SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Hrvoje Bogeljić

PRILAGODLJIVA WEB APLIKACIJA ZA POSREDOVANJE PRI ONLINE NARUDŽBAMA

ZAVRŠNI RAD

Varaždin, 2012.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Hrvoje Bogeljić

Izvanredni student

Broj indeksa: S-39853/10-III-Izv

Smjer: Primjena informacijske tehnologije u poslovanju

Stručni studij

PRILAGODLJIVA WEB APLIKACIJA ZA POSREDOVANJE PRI ONLINE NARUDŽBAMA

ZAVRŠNI RAD

Mentor:
Doc. dr. sc. Markus Schatten

Varaždin, rujan 2012.

Sadržaj

1. Uvod

Tema ovog rada je realizacija prilagodljive web aplikacije za posredovanje pri online narudžbama hrane i pića iz ugostiteljskih objekata. Motivacija za odabir ove teme proizašla je iz spoznaje kako je online naručivanje postalo globalni trend, pa je tako i online naručivanje hrane u svijetu postala uobičajena pojava. U trenutku stvaranja ove ideje u Republici Hrvatskoj postojao je samo jedan servis koji nudi usluge takve vrste, dok u ovom trenutku djeluje već nekolicina takvih servisa, što govori u prilog kvaliteti ideje. Zanimljivo je napomenuti kako primjerice u SAD-u čak 85% ugostiteljskih objekata nudi uslugu online naručivanja, a veliki dio od njih i online plaćanje.

U nastavku ovog rada opisana je realizacija jedne takve aplikacije, te tehnologije na kojima se zasniva. Konkretno aplikacija je razvijena i testirana lokalno na razvojnom poslužitelju, pa je tako u prvom dijelu opisano postavljanje razvojnog poslužitelja. Zatim u istom dijelu opisane su i komponete korištenog paketa aplikacija razvojnog poslužitelja, koje su i služile kao osnova za razvoj ove aplikacije.

U drugom dijelu opisana je sama aplikacija, njena koncepcija i osnovna svrha. Prikazana je primjena korištenih tehnologija u realizaciji konkretne aplikacije.

U trećem dijelu rada date su korisničke upute, dok je u četvrtom dijelu opisana korisnička perspektiva aplikacije.

Nakon toga rad je zaključen kritičkim osvrtom na realizaciju aplikacije i procjenom naznaka za budući rad.

2. Razvojni poslužitelj

Internet aplikacije je moguće razvijati na lokalnom ili udaljenom poslužitelju. Ukoliko nemamo vlastiti razvojni (lokalni) poslužitelj, svaku promjenu u aplikaciji potrebno je učitati na javni (udaljeni) poslužitelj prije nego što je istu moguće testirati. Čak i uz brzu internet vezu, ovakav način razvoja predstavlja značajan gubitak vremena u fazi razvoja.
Na lokalnom računalu međutim testiranje je jednostavno i svodi se na klikanje ikone za pohranu promjena u uređivaču programskog koda i zatim klikanje gumba za ponovno učitavanje u internet pregledniku. Također prednost razvojnog poslužitelja je da se ne moramo brinuti oko sigurnosnih pitanja prilikom razvoja i testiranja, kao ni oko pitanja što korisnici vide ili rade sa aplikacijom, što ne možemo reći u u slučaju razvoja na javnom poslužitelju.

Postoje razni paketi aplikacija koji se zasnivaju na Apache poslužitelju koji objedinjuju aplikacije potrebne za razvoj dinamičnih web stranica. Među takvima je i Više platformski, Apache HTTP[1] poslužitelj, MySQL, PHP i Perl (XAMPP[2]) paket aplikacija koji je korišten kao razvojni poslužitelj za potrebe ovog rada.

2.1. XAMPP

XAMPP je besplatno web poslužitelj rješenje otvorenog koda, koje se uglavnom sastoji od Apache HTTP poslužitelja, MySQL baze podataka i interpretera za skripte pisane u PHP i Perl programskim jezicima.
Alternative XAMPP-u ovisno o platformi su WAMPP, MAMPP i LAMPP, gdje skraćenice znače "Windows, Apache, MySQL, PHP i Perl", "Mac, Apache, MySQL, PHP i Perl" i "Linux, Apache, MySQL, PHP i Perl". XAMPP je više-platformski, što znači da postoji distribucija za svaku od navedenih platformi.
XAMPP dolazi u obliku potpuno funkcionalne postavke paketa gore navedenih programa, tako da nije potrebno instalirati i postavljati svakog od njih. To znači da je potrebno instalirati samo jedan program kako bismo mogli razvijati web aplikacije na vlastitom razvojnom poslužitelju i to uz minimum uloženog vremena.
Tijekom instalacije paketa kreira se nekoliko defaultnih postavki. Sigurnosne postavke takve instalacije nisu striktne kao što su na javnom komercijalnom web poslužitelju, iz razloga što je

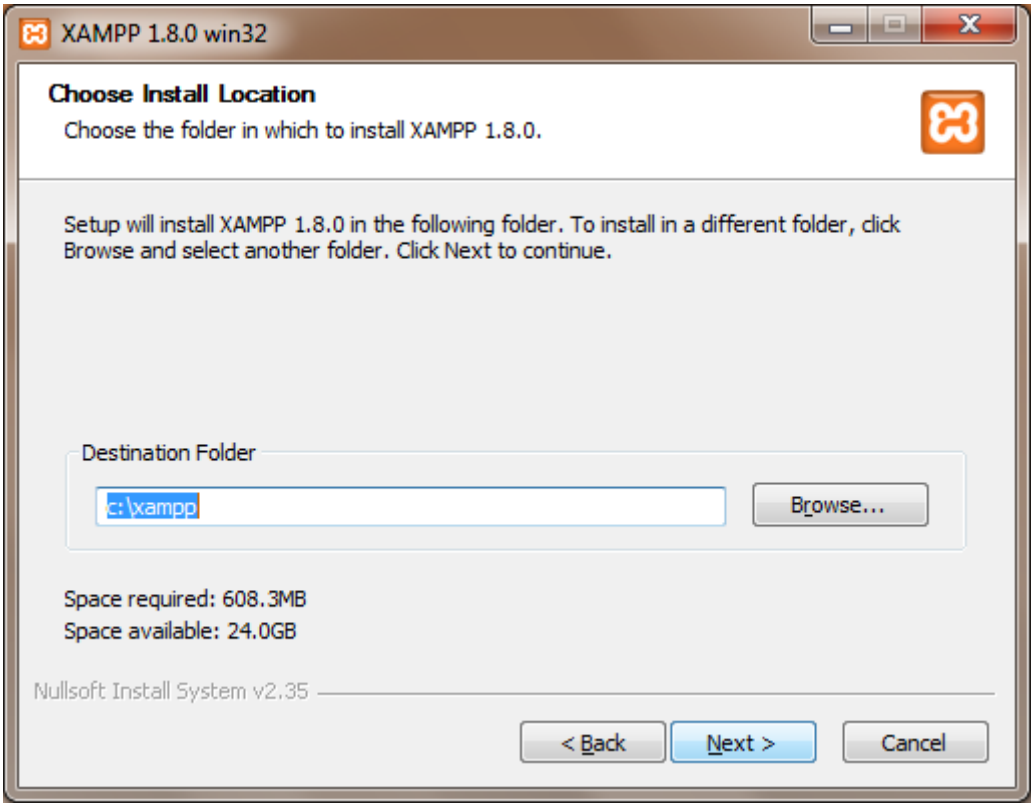
[1] HTTP (engl. HyperText Transfer Protocol) je glavna i najčešća metoda prijenosa informacija na Webu. Osnovna namjena ovog protokola je omogućavanje objavljivanja i prezentacije HTML dokumenata, tj. web stranica.
[2] XAMPP dokumentacija je dostupna na adresi: http://www.apachefriends.org/en/faq-xampp.html

paket odnosno njegova konfiguracija optimizirana za lokalnu uprabu. Iz tog razloga ne preporuča se instalacija takve konfiguracije u svrhu komercijanog web poslužitelja.

U ovom radu koristit će se XAMPP distribucija za Windows platformu, čija je instalacija prikazana u nastavku.

