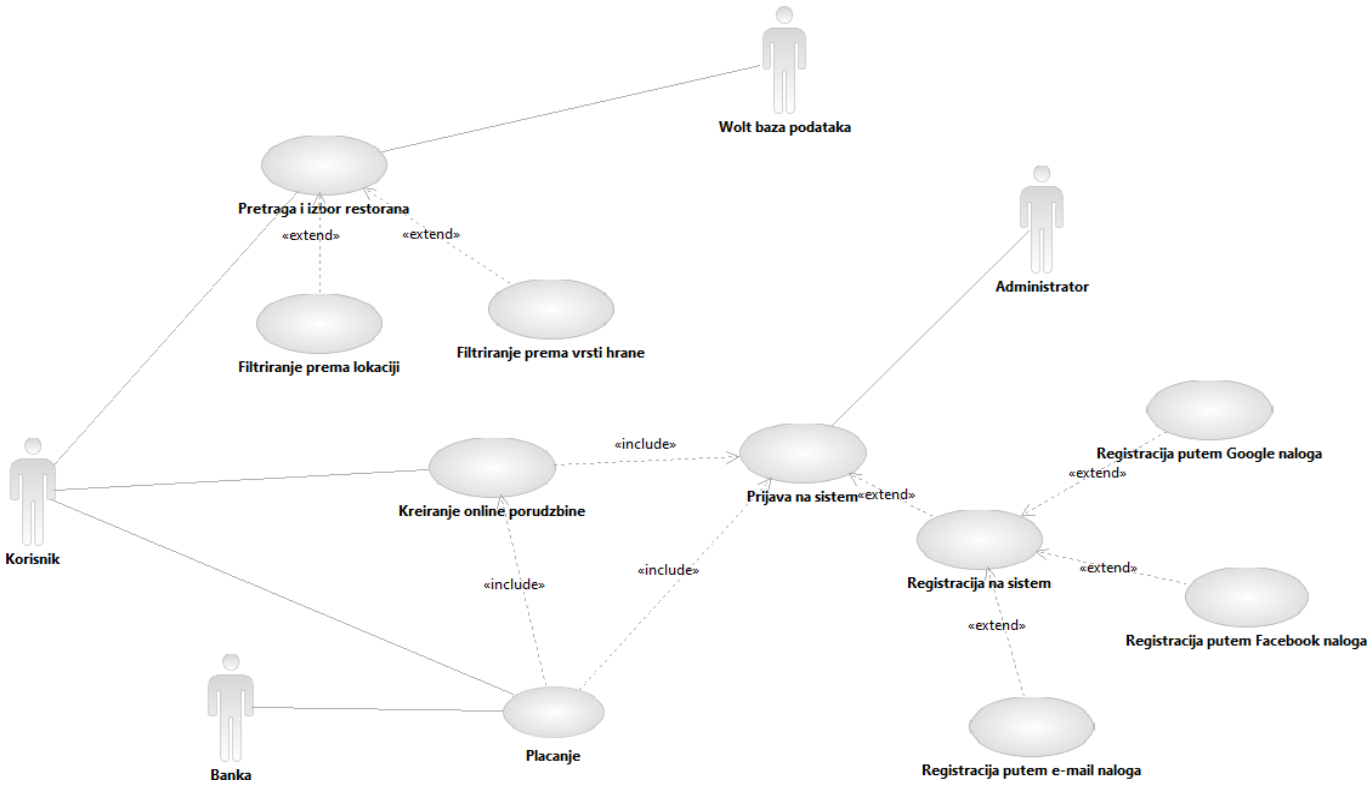
Preduslovi: Nema
Posledice: Korisnik je ulogovan na sistem
Tekst:

1. Korisnik dolazi na početnu stranu sajta
2. U okviru glavnog menija Korisnik selektuje opciju "Profil", nakon čega mu se otvara formular za unos korisničkog imena i lozinke
3. Korisnik unosi korisničko ime i lozinku
4. Ukoliko je korisničko ime ili lozinka pogrešna, Korisnik se vraća na 3. Korak
5. Ukoliko su korisničko ime i lozinka ispravni, Korisnik je uspešno prijavljen na sistem kao Korisnik

Scenario: Naručivanje hrane

Preduslovi: Da je korisnik prijavljen na sistem
Posledice: Naručena hrana
Tekst:

1. Korisnik dolazi na početnu stranicu
2. Na početnoj stranici bira restoran i hranu koju želi da naruči
3. Dodaje hranu u korpu
4. Selektuje opciju za korpu nakon čega mu se otvara korpa
5. U korpi selektuje dugme za naručivanje
6. Unosi podatke za plaćanje i za dostavu

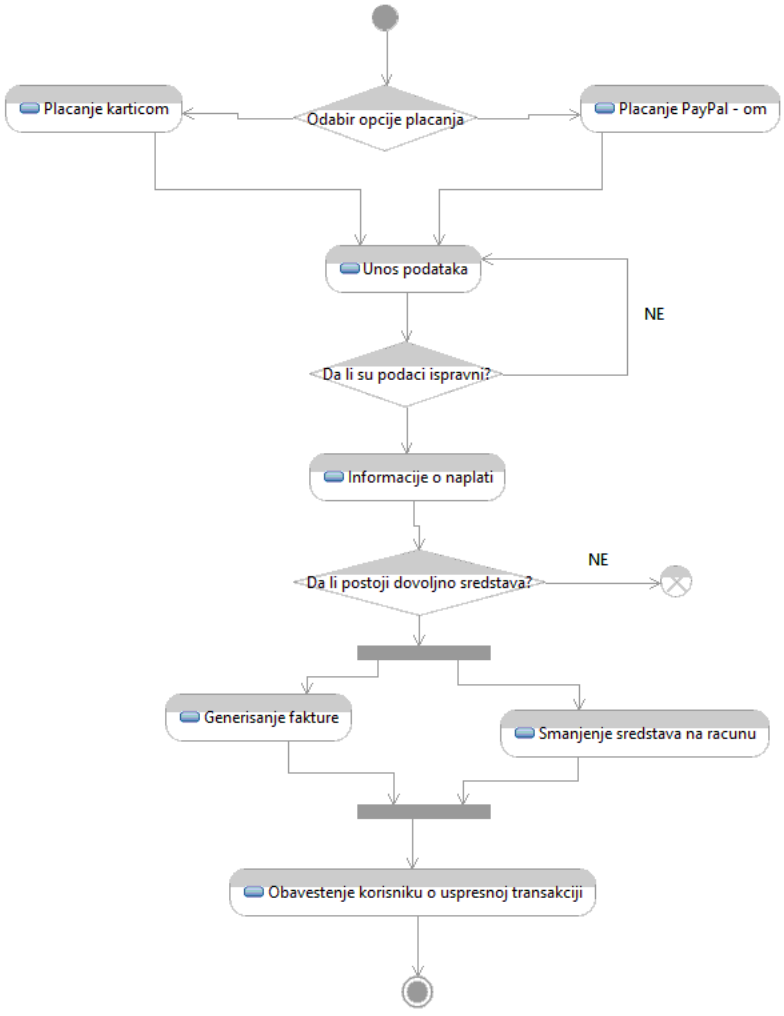


Slika 4. Dijagram slučaja korišćenja

3.3. Dijagram aktivnosti

Dijagrami aktivnosti su grafički prikazi tokova rada sastavljenih od koraka aktivnosti i akcija, sa podrškom za izbor, ponavljanje i istovremeno izvršavanje. Dijagrami aktivnosti namenjeni su modelovanju računarskih i organizacionih procesa (tokova rada), kao i tokova podataka koji se prepliću sa povezanim aktivnostima.

Objektni čvorovi čuvaju podatke koji se unose u izvršne čvorove i izlaze iz njih, i kreću se preko ivica toka objekata. Kontrolni čvorovi određuju redosled izvršavanja čvorova putem ivica kontrolnog toka. Drugim rečima, iako dijagrami aktivnosti prvenstveno prikazuju opšti kontrolni tok, oni mogu takođe uključivati elemente koji prikazuju tok podataka između aktivnosti kroz jednu ili više skladišta podataka.



Slika 5. Dijagram aktivnosti za plaćanje

3.4.Arhitekture veb aplikacije

MVC *(Model-view-controller*) arhitektura je projektni uzorak (*pattern*) koji se obično koristi za razvoj korisničkih interfejsa. Počiva na ideji o ponovnoj upotrebi već postojećeg softverskog koda, olakšavanju razvoja i kasnijem održavanju aplikacionog softvera metodom razdvajanja na posebne komponente: model, prikaz podataka (*view*) i kontrolor, pri čemu je komponenta za prikaz informacija odvojena od interakcije korisnika sa tim informacijama.