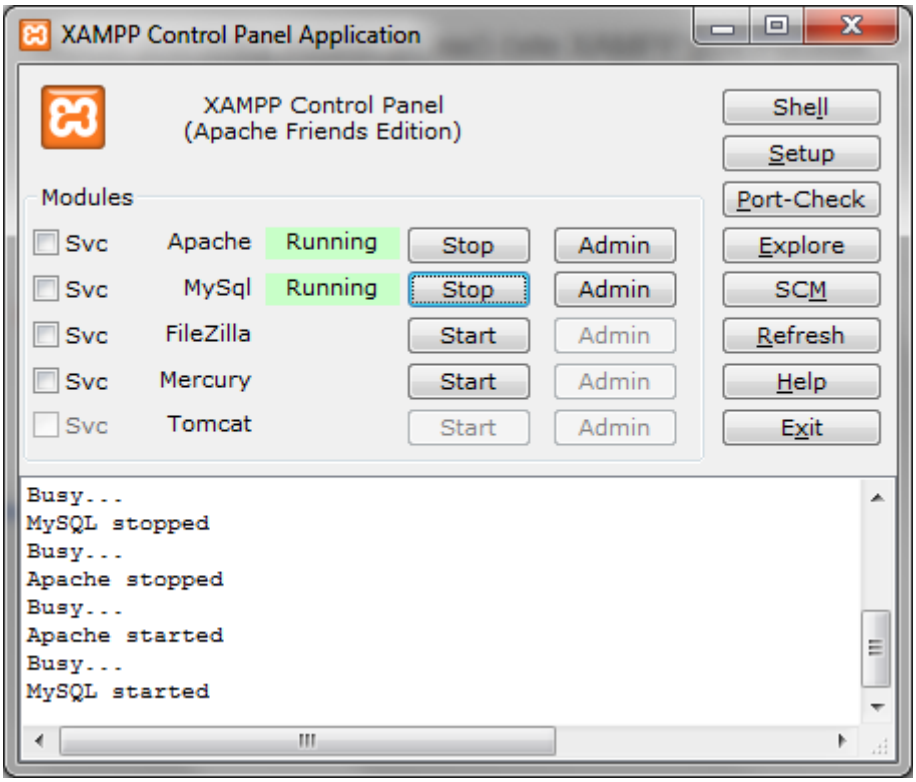
2.1.1. Instalacija na Windows

Instalacija XAMPP pakaeta je krajnje jednostavna. Kako bismo instalirali XAMPP potrebno je posjetiti adresu http://www.apachefriends.org/en/xampp.html, zatim u ovom slučaju odabrati Windows distribuciju i preuzeti paket. Paket se nudi u opcijama Installer, ZIP i 7zip ovisno o načinu na koji želimo pokrenuti XAMPP. Najlakši način za instalaciju XAMPP-a je korištenje Installer verzije prkazane na lici 2.1.



Slika 2.1. Installer verzija XAMPP paketa

Nakon dovršetka instalacije, XAMPP se nalazi pod Start | Programs | XAMPP. Za pokretanje, odnosno zaustavljanje svih poslužitelja kao i instalaciju, odnosno deinstalaciju usluga može se koristiti XAMPP Control Panel aplikacija prikazana na slici 2.2.



Slika 2.2. Control Panel aplikacija

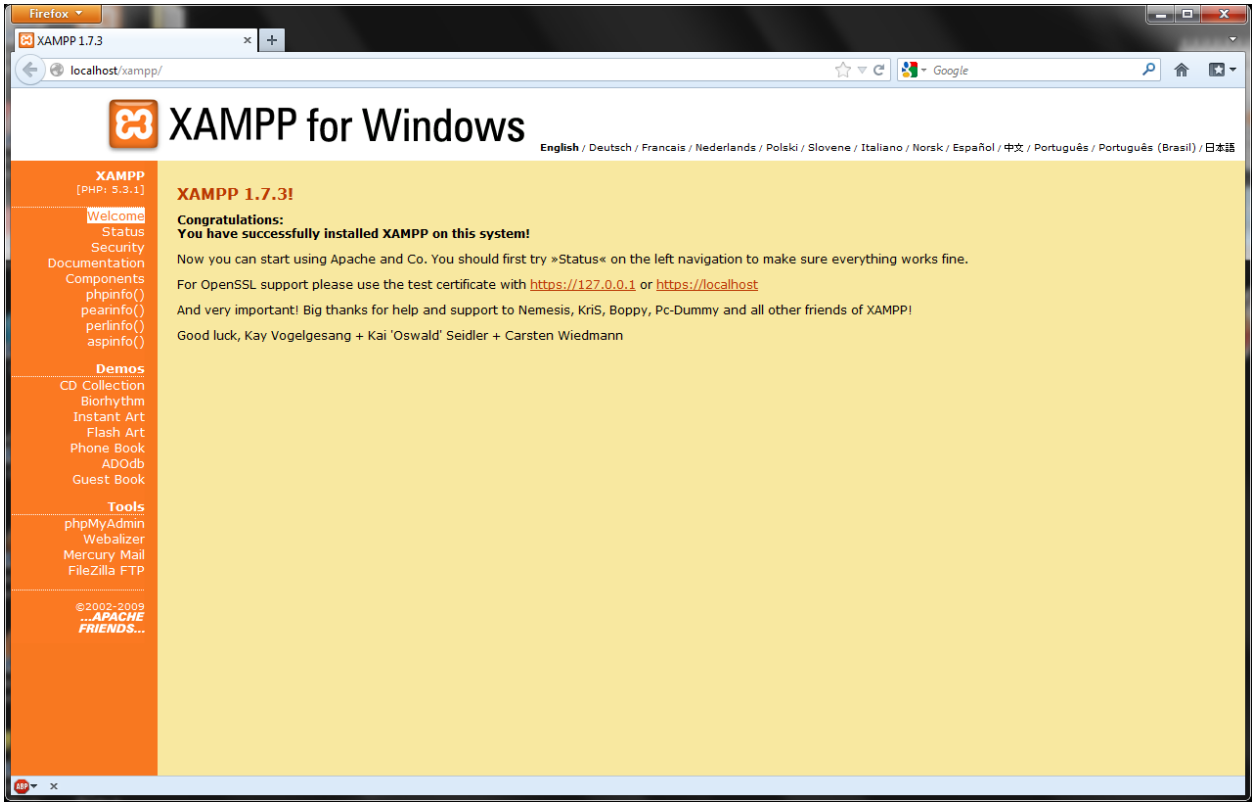
2.1.2. Testiranje instalacije

U ovoj fazi potrebno je provjeriti uspješnost instalacije. Kako bi smo to postigli u internet preglednik potrebno je unesti neku od sljedećih adresa:

- http://127.0.0.1/
- http://localhost/

Prva adresa je IP adresa koju sva računala koriste kako bi referencirala sama sebe, a druga je alias za potpuno isto.

Ukoliko je sve proteklo u redu nakon unosa adrese u internet preglednik trebali bismo vidjeti XAMPP pozdravnu stranicu prikazanu na slici 2.3.



Slika 2.3. XAMPP pozdravna stranica

U XAMPP srukturi direktorija nalazi se direktorij pod imenom „htdocs“ u koji se smještaju datoteke web aplikacija. U tom direktoriju web poslužitelj (Apache) traži datoteke koje će poslužiti. Uobičajeno je za svaku aplikaciju koju stvaramo stvoriti direktorij unutar „htdocs“ direktorija kako bi se izbjegli mogući konflikti. Za potrebe ovog rada unutar „htdocs“ direktorija stvoren je direktorij pod imenom „klopa“ u koji će se pohranjivati datoteke web aplikacije koja je tema ovog rada. Nakon toga aplikciji pristupamo unosom sljedeće adrese u preglednik:

http://localhost/klopa/

Web poslužitelji su uobičajeno konfigurirani na način da traže index datoteku unutar direktorija navedenog u zahtjevu. To može biti index.htm, index.html, index.php itd., čiji sadržaj se zatim prikazuje kao početna stranica aplikacije. Na taj način u prethodnoj URL[3] adresi ustvari zatražena je početna stranica aplikacije „Klopa“.

Web poslužitelji su visoko kofigurabilni, pa je tako i pravilo smještanja datoteka u „htdocs“ direktorij moguće promijeniti, no to nije tema ovog rada.

[3] URL (akronim za engl. Uniform Resource Locator) je Web adresa određenog resursa na Internetu. Resurs na koji pokazuje URL adresa može biti HTML dokument (web stranica), slika, ili bilo koja datoteka koja nalazi na određenom web serveru.

2.2. Apache web poslužitelj

Web poslužitelj je poslužitelj koji je odgovoran za prihvaćanje HTTP zahtjeva od web klijenata i posluživanje HTTP odgovora istima, obično u obliku web stranica koje sadrže statički sadržaj kao što su tekst, slike i slično, te dinamički sadržaj, odnosno skripte. Apache Web poslužitelj najpopularniji je i najčešće korišteni web poslužitelj tijekom zadnjeg desetljeća. Apache je više platformski, lagan, robustan, a koristi se u malim tvrtkama, kao i velikim korporacijama. Apache je također besplatan i open-source. Apache web poslužitelj ima gotovo neograničene mogućnosti zbog svoje velike modularnosti, koja omogućuje da se integrira s brojnim drugim aplikacijama. Jedan od najpopularnijih paketa je LAMP poslužitelj paket web aplikacijia, koji uključuje Apache poslužitelj, MySQL, PHP, Perl i Python.

„Apache web poslužitelj je razvijen od strane Apache Software Foundation. Projekt Apache nastaje 1995. godine, inačica 1.0 izlazi 1. prosinca 1995. U razdoblju samo jedne godine njegova popularnost nadilazi onu NCSA (National Center for Supercomputing Application) odnosno poslužitelja iz kojeg proizlazi i sam Apache. Inačica 2.0 Apache izlazi tijekom konferencije ApacheCon, održane u ožujku 2000. u Orlandu, Florida. Velika popularnost ovog softwera je dokaz njegove kvalitete iako spada u open-source, prema istraživanjima Netcrafta 2005., od 75 milijuna web stranica, oko 52 mulijuna koriste Apache web-poslužitelj, u listopadu 2006. godine brojke rastu na 60 milijuna odnosno (60,32%) ukupno postojećih web stranica.“ **[Wikipedija "Apache (webposlužitelj)“ < http://hr.wikipedia.org/wiki/Apache_%28webposlužitelj%29 > (03. kolovoz 2012.)]**

Biti u stanju konfigurirati i osigurati Apache web poslužitelj je jedna od najvažnijih zadaća za (Linux) administratore sustava. Web poslužitelji su središta informacija. Loša konfiguracija ili ugroženost nekom prijetnjom poslužitelja može izložiti velik broj ljudi neželjenom sadržaju i potencijalno izazvati goleme štete uključenim stranama.

2.3. PHP

„PHP[4] je nastao iz PHP/FI kojeg je 1995. godine napravio Rasmus Lerdorf, kombinirajući Perl skripte na svojim osobnim web vanjskicama. Taj software je nazvao 'Personal Home Page Tools / Forms Interpreter'. S vremenom je na to dodavao neke funkcije iz programskog jezika C za komunikaciju s bazama podataka i publiciranje dinamičkih web stranica. Rasmus je javno

[4] PHP: Hypertext preprocessor

objavio kod svog PHP/FI da bi ga svi mogli koristiti, ali i ako žele sudjelovati u budućem razvoju i poboljšanju.“ **[Galić; 2010; str. 14]**

PHP je jezik koji se koristiti kako bi poslužitelj dinamički generirao sadržaj web stranice, odnosno HTML[5] kod. Dinamički generirani HTML kod je kod koji je potencijalno svaki put drugačiji kada preglednik, odnosno web klijent zahtjeva web stranicu. Ovo znači da je sadržaj rezultat obrade PHP koda. Kada se sadržaj generira na ovakav način, klijent kojem je sadržaj isporučen ne može vidjeti programski kod koji je generirao taj sadržaj, ako bi primjerice pogledao izvorni kod stranice. Klijent u tom slučaju ima uvid u čisti generirani HTML kod stranice.

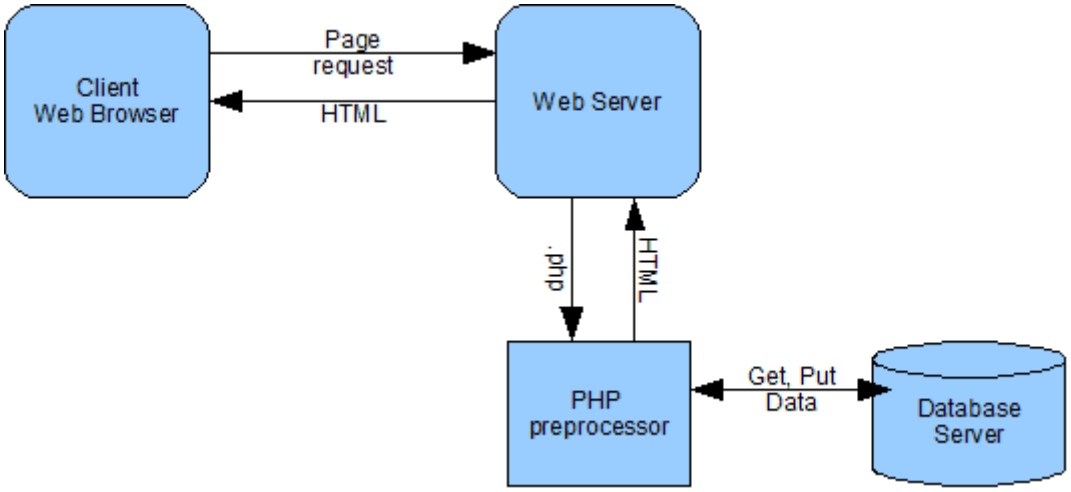
„Kada se govori o programskom jeziku PHP on je u pravilu dio troslojne arhitekture budući da se u većini slučajeva dio podataka pohranjuje u bazu podataka. Troslojnu arhitekturu čine, kao što joj i ime govori, tri osnovna sloja. Prvi sloj je klijentski koji uključuje preglednik i samu mrežu Internet. Srednji sloj je poslužitelj weba na kojem se izvršavaju skriptni jezici ili izvršne datoteke, dok je posljednji, treći sloj, onaj u kojem se nalazi sustav za upravljanje bazom podataka i sama baza podataka.“ **[Paunović, Tomić; 2006; str. 8-9]**

PHP kod dolazi u obliku skripti, koje su obične tekstualne datoteke, a mogu se pisati i u najobičnijem uređivaču teksta. PHP skripta započinje sa oznakom „<?php“, a završava sa oznakom „?>“. Sve unutar tih oznaka smatra se PHP programom i na taj način skripta se odvaja od ostalog sadržaja u toj datoteci. Datoteka koja sadrži PHP kod, odnosno skriptu, može imati ekstenziju: „*.php“, „*.php3“, „*.phtml“ i dr., a može sadržavati tekst, HTML tagove i skripte. Standardna ekstentija za PHP datoteku je „*.php“ , iako je moguće istu datoteku snimiti i sa „*.htm“ ili „*.html“ ekstenzijom, pri čemu se tada web poslužitelj mora podesiti da i takve datoteke prije slanja u preglednik prosljedi PHP-u. Ovakvu metodu nerjetko koriste web programeri kako bi sakrili činjenicu da koriste PHP jezik za razvoj web aplikacija.

PHP je programski jezik interpreterskog tipa. PHP interpreter je softver na web poslužitelju koji čita te datoteke i čini ih smislenima, dajući web poslužitelju HTML kod i upute o tome što treba raditi sljedeće. U praksi se ovo odvija na sljedeći način, korisnik putem preglednika zatraži web stranicu na nekom poslužitelju, koji na osnovu ekstenzije prepoznaje da se radi o PHP datoteci i poziva instalirani PHP da interpretira i izvede operacije navedene u PHP skripti. Zatim se

[5] Hypertext MarkupLanguage

rezultat vraća web poslužitelju, koji sve šalje pregledniku, čija je uloga da tu datoteku grafički oblikuje i prikaže korisniku. Ovaj proces shematski je prikazan na slici 2.5. u troslojnoj arhitekturi koja uključuje i sustav za upravljanje bazom podataka.



Slika 2.5. Proces zahtjeva za web vanjskicom koji uključuje PHP

Tekstualna datoteka, odnosno skripta, interpretira se liniju po liniju koda, svaki put kada se datoteci pristupi. Ovaj koncept je različit od jezika kao što su Java ili C + +, koji se prevode (engl. Compile) u strojni jezik računala. Ti jezici pišu se također u tekstualnim datotekama, ali nakon toga pokreće se naredba koja pretvara te tekstualne datoteke u nešto drugo: class datoteke, binarne datoteke, komade nečitljiva koda koji koristi računalo.