MVC arhitektura se sastoji od određenih komponenti u kojima je svaka zadužena za obavljanje specifičnih funkcija. Takvo rešenje je mnogo bolje u odnosu na ranije rešenja u kojima se sve nalazilo na jednom mestu, što je dovodilo do otežanog čitanja izvornog koda, otežanog pronalaženja i ispravljanja grešaka, kao i do znatno manje funkcionalnosti i fleksibilnosti. MVC ja kao okvir softverske arhitekture nastao pre nego što je izmišljen veb pretraživač, i u početku je korišćen samo za kreiranje grafičkog korisničkog okruženja.

Model predstavlja stanje sistema koje se može promeniti izvršavanjem operacija nad objektima u modelu podataka. Za model nije bitno kako se upravlja podacima, niti način na koji se podaci prikazuju korisniku.

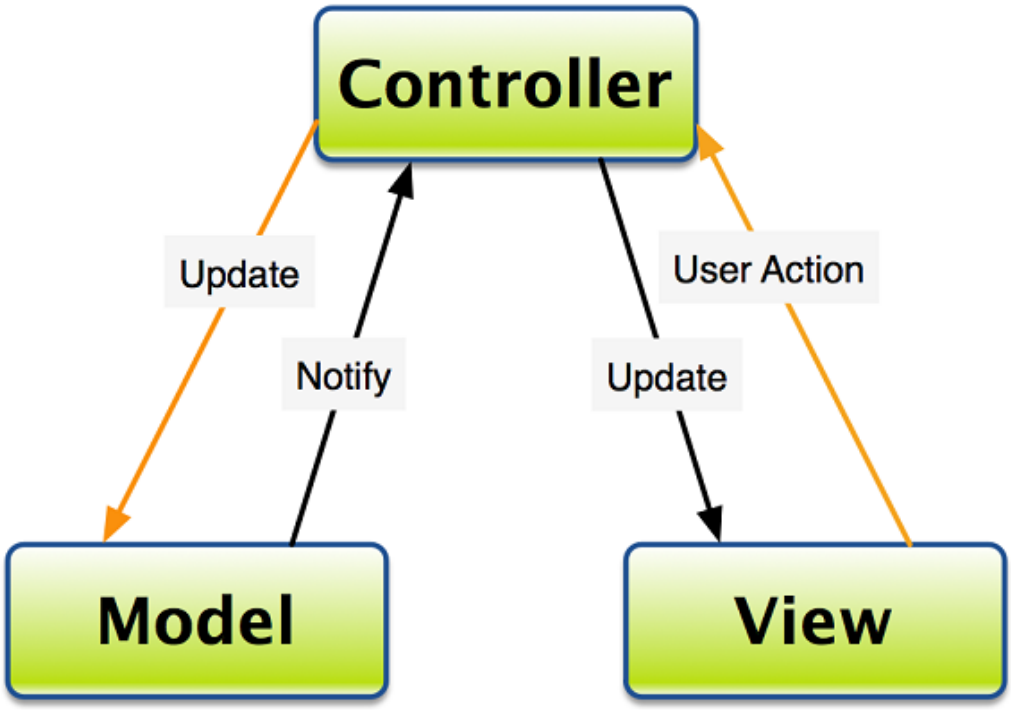
Prikaz (*View*) prikazuje korisniku stanje modela, obezbeđuje korisnički interfejs, pomoću koga korisnik unosi podatke i poziva odgovarajuće operacije koje treba da se izvrše nad podacima.

Kontroler osluškuje i prihvata zahteve klijenta za izvršenje neke operacije. Zatim poziva operaciju koja je definisana u modelu, i ukoliko model promeni stanje, prikazuje ga korisniku.

Norveški informatičar Trigvi Riensku (*Trygve Reenskaug*) je tokom posete Ziroksu sedamdesetih godina prvi put predstavio MVC arhitekturu primenjenu na Smalltalk-76. On je MVC kao apstraktni prikaz arhitekture softvera definisao na sledeći način:

- Model predstavlja podatke. Podaci mogu biti samo jedan objekat ili struktura objekata. Veza između modela i onoga što vidi korisnik trebala bi da bude 1 na 1. Komunikacija između modela i spoljašnjeg sveta bi trebao da se obavlja uz pomoć kontrolora.
- Pogled ili vizualna prezentacija modela, je povezana sa modelom ili jednim njegovim delom. Ponaša se kao filter, koji iz modela naglašava određene vrednosti koji opisuju podatke, dok neke druge potiskuje i tako filtrirane podatke prikazuje korisniku. Takođe, ima mogućnost ažuriranja modela, slanjem odgovarajućih zahteva korisnika.
- Kontrolor je veza između korisnika i podataka. On korisniku, u zavisnosti od njegovog zahteva prikazuje podatke.

Zatim su osamdesetih godina Džejms Altof (James Althoff) i drugi upotrebili verziju MVC-a za Smalltalk-80 biblioteku klasa. Tek u članku iz 1988-e MVC je predstavljen kao poseban pojam.

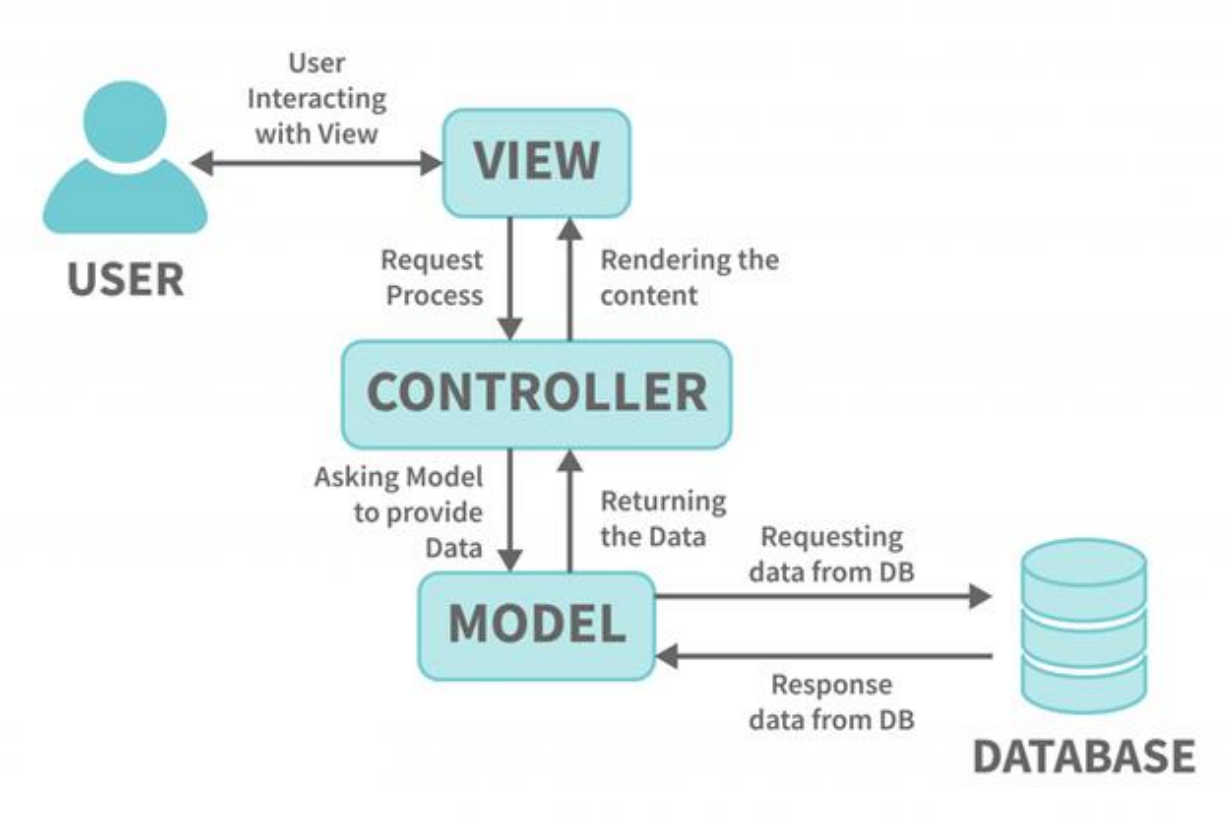


Slika 6. MVC šablon

MVC je široko prihvaćen kao projektni uzorak za aplikacije na svetskoj mreži, u svim značajnijim programskim jezicima. Nekoliko komercijalnih i nekomercijalnih programskih okvira (*framework*) koriste ovaj projektni uzorak. Svaki od njih varira u interpretaciji načina na koji su podeljene odgovornosti između klijenta i servera. Kod onih ranijih, ceo model, prikaz i kontrolor su se nalazi na serveru. Klijent je samo slao zahtev (na primer preko formulara), a potom primao kompletnu novoformiranu veb stranicu.

U veb aplikacijama komponenta za prikaz sadrži: HTML, CSS, Javaskript, XML ili JSON i druge. U veb aplikacijama kontrolor predstavlja prvi sloj koji se poziva kada veb pretraživač pozove neku veb adresu.

Napredak tehnologije je omogućio da se MVC komponente delom izvršavaju kod klijenta (Ajax).