„Jedan od razloga vrlo dobre prihvaćenosti programskog jezika PHP je i vrlo slična sintaksa programskom jeziku C. Velik broj osnovnih funkcija ima istu sintaksu. Početnicima je također vrlo dobar jer nije ograničavajući kao neki drugi programski jezici budući da nema deklaracija tipova varijabli, a moguće je i istu varijablu koristiti za pohranu različitih vrsti vrijednosti. Iako ove karakteristike programski kod čini manje osjetljivim na pogreške jednostavnije ih je i napraviti budući da prevodilac (eng. Compiler) na njih neće upozoriti.“ **[Paunović, Tomić; 2006; str. 11]**

2.4. MySQL

MySQL je sustav za upravljanje relacijskim bazama podataka (RDBMS[6]), i kao takav vjerojatno je najpopularnije rješenje otvorenog koda koje se koristi kao poslužitelj baze podataka. MySQL je najčešće korištena baza podataka za aplikacije pisane u PHP programskom jeziku. MySQL je

[6] RBBMS (engl. *relational database management system*; hr. relacijski susstav za upravljanje bazom podataka)

nastao kao vlasništvo švedske tvrtke MySQL AB koja je danas u vlasništvu tvrtke Oracle Corporation i nazvan je prema kćeri osnivača My. „Zanimljivo je da je MySQL prvobitno nastao kao interni projekt. Prva verzija ovoga software-a izdana je 23.05.1995., najnovija verzija 5.0.51a (Community poslužitelj) izdana je 06.12.2007. Treba reći da je source kod ove aplikacije napisan u C i C++ programskom jeziku. Iz tih razloga ova aplikacija radi na mnogo različitih OS platforma.“ **[Galić; 2010; str. 23]**

Svoju popularnost stekao je iz nekoliko razloga kao što su: MySQL je brz, stabilan, lako se uči, moguće ga je pokretati na popularnim sistemima (Windows, Linux, Mac OS X, Unix-type), podržava razne programske jezike (kao što su C, C++, C#, Java, Perl, PHP, Python, VB, i VB.NET), opširno je dokumentiran i dostupan je pod GPL[7] licencom.

MySQL pripada Klijent/Poslužitelj arhitekturi SUBP-a. Takav sustav podjeljen je na dvije komponente (Poslužiteljska i Klijentska). Poslužiteljska komponenta zadužena je za interakciju sa bazom podataka, a klijentska komponenta (aplikacijski porgram) komunicira sa poslužiteljom. Klijenti šalju zahtjeve prema poslužitelju koji za uzvrat obrađuje zahtjev i vraća razultat zahtjeva natrag prema klijentu. Ovakav sustav ima nekoliko prednosti. Naime klijent i poslužitelj ne moraju se izvršavati na istom fizičkom računalu. Na taj način odvajanjem klijenta od poslužitelja proširuje se raspon vrste klijenata koji mogu pristupiti bazi podataka. Klijenti mogu biti MySQL alati i aplikacije pisane u raznim programskim jezicima.

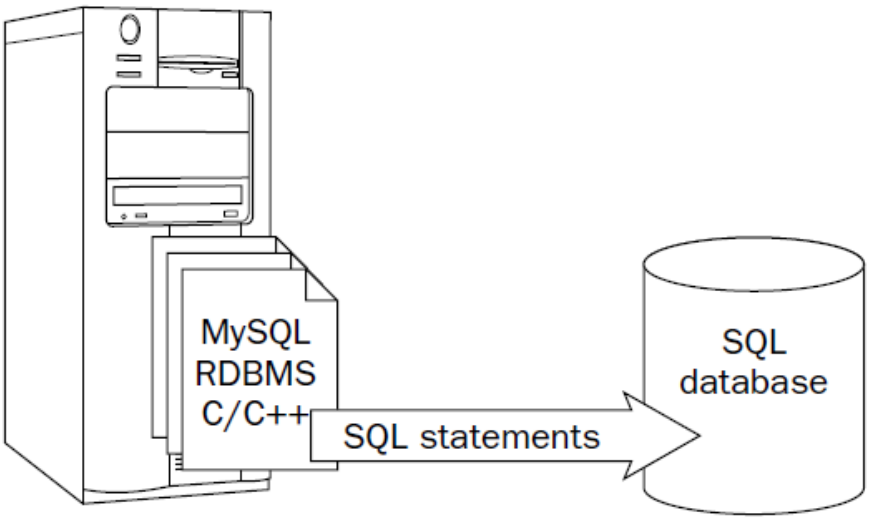
Gotovo svi poznati sustavi za baze podataka (Oracle, Microsoft SQL Poslužitelj, itd.) su Klijent/Poslužitelj sustavi. Takvi sustavi različiti su od Datotečnih poslužitelja (File/Server), kakvi su primjerice Microsoft Access, dBase, and FoxPro, čija mana je neučinkovitost pri korištenju u mreži, kako broj korisnika raste.

„MySQL podržava kao što mu i ime sugerira SQL (Structured Query Language). SQL je prije svega jezik korišten za interakciju sa bazom podataka, te služi za definiranje strukture baze podataka, pohranu podataka, manipulaciju podacima, kontrolu pristupa podacima i osiguranje integriteta podataka.“ **[Sheldon, Moes; 2005; str. 10]**

Na slici 2.4. prikazana je interakcija SQL jezika sa RDBMS. Slika prikazuje MySQL kao poslužitelj na nekoj platformi, zatim bazu podataka, pohranjenu interno ili eksterno, koja sadrži datoteke baze podataka. Iako RDMS olakšava kreaciju i održavanje baze podataka pomoču C/C++ aplikacija, SQL ustvari kreira i održava bazu podataka.

---

[7] **GNU General Public License** (kratice **GNU GPL** i samo **GPL**) je vjerojatno najpoznatija i najšire korištena licenca za slobodan softver, koju je originalno kreirao Richard Stallman za projekt GNU, a o kojoj se danas brine Free software foundation (FSF). GPL je napisan tako da sačuva slobode korisnika softvera: pravo na korištenje u bilo koju svrhu, pravo na izradu kopija i pravo na proučavanje, mijenjanje i redistribuciju modificiranog programa.



Slika 2.4. Interakcija SQL jezika sa RDBMS. [Sheldon, Moes; 2005; str. 10]

2.4.1. Pristup MySQL-u

MySQL sustavu za upravljanje bazom podataka moguće je pristupiti na nekoliko načina, među kojima su: pristup putem naredbene linije, pristup putem sučelja (GUI[8]), te pristup korištenjm programskog jezika poput PHP-a. Prije nego što se započne sa pristupom MySQL poslužitelju, isti je potrebno pokrenuti na razvojnom poslužitelju.

Za pristup MySQL-u putem naredbene linije, Windows korisnici trebaju pokrenuti naredbeni redak (*engl. command prompt*), te unesti sljedeću naredbenu liniju:

C:\>mysql – u [korisničko_ime] – p [lozinka]

Budući da je MySQL instalacija u ovom slučaju prilagođena za rad sa razvojnim poslužiteljom, inicijalni MySQL korisnik je „*root*“ i nema postavljenu lozinku. Prema tome naredba koju treba unijeti u naredbeni redak je sljedeća:

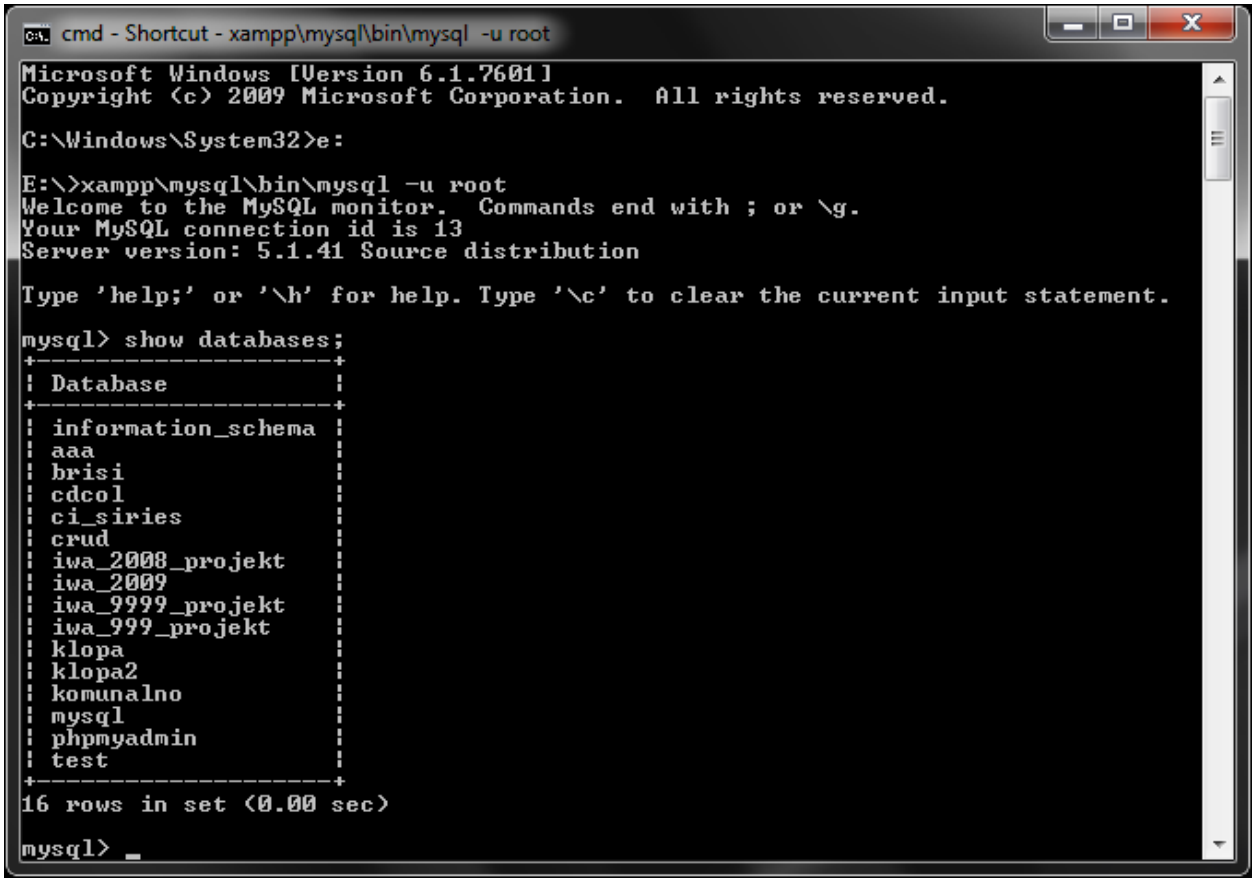
C:\>mysql – u root

Ukoliko putanja prema mysql bin direktoriju nije dodana u windows varijablu okruženja (*engl. environment variable*), potrebno je unijeti potpunu putanju prema istome, odnosno sljedeću naredbu:

[8] GUI (engl. *graphical user interface*; hr. sučelje) je metoda interakcije sa računalom kroz manipulaciju grafičkim elementima i dodacima uz pomoć tekstovnih poruka i obavještenja.

C:\>xampp\mysql\bin\mysql -u root

Nakon unosa ove naredbe, pojavljuje se sljedeći prozor prikazan na slici 2.5., u kojem je unešena SQL naredba „SHOW DATABASES;“ kojom se ispišu sve baze podataka dostupne na poslužitelju.

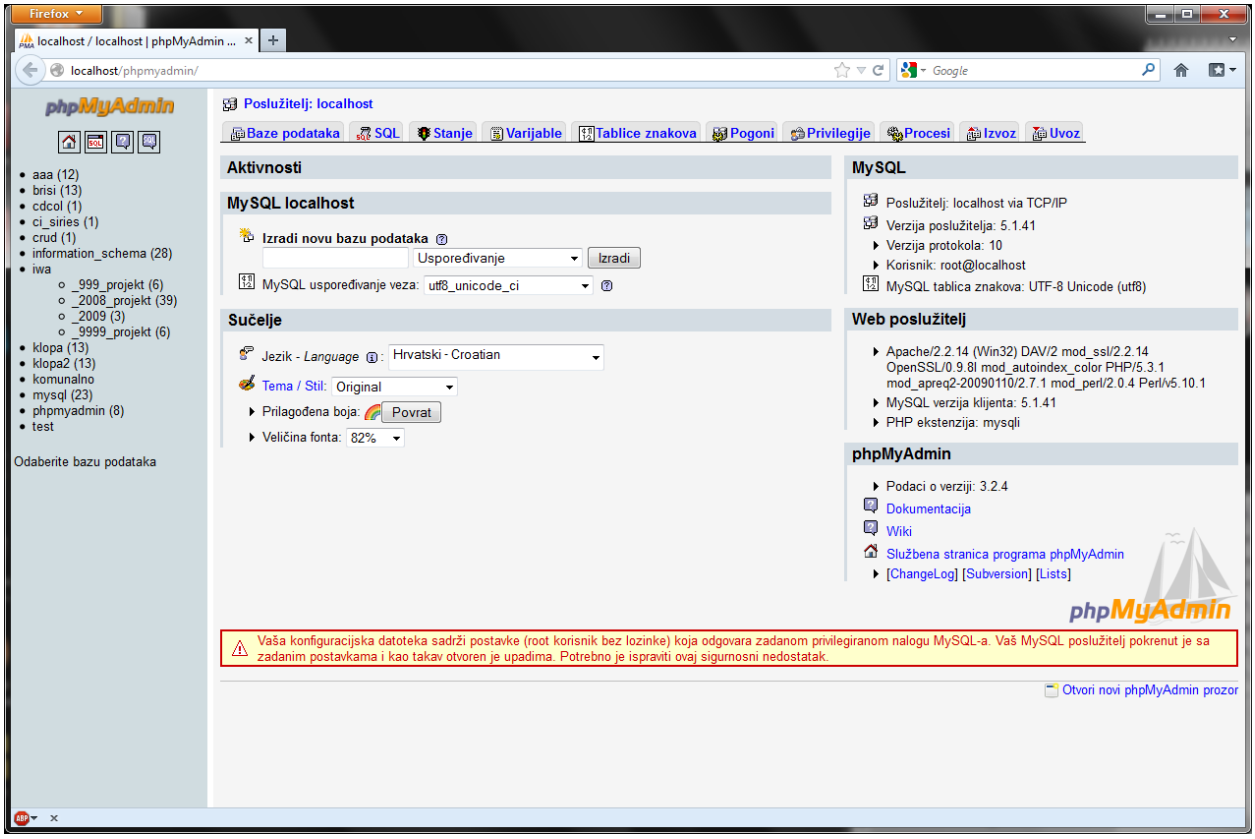


Slika 2.5. Pristup MySQL-u putem naredbene linije

Za pristup MySQL-u putem sučelja potrebno je pokrenuti Apache web poslužitelj na razvojnom poslužitelju i zatim pokrenuti phpMyAdmin sučelje. Sučelje je moguće pokrenuti korištenjem ranije spomenute Xampp Control Panel aplikacije ili unosom sljedeće adrese u web preglednik:

http://localhost/phpmyadmin/

Izgled phpMyAdmin sučelja prikazan je na slici 2.6.



Slika 2.6. izgled phpMyAdmin sučelja

Za pristup MySQL-u putem PHP-a potrebno je stvoriti konekciju na bazu podataka. Konekcija koju stvaramo ima dvije komponente, spajanje na poslužitelj baze podataka i selekcija baze podataka. Za spajanje na poslužitelj može se koristiti PHP funkcija mysql_connect( ), koja prihvača tri parametra: ime računala, odnosno adresu na kojoj je smješten MySQl poslužitelj, naziv korisničkog računa koji je ovlašten za spajanje na MySQL poslužitelj, te lozinku za taj račun. Sintaksa mysql_connect( ) funkcije je sljedeća:

```
<?php
$BP_poslužitelj = 'host_name';
$BP_korisnik =  'user_name';
$BP_lozinka = 'password';
$dbc = mysql_connect($BP_poslužitelj, $BP_korisnik, $BP_lozinka)
      or die('Could not connect: ' . mysql_error());
?>
```

Nakon spajanja na poslužitelj slijedi druga komponenta konekcije, a to je selekcija baze podataka. Za selekciju baze podataka koristimo PHP funkciju mysql_select_db( ), u kojoj navodimo ime baze podataka na koju se spajamo kao prvi parametar i opcionalno možemo navesti identifikator veze kao drugi parametar.