Slika 7. MVC arhitektura u veb aplikacijama

Radi lakšeg pregleda funkcionalnosti aplikacije, u nastavku je dat spisak kontrolera koji su implementirani u okviru MVC arhitekture. Svaki kontroler ima jasno definisanu ulogu i odgovornost u sistemu, a tabela prikazuje njihova imena i opis funkcionalnosti koje obavljaju.

Redni broj	**Naziv kontrolera**	**Opis**
1.	MainController	Glavni kontroler, svi ostali kontroleri ga nasleđuju, sadrži funkcije za rad sa sesijama i konekciju sa bazom
2.	OrderController	Vrši naručivanje
3.	ProductController	Služi za prikaz proizvoda

Tabela 1. Spisak kontrolera u aplikaciji

3.5.Dijagram sekvenci

Dijagrami sekvenci su važan deo grupe dijagrama interakcije. Dijagrami sekvence se koristi zajedno sa dijagramom komunikacije i vremenskim dijagramom da bi se precizno prikazalo kako delovi sistema međusobno interaktuju. Dijagrami sekvence pokazuju prave informacije i logični su ljudima.

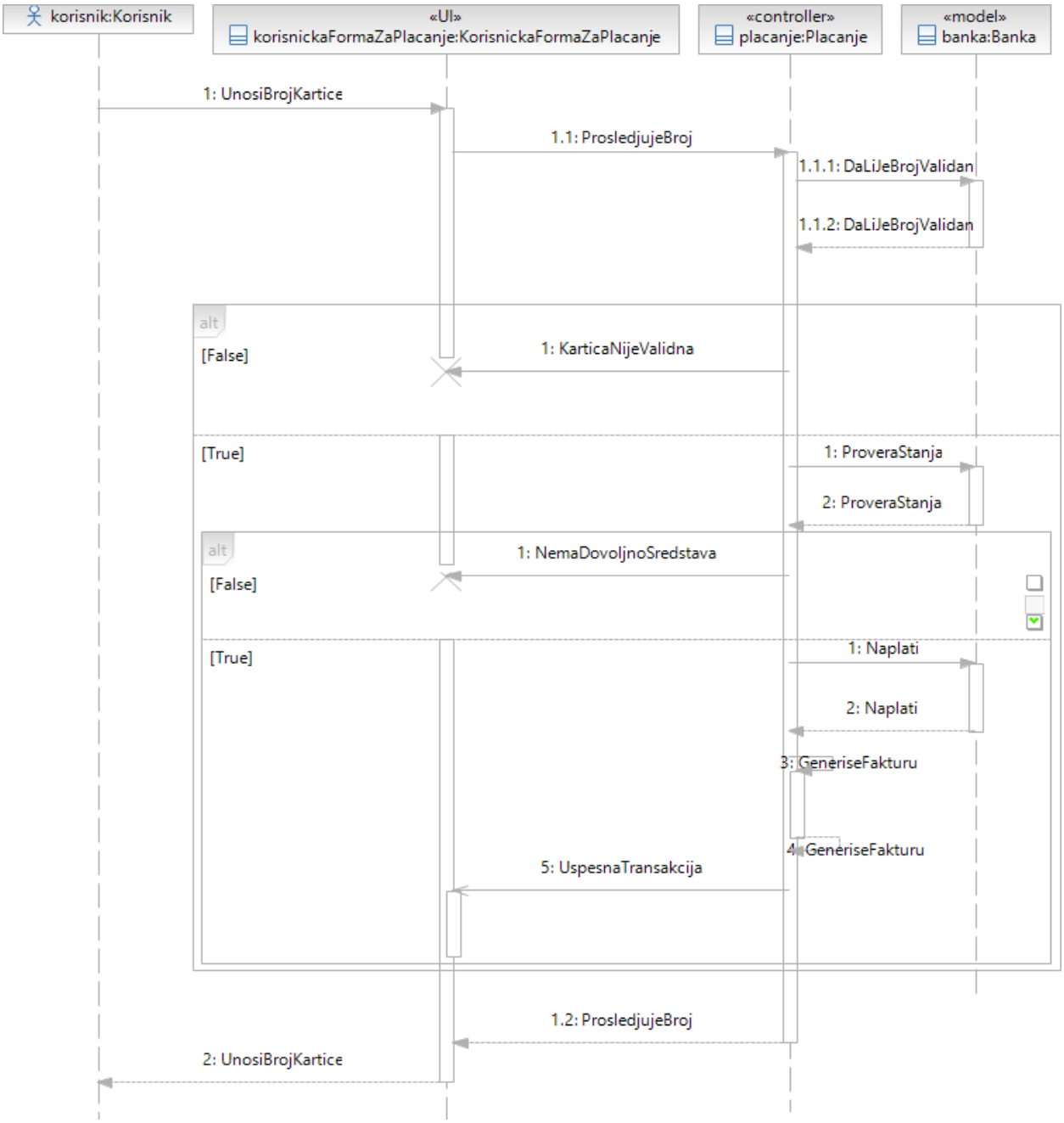
Cilj dijagrama sekvenci je otkrivanje redosleda interakcija između delova sistema. Korišćenjem dijagrama sekvenci, može se opisati koje interakcije i kojim redosledom će se dogoditi kada se izvrši određen slučaj korišćenja.

Dijagram sekvenci se sastoji od kolekcije učesnika i delova sistema koji su u međusobnoj interakciji za vreme sekvence. Neobazirući se na vertikalnu poziciju, učesnici se uvek ređaju horizontalno, bez međusobnog preklapanja.

Svaki učesnik ima odgovarajuću životnu liniju koja ide niz stranu. Životna linija pokazuje da učesnik postoji u datom trenutku u sekvenci, i značajna je samo kada se učesnik stvara i/ili briše za vreme sekvence.

Dijagram sekvenci pokazuje kojim se redosledom interakcije dešavaju, tako da je vreme veoma važan faktor. Redosled kojim su interakcije raspoređene na dijagramu sekvenci ukazuju kojim će se redosledom interakcije dešavati u vremenu. Vreme se na dijagramu sekvenci odnosi na redosled, a ne na dužinu trajanja.

Najmanji deo interakcije je događaj. Događaj je bilo koji trenutak interakcije u kome se nešto dešava. Događaji su osnova signala i poruka. Interakcija na dijagramu sekvenci se dešava kada jedan učesnik odluči da pošalje poruku drugom učesniku. Na poruku treba gledati kao na događaj koji pošiljaoc poruke šalje primaocu poruke da bi on uradio nešto. Ponekad učesnik može poslati poruku sam sebi. Na primer, klasa poziva sama sebe tako što poziva svoje neke operacije.