Sintaksa mysql_select_db( ) funkcije je sljedeća:

```
<?php
…
$BP_bazaPodataka = 'db_name';

$db = mysql_select_db($BP_bazaPodataka, $dbc);
?>
```

Nakon uspješne konekcije i selekcije baze podataka, podacima koji su sadržani u istoj možemo pristupati pomoču PHP aplikacije. Za pristup podacima koristi se PHP mysql_query( ) funkcija. Sintaksa mysql_query( ) funkcije je sljedeća:

```
<?php
…
$sql = "SELECT * FROM Table";

$rs = mysql_query($sql);
?>
```

Kada koristimo ovu funkciju obično to radimo na način da prvo definiramo varijablu (`$sql`) čija je vrijednost SQL naredba (`"SELECT * FROM Table"`), a zatim definiramo varijablu (`$rs`) koja će sadržavati vrijednost rezultata koji je vratila mysql_query( ) funkcija. Kao prvi parametar toj funkciji obavezno prosljeđujemo SQL upit. Nakon što mysql_query( ) funkcija obradi SQL naredbu i dodjeli rezultat obrade definiranoj varijabli (`$rs`), tu varijablu zatim koristimo u programskom kodu kako bismo prikazali rezultate u aplikaciji.

2.4.2. Referencijalni integritet

Vanjski ključ u bazi podataka je stupac u tablici koji upućuje na primarni ključ druge tablice u cilju održavanja odnosa, odnosno veze između te dvije tablice. Primjerice u aplikaciji koja je tema ovog rada postoji tablica Restoran koja ima stupac nazvan FirmaID. Taj stupac upućuje (referencira) na tablicu Firma i on je vanjski ključ u tablici Restoran. Korištenjem vanjskih ključeva sustavu za upravljanje bazom podataka se daje do znanja da stupac koji je vanjski ključ neke tablice ima vezu na stupac koji je primarni ključ referencirane tablice. Ova sposobnost potencijalno je vrlo dobra, a naziva se „foreign-key constraint“. Kada je sustav za upravljanje bazom podataka svjestan ove veze, on može provjeriti da vrijednost koja se pohranjuje u polje vanjskog ključa postoji u stupcu primarnog ključa referencirane tablice. U slučaju da ta vrijednost ne postoji sustav će odbiti unos vrijednosti. Sposobnost poslužitelja baze podataka da

odbije zapis podataka zbog toga što oni ne zadovoljavaju uvjetima povezanih tablica naziva se referencijalni integritet. MySQL trenutno nudi samo jedan tip tablica koje podržavaju potpunu funkcionalnost vanjskih ključeva, a to je InnoDB tip tablice.

Zanimljivo je napomenuti da se isti efekti korištenja referencijalnog integriteta mogu postići i kroz interakciju aplikacije sa bazom podataka. U tom slučaju prije umetanja ili ažuriranja podataka u tablici, prvo je potrebno dohvatiti vrijednosti iz stupca primarnog ključa, zatim provjeriti dali vrijednost koju ćemo umetnuti u polje vanjskog ključa odgovara vrijednostima iz prvog koraka, te ako vrijednost odgovara umetnuti ju. U svakom slučaju imati mogućnost upotrebe referencijalnog integriteta zasigurno je dobra stvar.

2.4.3. Transakcije

„Prilikom izrade aplikacije koja pristupa podacima u MySQL bazi podataka, mora se uzeti u obzir hoće li više korisnika pokušati pregledavati i mijenjati podatke u isto vrijeme. Ako korisnici pokušaju obaviti te operacije istovremeno, rezultat mogu biti nedosljedni i netočni podaci. Kako bi se izbjegle takve vrste problema, mogu se koristiti transakcije i izolirati svaku operaciju kako bi se bi se osigurala točnost i konzistentnost podataka.

U kontekstu SQL-a i relacijskih baza podataka, transakcija je skup jednog ili više SQL izraza koji obavljaju niz povezanih aktivnosti. Izjave su grupirane zajedno i tretirane kao jedinstvena jedinica čiji uspjeh ili neuspjeh ovisi o uspješnom izvršenju svake izjave u transakciji.“ **[Sheldon, Moes; 2005; str. 455]**

Ovo znači da ako bilo koja od naredbi iz grupe u transakciji ne prođe, cijela grupa naredbi ne prolazi, a baza se vraća u stanje u kojem je bila prije nego što se pokušala izvesti prva naredba. Ovakav pristup naziva se „commit / rollback“ pristup. Transakcije se koriste u bazama podataka kako bi se osigurala konzistentnost i točnost podataka, te kako bi se podaci zaštitili u slučaju havarije.

Transakcijski sposobna baza podataka mora podržavati četiri svojstva poznata po akronimu ACID[9], koja su definirana kako slijedi:

- **Atomarnost** - Operacije koje čine svaku transakciju tretiraju se kolektivno kao jedna, ili atomska, jedinica. Ili su sve operacije obavljene ili nijedna nije. Teka kada se sve operacije izvedu uspješno, rezultati te transakcije primjenjuju se u bazi podataka.

[9] ACID (engl. *atomicity*, *consistency*, *isolation*, *durability;* hr. atomarnost, konzistentnost, izolacija, trajnost)

- **Konzistentnost** - Svaka transakcija mora ostaviti bazu podataka u istom konzistentnom stanju u kojem je bila i na početku transakcije. Čak i ako se neke od izjava u transakciji izvedu i transakcija neuspije, baza podataka će završiti u konzistentnom stanju.
- **Izolacija** - Svaka transakcija je izolirana od svih drugih. Učinci Transakcije A nisu vidljivi Transakciji B dok Transakcija A nije završena. Ako je transakcija u tijeku, privremeno stanje podataka neće biti vidljivo drugim transakcijama.
- **Trajnost** - Kada je transakcija dovršena, promjene su trajne. Čak i ako se baza podataka sruši, informacije iz uspješne transakcije će biti dostupne i potpune.

U starijim verzijama MySQL-a transakcije nisu bile podržane, što je predstavljalo veliki problem za programere web aplikacija. Trenutno MySQL nudi nekoliko tipova tablica (InnoDB, BDB) koje podržavaju transakcije.

2.4.4. Tipovi tablica

MySQL podržava nekoliko tipova tablica među kojima su BDB, MEMORY, ISAM, INNODB, MERGE i MyISAM. Zadani (*engl. default*) tip tablice do MySQL verzije 5.5. bio je MyISAM, a od te verzije zadani tip tablica postaje InnoDB.

Sintaksa za definiciju tipa tablice je sljedeća:

```
CREATE TABLE Table_name (
Column_name COLUMN_TYPE COLUMN_ATTRIBUTES,
) ENGINE = InnoDB;
```

„U ovoj definiciji, ENGINE klauzula dodana je nakon definicije posljednjeg stupca i završne zagrade. Primijećujemo da jednostavno navedemo ENGINE ključnu riječ, znak jednakosti, i jedan od tipova tablica. Svaka tablica u MySQL-u podržava određeni skup funkcionalnosti i služi određenoj svrsi. Osim toga, svaka vrsta tablice povezana je sa srodnim pogonom za pohranu (rukovatelj) koji obrađuje podatke u toj tablici. Za Na primjer, MyISAM ENGINE obrađuje podatke u MyISAM tablicama.“ **[Sheldon, Moes; 2005; str. 159]**
Slika 2.7.sadrži opis vrsta tablica u MySQL-u.

Table type	Description
BDB	A transaction-safe table that is managed by the Berkeley DB (BDB) handler. The BDB handler also supports automatic recovery and page-level locking. The BDB handler does not work on all the operating systems on which MySQL can operate. For the most part, InnoDB tables have replaced BDB tables.
MEMORY	A table whose contents are stored in memory. The data stored in the tables is available only as long as the MySQL server is available. If the server crashes or is shut down, the data disappears. Because these types of tables are stored in memory, they are very fast and are good candidates for temporary tables. MEMORY tables can also be referred to as HEAP tables, although MEMORY is now the preferable keyword.
InnoDB	A transaction-safe table that is managed by the InnoDB handler. As a result, data is not stored in a .MYD file, but instead is managed in the InnoDB tablespace. InnoDB tables also support full foreign key functionality in MySQL, unlike other tables. In addition, the InnoDB handler supports automatic recovery and row-level locking. InnoDB tables do not perform as well as MyISAM tables.
ISAM	A deprecated table type that was once the default table type in MySQL. The MyISAM table type has replaced it, although it is still supported for backward compatibility. Eventually, ISAM tables will no longer be supported.
MERGE	A virtual table that is made up of identical MyISAM tables. Data is not stored in the MERGE table, but in the underlying MyISAM tables. Changes made to the MERGE table definition do not affect the underlying MyISAM tables. MERGE tables can also be referred to as MRG_MyISAM tables
MyISAM	The default table type in MySQL. MyISAM tables, which are based on and have replaced ISAM tables, support extensive indexing and are optimized for compression and speed. Unlike other table types, BLOB and TEXT columns can be indexed and null values are allowed in indexed columns. MyISAM tables are not transaction safe, and they do not support full foreign key functionality.