Slika 8. Prikaz MVC arhitekture preko dijagrama sekvenci za plačanje

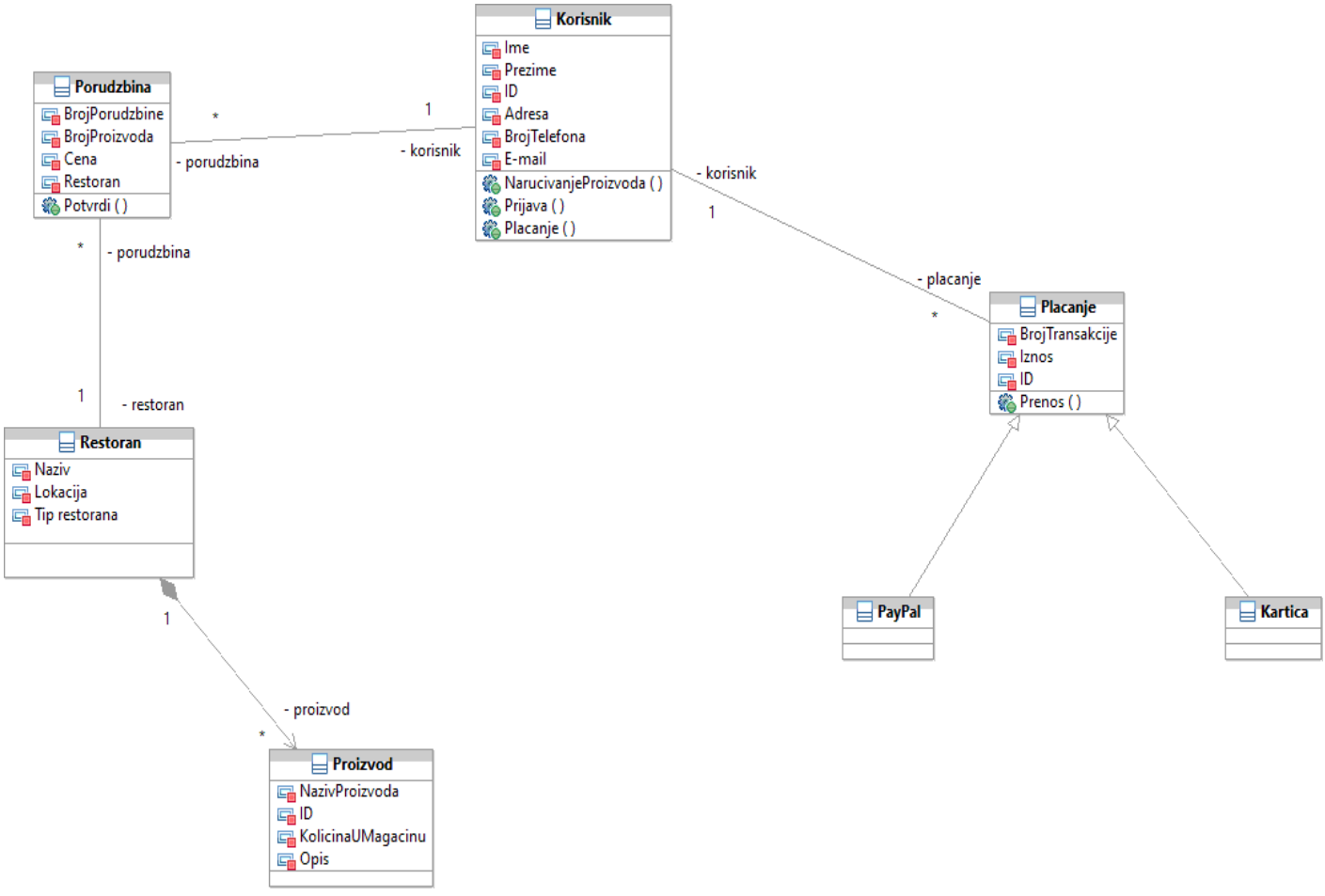
3.6. Logički dijagram klase

U softverskom inžinjerstvu, dijagram klasa ili klasni dijagram u objedinjenom jeziku za modelovanje (UML) je tip dijagrama statičke strukture koji opisuje strukturu sistema prikazivanjem klasa sistema, njihovih atributa, operacija (metoda) i odnosa između objekata.

Dijagram klasa je glavni gradivni blok objektno-orijentisanog modelovanja. Koristi se za opšte konceptualno modelovanje strukture aplikacije, kao i za detaljno modelovanje prevođenja modela u programski kod. Klase u dijagramu klasa predstavljaju glavne elemente, interakcije u aplikaciji i klase koje treba da budu isprogramirane.

U dijagramu, klase su predstavljene kutijama koje se sastoje iz tri dela:

- Gornji deo sadrži naziv klase. Ispisan je podebljano i centrirano.
- Srednji deo sadrži atribute klase.
- Donji deo sadrži operacije koje klasa može da izvrši.

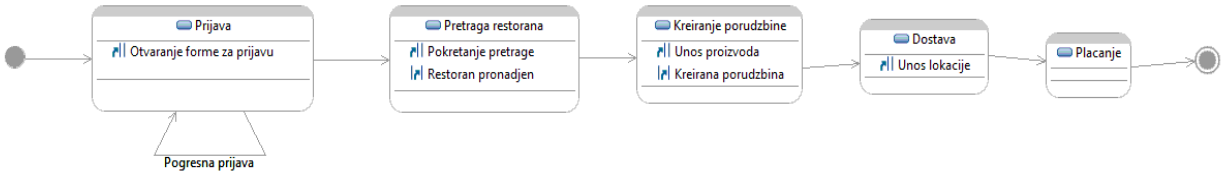


Slika 9. Logički dijagram klase aplikacije

3.7.Dijagram stanja

Dijagram stanja je dijagram koji se koristi na polju kompjuterskih nauka, koji predstavlja ponašanje sistema koji je sastavljen od konačnog broja stanja. Postoje mnogi oblici dijagrama stanja, koji se neznatno razlikuju i imaju različitu semantiku.

Dijagrami stanja se koriste za opisivanje ponašanja sistema. Oni mogu da opišu moguća stanja objekta kako se događaji pojavljuju. Svaki dijagram obično predstavlja objekte jedne klase i prati različita stanja tih objekata kroz sistem. Dijagram stanja se može upotrebiti da grafički predstavi automate konačnih stanja.

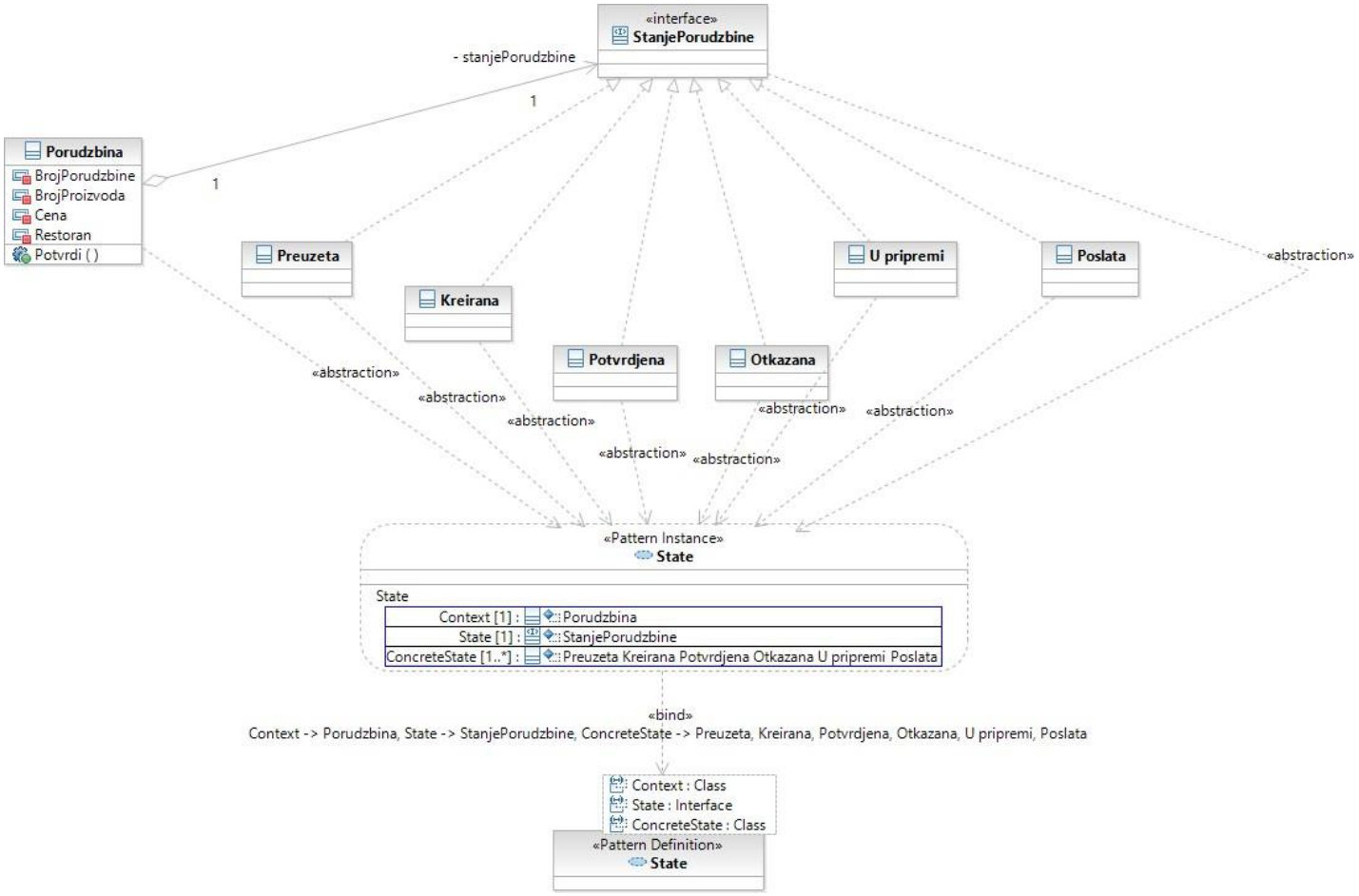


Slika 10. Dijagram stanja aplikacije

3.8.Obrasci projektovanja aplikacije

Obrasci projektovanja aplikacije ili dizajn paterni (*design pattern*) u računarskom inžinjeringu je stalno ponavljano rešenje učestalom problemu u softverskom dizajnu. U principu, ovo nije krajnje rešenje koje se može direktno pretvoriti u izvorni kod, nego je više opis ili šablon po kojem se problem može rešiti u raznim situacijama. Objektno orijentisani dizajn paterni obično pokazuju odnos između raznih klasa ili objekata, bez detaljnog opisa tih klasa ili objekata.

Obrasci projektovanja aplikacije mogu ubrzati razvoj softvera tako što daju isprobane i testirane programske paradigme.

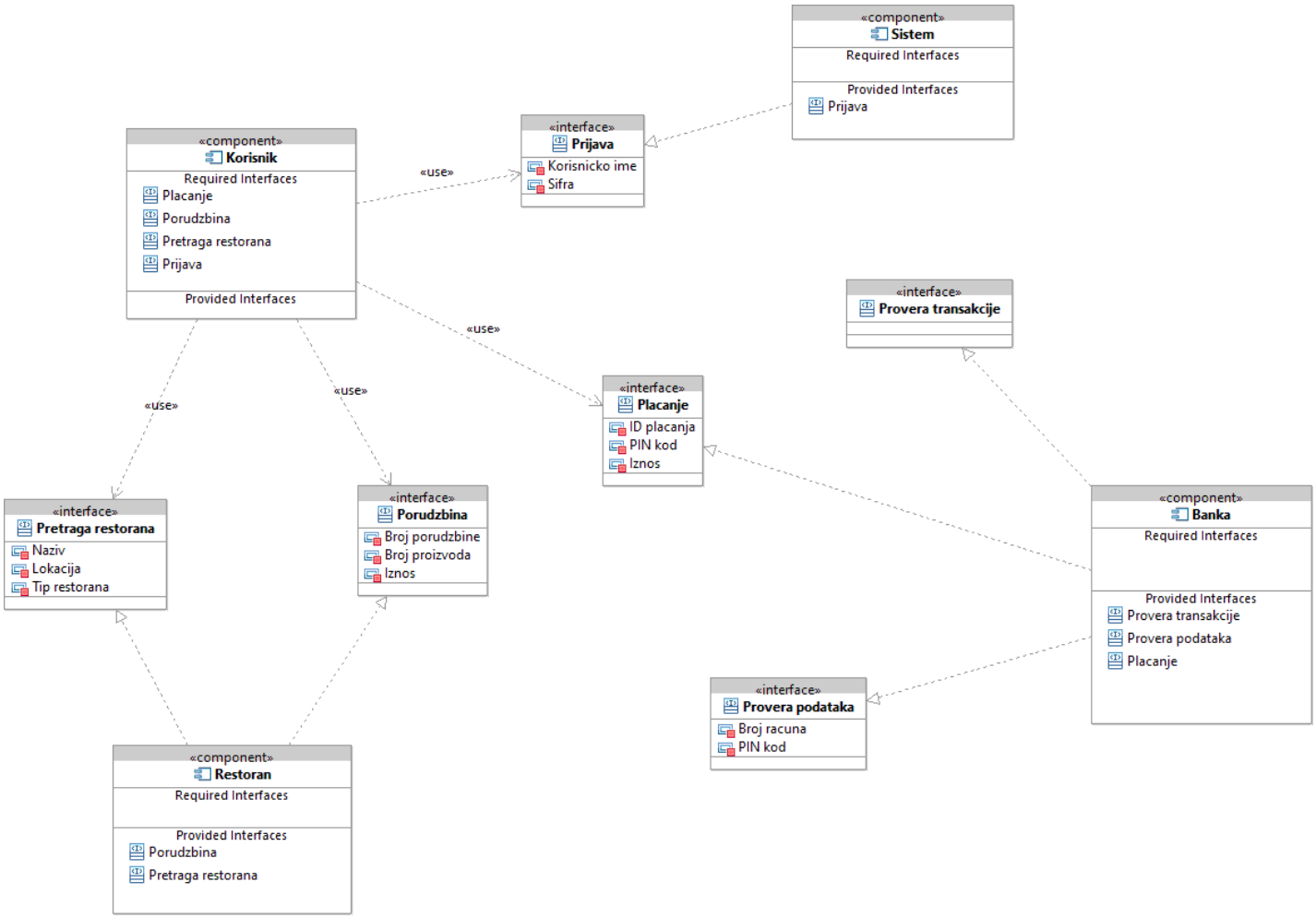


Slika 11. State dizajn patern

3.9.Dijagram komponenti

Dijagram komponenti modeluje strukturu softvera, ukazajući na zavisnost između tih softverskih komponenata. Crta se kao graf komponenti, ponekad grupisanih u pakete ili podsisteme, na kojem se prikazuje zavisnost između njih.

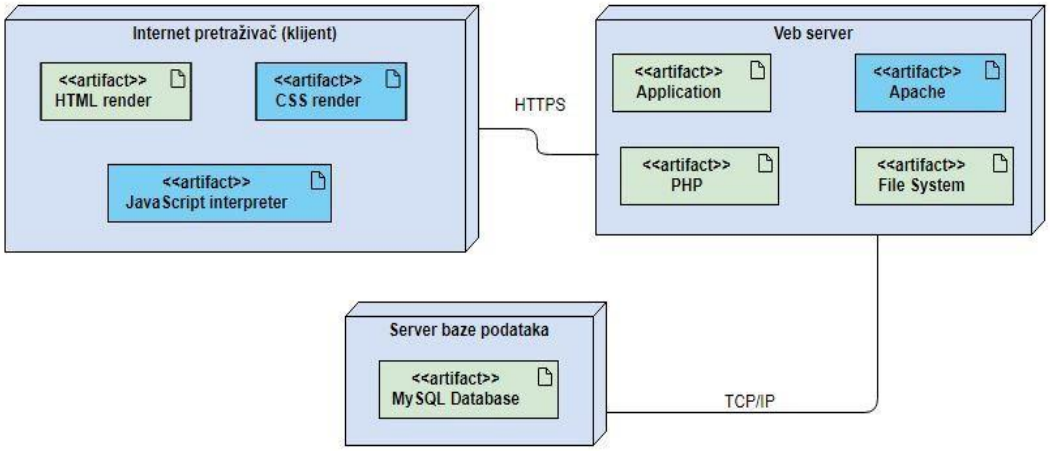
Namena dijagrama komponenti je da definiše softverske module i njihove međusobne relacije. Svaka komponenta softvera je deo koda koji se nalazi u memoriji na nekom delu hardvera i mora da se definiše interfejs koji omogućava drugim komponentama da komuniciraju sa njom. Navodeći ključne elemente kao što su komponente, interfejsi komponentni i zavisnosti može se opisati fizička impelementacija sistema u terminima softverskih modula i njihovih međusobnih odnosa.



Slika 12. Dijagram komponenti aplikacije

3.10. Dijagram isporuke

Dijagram isporuke prikazan je na slici 13, na kojoj su prikazane konfiguracije softverskih i hardverskih kompnenti. U okviru ovog dijagrama dati su čvorovi sistema, koji su međusobno povezani protokolima putem kojih komuniciraju. Aplikacija podrazumeva primenu servera baze podataka, servera aplikacije i računare s pristupom internetu.



Slika 13. Dijagram isporuke

3.11. Dijagram strukture podataka

Name: users

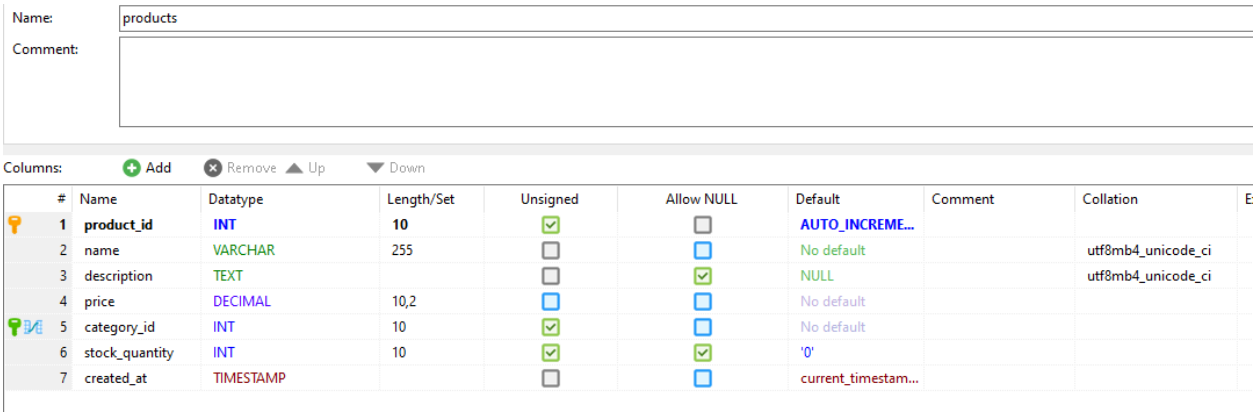
Comment:

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsigned	Allow NULL	Default	Comment	Collation
1	user_id	INT	10	☑	☐	AUTO_INCREME...		
2	first_name	VARCHAR	255	☐	☐	No default		utf8mb4_unicode_ci
3	last_name	VARCHAR	255	☐	☐	No default		utf8mb4_unicode_ci
4	email	VARCHAR	255	☐	☐	No default		utf8mb4_unicode_ci
5	password	VARCHAR	255	☐	☐	No default		utf8mb4_unicode_ci
6	phone	VARCHAR	20	☐	☑	NULL		utf8mb4_unicode_ci
7	created_at	TIMESTAMP		☐	☐	current_timestam...		