Slika 2.7. Opis vrsta tablica u MySQL-u. [Sheldon, Moes; 2005; str. 159]

MyISAM je zadani tip tablica na većini MySQL instalacija. Nekada je to bio i jedini tip tablica dostupan u MySQL-u. MyISAM tablice iznimno su brze i stabilne, no međutim, ne podržavaju transakcije. One nude samo zaključavanje podataka na razini tablice. MyISAM tablice su optimizirane za brzinu u dohvaćanje podataka sa SELECT izjavama. Zbog optimizacije i nedostatka podrške za transakcijama, MyISAM tablice najbolji su odabir za tablice na kojima će se obavljati SELECT operacije daleko češće nego UPDATE ili DELETE operacije.

Primjerice ako stvaramo košaricu za kupovinu, odnosno popis za narudžbu kao što je to slučaj u ovome radu, tablice u kojima ćemo čuvati podatke o narudžbama biti će češće predmet INSERT, UPDATE ili DELETE operacija nego SELECT operacija. Također u ovom slučaju bitna je i podrška za transakcijama iz razloga što ne želimo imati narudžbe bez stavaka u svojoj bazi podataka. Iz tog razloga za takve tablice bolja je upotreba InnoDB vrste tablica, koje su korištene u ovom radu.

InnoDB tablice pružaju potpunu (ACID) podršku za transakcije i zaključavanje na razini retka. Kao što je već ranije spomenuto InnoDB tablice zamjenile su MyISAM tablice kao zadani tip tablica od MySQL verzije 5.5.

Prednosti korištenja InnoDB tablica su sljedeće:

- Ako se poslužitelj sruši zbog nekog hardverskog ili softverskog problema, bez obzira na to što se događalo u bazi podataka za to vrijeme, nije potrebno učiniti ništa posebno nakon ponovnog pokretanja baze podataka. InnoDB automatski finalizira sve promjene koje su potvrđene prije trenutka pada poslužitelja, i poništava sve promjene koje su u tijeku, ali nisu potvrđene.
- InnoDB međuspremnik (engl. *buffer*) sprema podatke iz tablica i indekse kako se podacima pristupa. Često korišteni podaci se obrađuju izravno iz memorije. To ubrzava obradu toliko, da namjenski poslužitelji baza podataka dodijeljuju i do 80% svoje fizičke memorije InnoDB međuspremniku.
- Moguće je postavljanje vanjskih ključeva i nametanje referencijalnog integriteta. Prilikom ažuriranja ili brisanja podataka, ukoliko su ti podaci povezani sa podacima u drugim tablicama, također automatski dolazi do promjena i u povezanim podacima.
- Ako se podaci oštete na disk ili u memoriju, checksum mehanizam upozorava na lažne podatake prije nego što ih uptrebimo.
- Prilikom dizajniranja baze podataka sa pripadajućim stupcima primarnih ključeva za tablice, operacije koje uključuju te stupce automatski su optimizirane. Na taj način vrlo se brzo referencira stupac primarnog ključa u klauzulama kao što su: WHERE, GROUP BY, ORDER BY, kao i u operacijama spoja.

Sve ove značajke dovele su do činjenice da InnoDB tablice sada imaju i bolje performanse od MyISAM tablica.

3. Realizacija aplikacije

Ideja je bila razviti aplikaciju koja će omogućiti izravnu vezu između korisnika i ugostiteljskog objekta, korištenjem internet tehnologije, u svrhu upućivanja, odnosno zaprimanja narudžbi.

Aplikacija je zamišljena da korisnicima pruži jednistveni pregled ugostiteljskih objekata koji dostavljaju u željenom kvartu, pregled njihove ponude odnosno jelovnika, te besplatno naručivanje u svega nekoliko klikova. Također aplikacija je zamišljena da ugostiteljskim objektima pruži prezentaciju svoje ponude na mediju koji je dostupan svima, tj. internetu. Kako je internet postao važno sredstvo komunikacije s potrošačima zbog svojih prednosti nad klasičnim medijima, ugostiteljskim objektima omogućava se predstavljanje velikom broju potencijalnih konzumenata njihovih usluga.

Osnovna svrha aplikacije je da posreduje pri narudžbama na opisani način, te da pri tome pohranjuje podatke o ostvarenoj naknadi. Naknada se ostvaruje u obliku provizije na realiziranu narudžbu, koja se naplaćuje od ugostiteljskih objekata.

Aplikacija je razvijena s namjerom da bude prilagodljiva u smislu da pruža mogućnost promjene cijene pojedine stavke iz jelovnika, zatim promjene postotka provizije na narudžbe, te promjene stope PDV-a. U tu svrhu baza podataka razvijena je na način da novonastale promjene u postavkama rada aplikacije ne utječu na već prethodno obavljene narudžbe.

3.1. Opis zadatka

Sustav treba omogućiti online naručivanje iz ugostiteljskih objekata. On treba sadržavati ponudu ugostiteljskih objekata iz kojih je moguće naručivati, te ponudu njihovih jelovnika.

Korisnici sustava su:

- Administarator,
- Korisnik – registrirani i prijavljeni korisnik
- Gost – neregistrirani, anonimni posjetitelj .

Anonimni korisnik odnosno gost može pregledavati ponudu restorana kao i njihovih jelovnika. Prilikom pregleda ponude istu može filtrirati prema željenoj lokaciji dostave.

Prijavljeni korisnik može naručivati iz ugostiteljskih objekata, te može vidjeti podatke o vlastitim narudžbama. Kod naručivanja potrebno je omogućiti dodavanje stavki na popis za narudžbu, zatim je potrebno omogućiti promjenu količine svake pojedine stavke na popisu za narudžbu, te uklanjanje svake pojedine stavke sa popisa za narudžbu. Prilikom naručivanja također je potrebno onemogućiti višestruko dodavanje iste stavke na popis za narudžbu, već se isto treba realizirati kroz promjenu količine te stavke. Sličnu restrikciju potrebno je implementirati prilikom dodavanja stavki na popis za narudžbu na način da na jednom popisu mogu biti stavke sa jelovnika iz samo jednog ugostiteljskog objekta.

Administrator sustava dodaje firme i ugostiteljske objekte u sustav i zatim kreira ponudu, odnosno jelovnik svakog ugostiteljskog objekta, te unosi cijenu svake pojedine stavke jelovnika. Administrator sustava može ažurirati podatke o restoranima i mijenjati cijenu i status svake stavke na jelovniku. Administrator može vidjeti sve preglede kao anonimni i prijavljeni korisnik. Uz navedene preglede administrator može vidjeti podatke o svim narudžbama, o njihovom ukupnom broju, ukupnoj sumi, prosječnoj vrijednosti narudžbe, iznosu ostvarene provizije, te iznosu pdv-a iz sume narudžbi. Prilikom pregleda administrator može filtrirati rezultate prema ugostiteljskim objektima i datumima narudžbi.
Među najvažnijim ulogama administratora je konfiguracija sustava. Naime administrator određuje postavke aplikacije kao što su provizija na narudžbe i stopa pdv-a.

3.2. Dizajn baze podataka

Za potrebe promjenjive aplikacije za posredovanje pri narudžama neophodna je baza podataka u koju će se pohranjivati podaci.