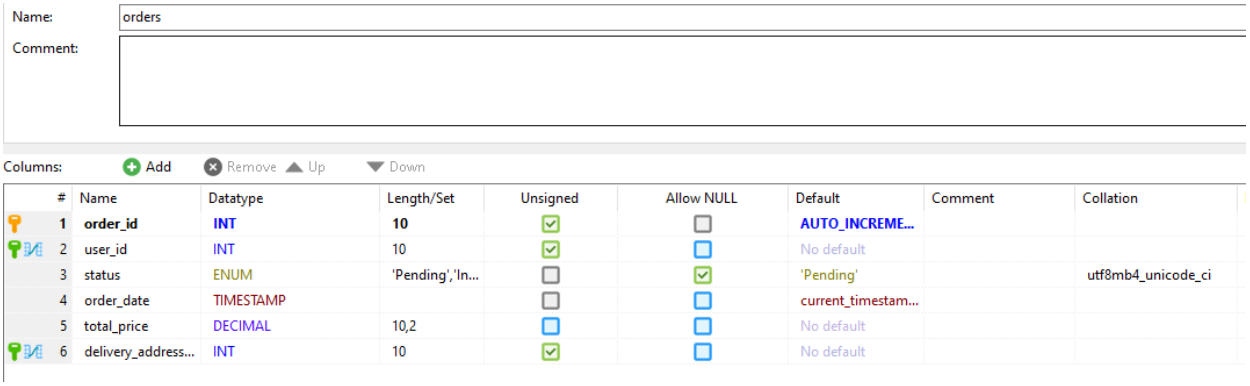
Slika 14. SQL tabela users

Tabela *users* služi za čuvanje podataka o korisnicima kada se prijave na sistem. Tu se čuvaju ime, prezime korisnika, email i šifra koji služe za prijavu, broj telefona koji služi za poziv tokom dostave i kada je taj nalog kreiran.



Slika 15. SQL tabela products

Tabela *products* sadrži podatke vezane za proizvode koji se nalaze u ponudi. U njoj se nalaze ime, opis cena proizvoda, i u kojoj količini je taj proizvod dostupan.



Slika 16. SQL tabela orders

Tabela *orders* služi za podatke o porudžbini. Uzima id korisnika koji je napravio tu porudžbinu, kada je poručeno, kolika je ukupna cena porudžbine i uzima id adrese za dostavu iz naredne tabele zbog adrese.

Name: delivery_addresses

Comment:

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsigned	Allow NULL	Default	Comment	Collation
1	address_id	INT	10	☑	☐	AUTO_INCREME...		
2	user_id	INT	10	☑	☐	No default		
3	address_line1	VARCHAR	255	☐	☐	No default		utf8mb4_unicode_ci
4	address_line2	VARCHAR	255	☐	☑	NULL		utf8mb4_unicode_ci
5	city	VARCHAR	100	☐	☐	No default		utf8mb4_unicode_ci
6	postal_code	VARCHAR	20	☐	☐	No default		utf8mb4_unicode_ci
7	country	VARCHAR	100	☐	☐	No default		utf8mb4_unicode_ci
8	phone	VARCHAR	20	☐	☑	NULL		utf8mb4_unicode_ci

Slika 17. SQL tabela delivery_addresses

Tabela *delivery_addresses* sadrži podatke o adresi korisnika koji je naručio, grad, državu i broj telefona radi dostave.

Name: categories

Comment:

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsigned	Allow NULL	Default	Comment	Collation
1	category_id	INT	10	☑	☐	AUTO_INCREME...		
2	name	VARCHAR	255	☐	☐	No default		utf8mb4_unicode_ci
3	description	TEXT		☐	☑	NULL		utf8mb4_unicode_ci

Slika 18. SQL tabela categories

Tabela *categories* sadrži ime i opis skupa proizvoda sa pod istom kategorijom.

Name: delivery_history

Comment:

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsigned	Allow NULL	Default	Comment	Collation
1	delivery_id	INT	10	☑	☐	AUTO_INCREME...		
2	order_id	INT	10	☑	☐	No default		
3	delivery_date	TIMESTAMP		☐	☐	current_timestam...		
4	delivery_status	ENUM	'Pending','D...	☐	☑	'Pending'		utf8mb4_unicode_ci

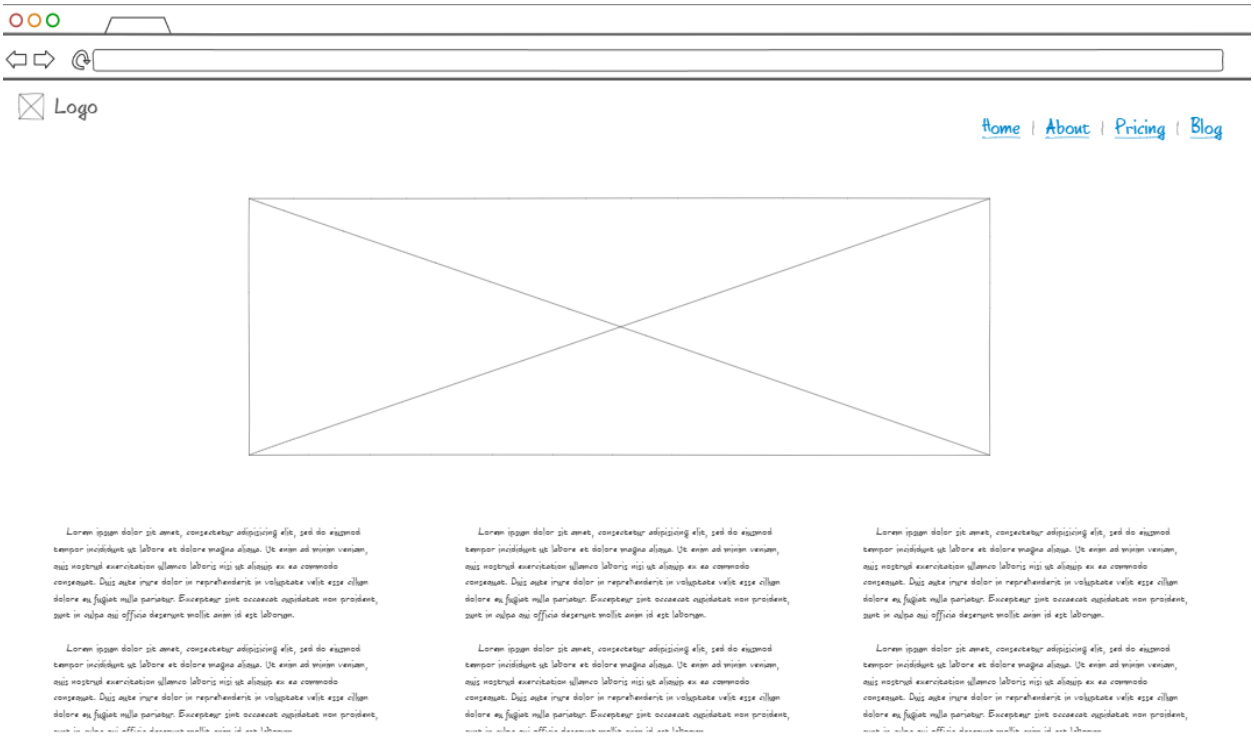
Slika 19. SQL tabela delivery_history

Tabela delivery_history sadrži istoriju dostava sa brojem porudžbine i datumom.

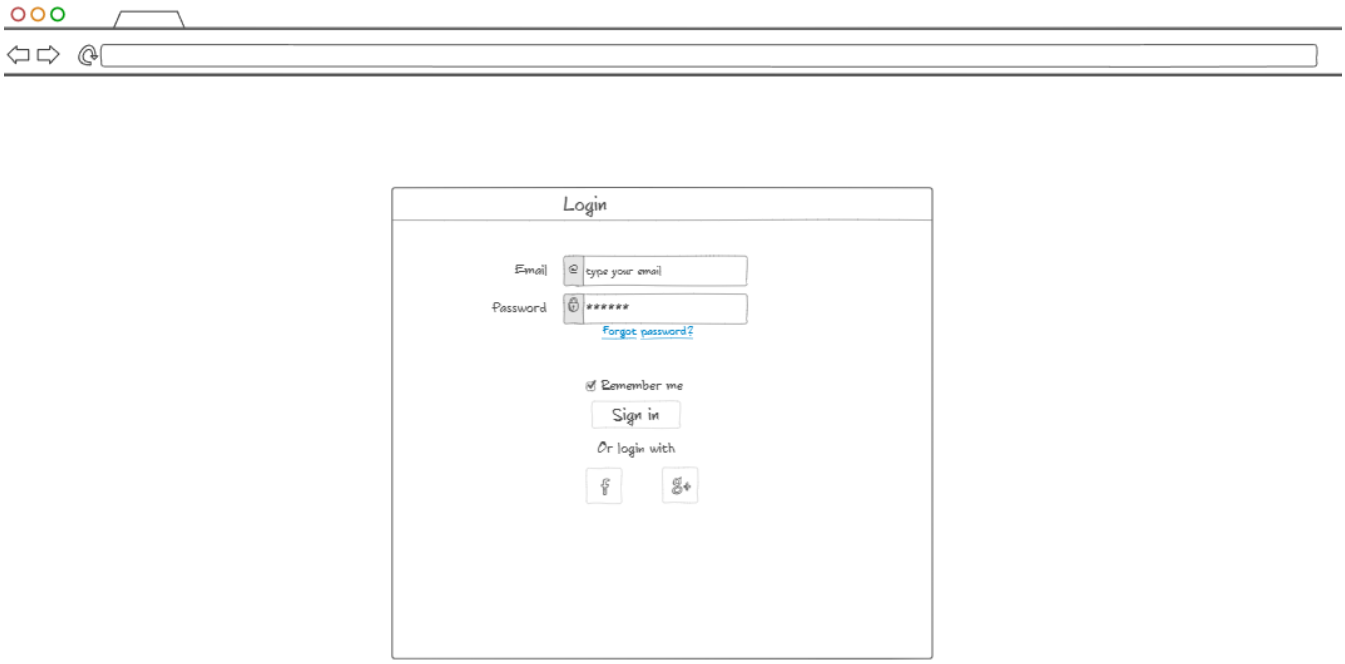
3.12. Projektovanje sloja aplikacije

U okviru ovog poglavlja prikazan je projekat sloja korisničkog interfejsa aplikacije. Korisnički interfejs predstavlja posrednika između korisnika i sistema, a njegov cilj je da omogući jednostavno, intuitivno i efikasno korišćenje aplikacije. Prilikom projektovanja posebna pažnja posvećena je preglednosti, jednostavnoj navigaciji i izgledu interfejsa, kako bi aplikacija bila pogodna za upotrebu na različitim uređajima.

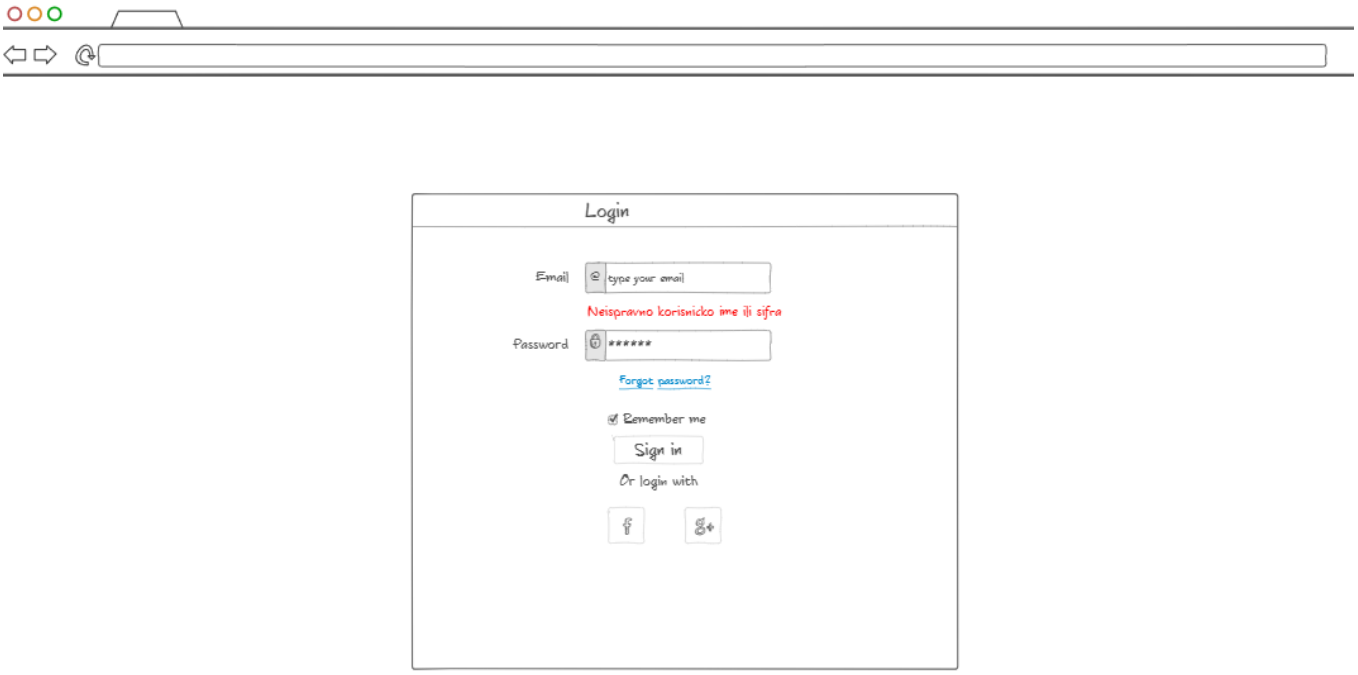
Na sledećim slikama prikazani su osnovni ekrani aplikacije, uključujući početnu stranicu, formu prijave, spisak restorana, opise posebnih restorana i njihove ponude, korpu za poručivanje i formu za plaćanje. Svaki od ovih ekrana dizajniran je tako da jasno prikaže dostupne funkcionalnosti i omogući korisniku da lako obavi željenu akciju.



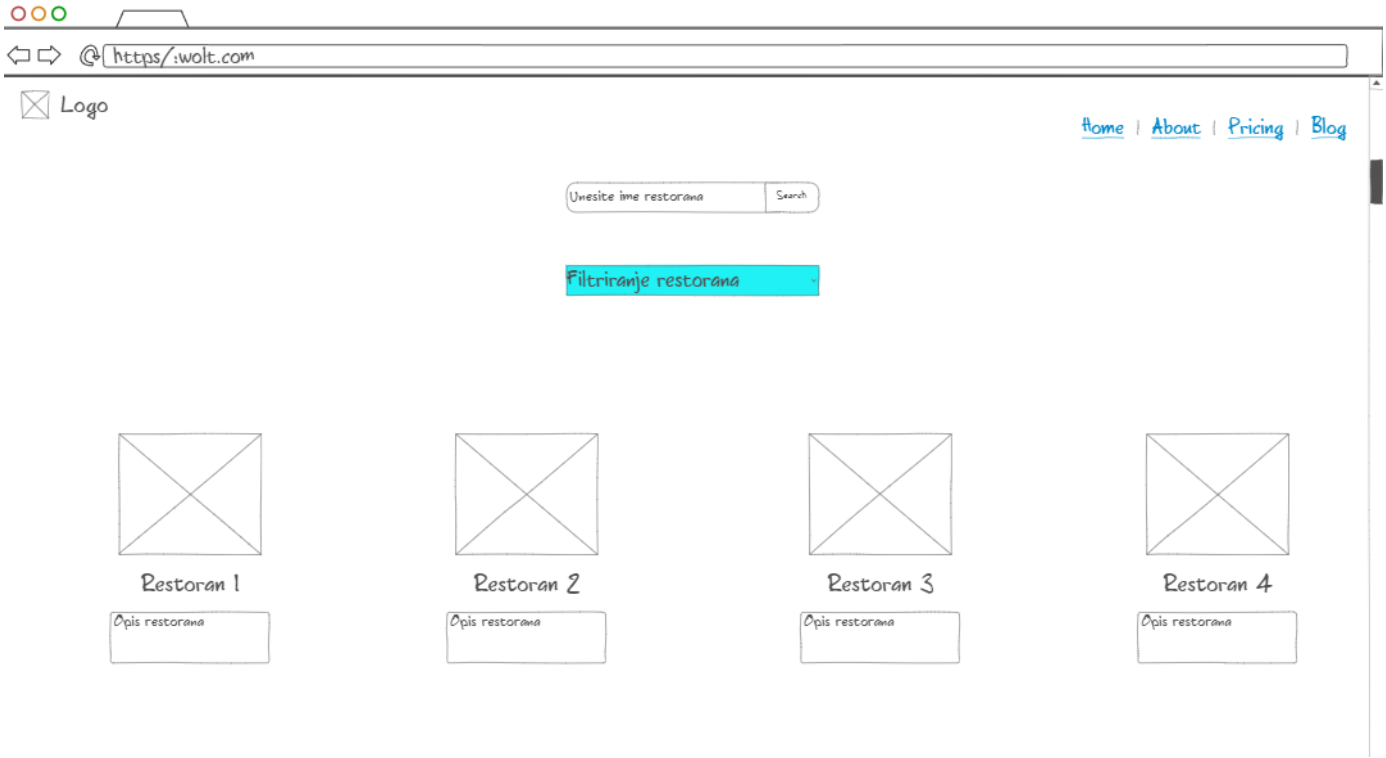
Slika 20. Korisnička forma "Početna strana"



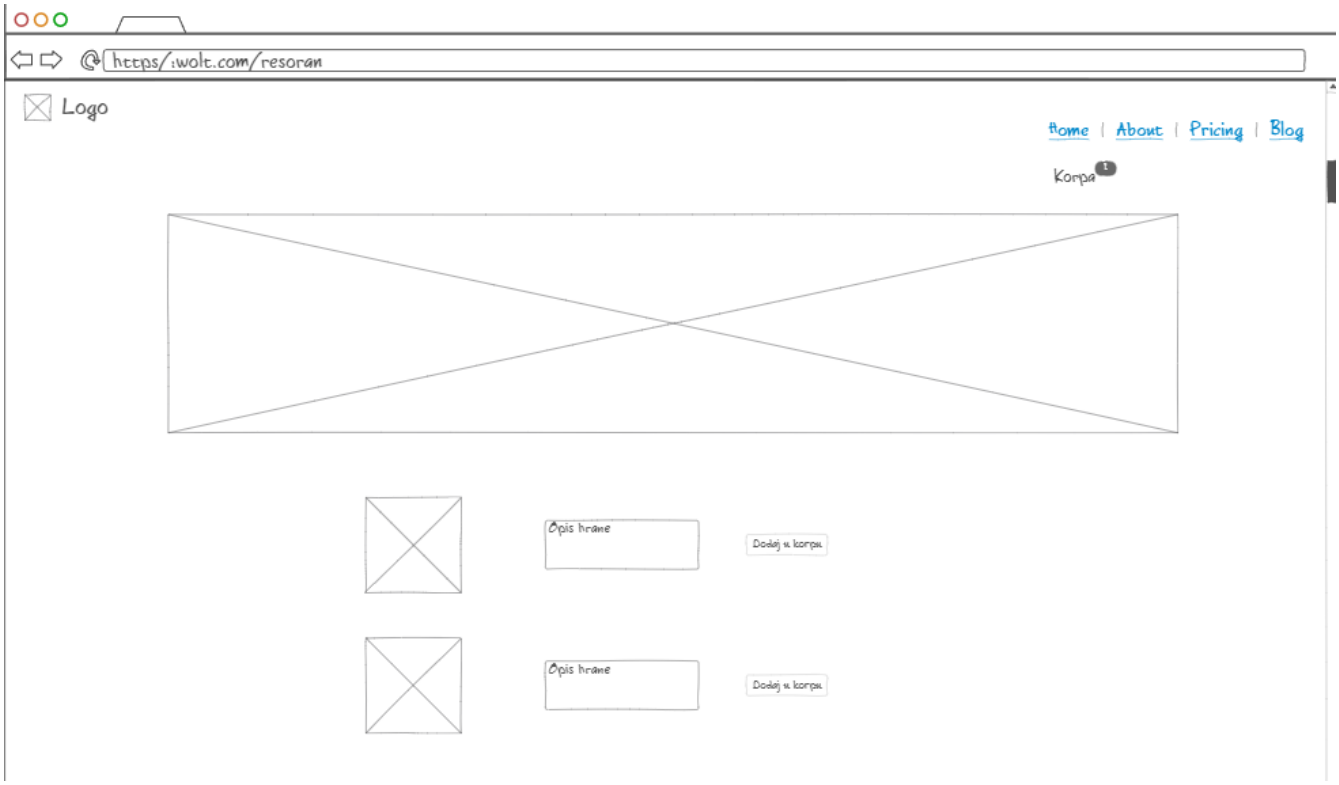
Slika 21. Korisnička forma “Prijava”



Slika 22. Korisnička forma “Greška prilikom prijave”

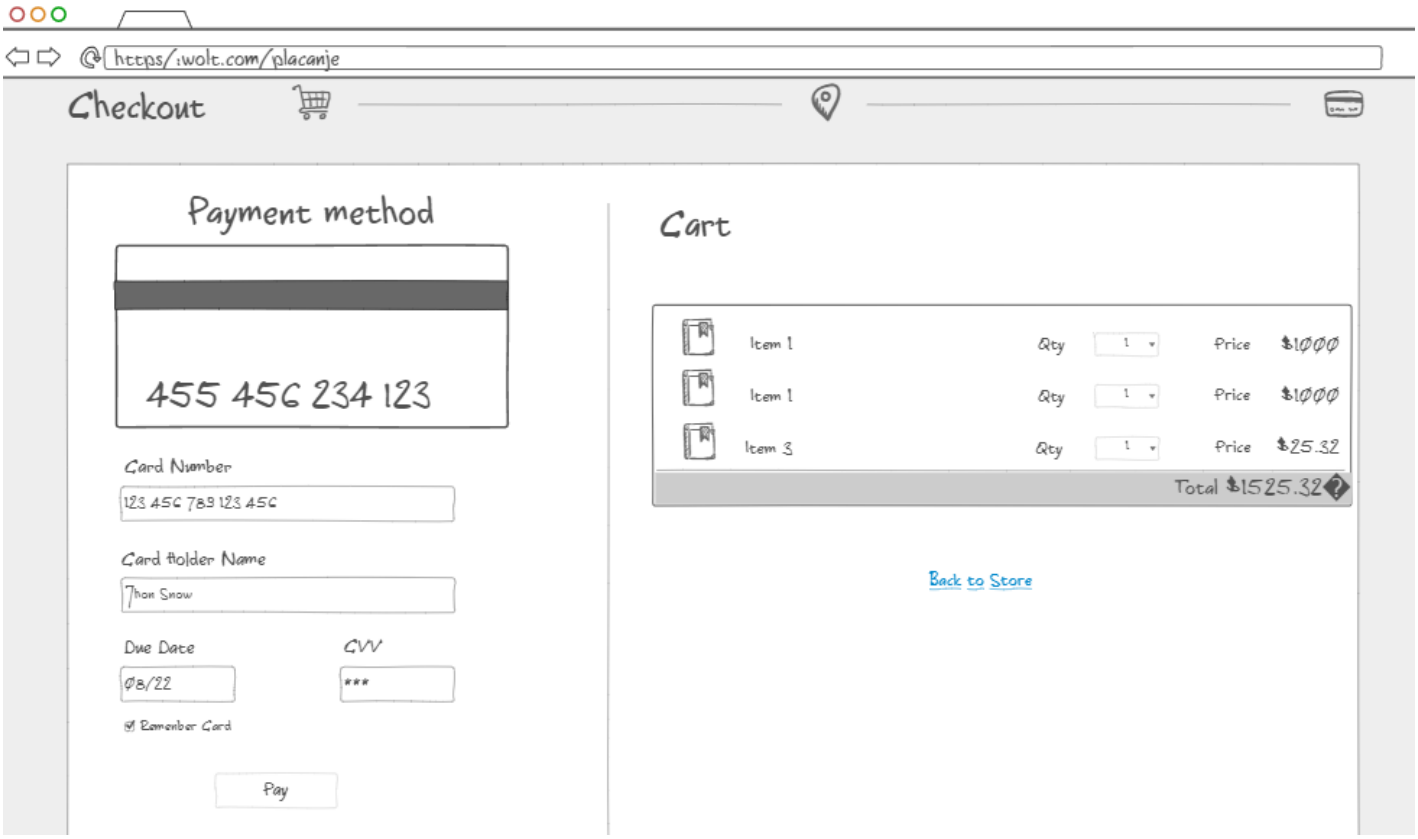


Slika 23. Korisnička forma “Biranje restorana”



Slika 24. Korisnička forma “Restoran”

Slika 25. Korisnička forma “Korpa”



Slika 26. Korisnička forma “Plaćanje”

4. Zaključak

U ovom radu prikazan je proces razvoja moderne veb aplikacije za naručivanje hrane. U okviru rada izrađeni su UML dijagrami koji jasno ilustruju strukturu i ponašanje sistema, definisana je SQL baza podataka kao osnova za pouzdano čuvanje i manipulaciju podacima, implementirana je MVC arhitektura radi bolje organizacije koda i razdvajanja slojeva aplikacije, kao i projekcija korisničkog interfejsa sa fokusom na preglednost i jednostavnost korišćenja.

Budući tok razvoja mogao bi obuhvatiti unapređenje postojećeg sistema kroz uvođenje naprednih funkcionalnosti, kao što su implementacija obaveštenja u realnom vremenu i korišćenje mobilnih aplikacija paralelno sa veb aplikacijom. Takođe, moglo bi se razmotriti korišćenje savremenih arhitekturnih pristupa, poput MVVM (*Model–View–ViewModel*) ili MVP (*Model–View–Presenter*) obrazaca, kao i prelazak sa tradicionalnih use case dijagrama na User Story-je i User Flow dijagrame, koji su danas zastupljeniji u agilnim metodologijama razvoja.

Na ovaj način, rad pruža osnovu za dalje unapređenje i modernizaciju sistema, dok istovremeno pokazuje primenu savremenih tehnologija u razvoju veb aplikacija koje imaju praktičnu primenu u realnom okruženju.

Literatura

[1] Njeguš, A. Metodologije razvoja softvera. Prezentacije sa predavanja. Univerzitet Singidunum, Beograd.

[2] Veinović M., Šimić G., Jevremović A., Tair M. Baze podataka. Udžbenik, Univerzitetska izdanja. Univerzitet Singidunum, Beograd.

[3] Kojić N. Veb dizajn: HTML, CSS i JavaScript. Udžbenik, Univerzitetska izdanja. Univerzitet Singidunum, Beograd.

[4] Ćuk A. Praktikum – napredno softversko programiranje. Prezentacije sa predavanja. Univerzitet Singidunum, Beograd.