Izgradnja baze podataka složen je postupak. Naime potrebno je prikupiti podatke, utvrditi koji su podaci dovoljno važni (relevantni) da bi ušli u bazu podataka i u koje tablice će ući koji podaci. Podaci i tablice se u fazi projektiranja dogovaraju s korisnicima jer ni projektant baze podataka ni programer ne mogu znati koji su podaci relevantni za neku obradu, kakvi su ti podaci i kako utječu jedni na druge. Nakon prikupljanja podataka, iste je potrebno smjestiti u odgovarajuče tablice. U koliko tablica treba te podatke spremiti? U jednu, dvije, tri ili više? Kako izbjeći redundanciju podataka, kako podatke organizirati na način da se podaci mogu čim lakše obrađivati i što je najvažnije, da se mogu unositi kada god je to potrebno. Ova pitanja rješavamo prilikom modeliranja baze podataka.

Za modeliranje baze podataka koriste se razne tehnike poput ERA (engl. entity-relationship, atribute u prijevodu: entiteti-veze-atributi) dijagrama, koja je korištena u ovome radu.

3.2.1. ERA model baze podataka

„Entity-relationship (ER) pristup modeliranju gleda poslovnu domenu u smislu entiteta koji imaju atribute i sudjeluju u odnosima.“ **[Halpin, Morgan; 2008; str. 306]** Iz toga sljedi da ERA model sadrži entitete koji se pojavljuju u bazi podataka zajedno sa njihovim međusobnim odnosima. „ERA model je grafička prezentacija znanja o objektima, vezama i svojstvima.“ **[Brumec; 2008/2009; slide 58]**

Osnovni koncepti ERA modela su:

- Entitet - je konkretna ili apstraktna posebnost, koncept ili objekt od interesa. Entitet je stvarni ili apstraktni predmet ili događaj o kojemu se u informacijskom sustavu pamte podaci. Postoje jaki entiteti, koji egzistiraju samostalno i slabi entiteti koji ne egzistiraju samostalno ili se ne identificiraju samostalno. Svakom entitetu pridružen je određen broj atributa.
- Atribut - (obilježje, svojstvo) je podatak koji identificira, klasificira, kvantificira, izražava kvalitetu ili stanje entiteta. To je jedinstveno obilježje entiteta
- Veza - je odnos (relacija) između dva entiteta. Veza se imenuje, a naziv veze opisuje ulogu entiteta u vezi. Prema broju tipova entiteta koji sudjeluju u vezi određuje se stupanj veze. Veze su dvosmjerne, odnosno označavaju međusobne odnose entiteta u oba smjera.
- Ograničenja – su donje i gornje granice sudjelovanja entiteta u vezi, odnosno kardinalnosti veza. Kardinalnost veza opisuje koliki broj entiteta jednog tipa može biti u vezi s entitetima drugog tipa.

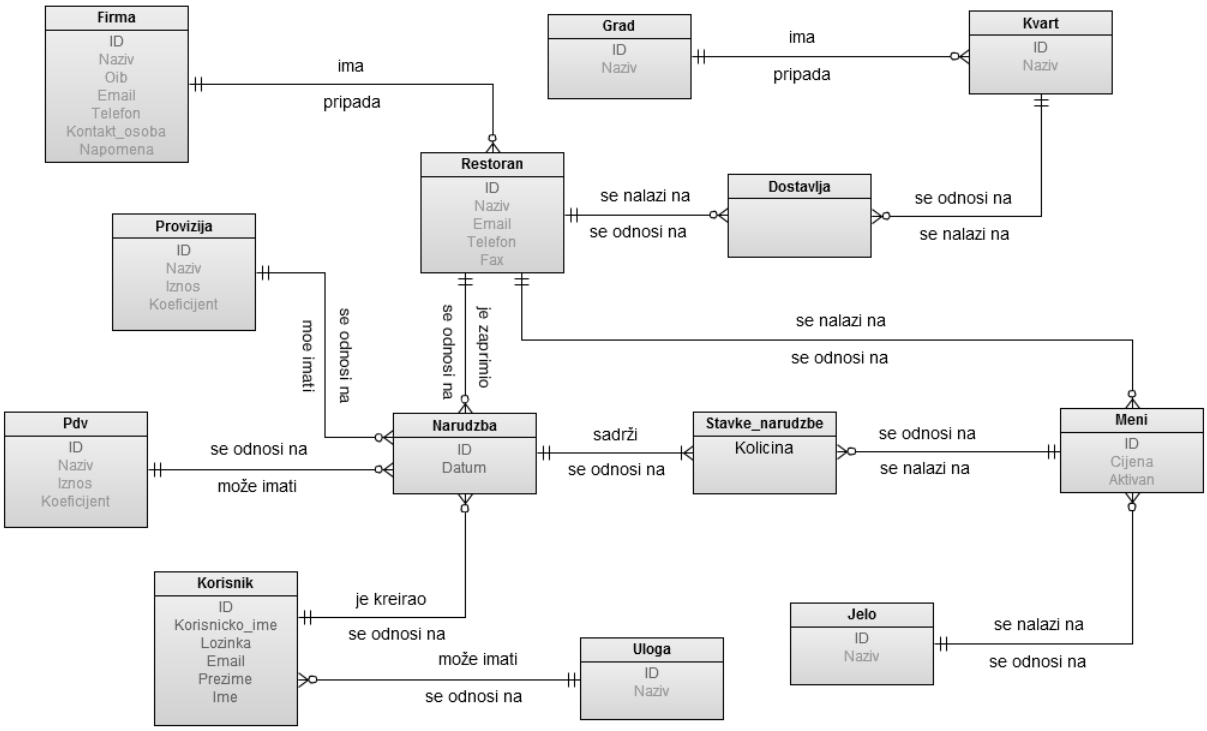
Postoji više različitih notacija za ERA modeliranje, među kojima su: Chenova, Barkerova, IE (engl. Information engineering) notacija, te IDEF1X notacija. U industriji, najpopularnije verzije su Barkerova IE notacija.

U ovom radu korištena je IE notacija još poznata kao Martinova. U IE notaciji entiteti su prikazani imenovanim pravokutnicima, a atributi se mogu prikazati unutar pravokutnika, ispod njegovog imena. Veze između entiteta prikazane su kao imenovane linije koje povezuju entitete. „Opcionalnost i kardinalnost veze između entiteta prikazani su simbolima na krajevima linija. Za označavanje da je uloga opcionalna, krug "O" stavlja se na drugom kraju linije, što znači

minimalna frekvencija sudjelovanja 0. Kako bi se naznačilo da je uloga obvezna, znak " ǀ " stavlja se na drugom kraju linije, što znači minimalna frekvencija sudjelovanja 1. Kardinalitet od "mnogo" označava se simblom „svračje noge“ "<". U kombinaciji s minimalnom frekvencijom od 0 ili 1, znak " ǀ " koristi se za označavanje maksimalne frekvencija od 1. Pri čemu tada, kombinacija " O ǀ " označava "najviše jedan", a kombinacija " ǀ ǀ " ukazuje "točno" 1.“ **[Halpin, Morgan; 2008; str. 319]**

ERA model sustava može se jednostavno transformirati u relacijsku shemu baze podataka postupcima dekompozicije entiteta i veza u konkretne tablice i navođenjem njihovih atributa, kao što će biti prikazano u ovom radu.

Model ove baze podataka sadrži trinaest entiteta sa njihovim atributtima i međusobnim odnosima, koji su prikazani ERA modelom na slici 3.1.



Slika 3.1. ERA model baze podataka

Iz ERA modela baze podataka vidi se da se baza podataka sastoji od jedanaest jakih entiteta i dva slaba entiteta. Slabi entiteti su oni koji ne mogu biti jedinstveno identificirani vlastitim atributima, dakle, mora se koristiti vanjski ključ u kombinaciji sa svojim atributima ili u kombinaciji sa drugim vanjskim ključem da se stvoriti primarni ključ. Vanjski ključ nekog

entiteta je primarni ključ entiteta s kojim je povezan. Na modelu ih prepoznajemo po tome što nemaju identificirajući atribut.

Jaki entiteti su:

1. Firma
2. Restoran
3. Grad
4. Kvart
5. Provizija
6. Pdv
7. Korisnik
8. Uloga
9. Narudzba
10. Jelo
11. Meni

Slabi entiteti su:

1. Dostavlja
2. Stavke_narudzbe

Kompleksnost odnosa između jakih entiteta razlog je uvođenja slabih entiteta. Uloga slabog entiteta je da ostvari vezu između dva jaka entiteta. Primjerice veza između entiteta Kvartovi i Restorani je veza M:N (više na prema više), što bi začilo da jedan restoran može dostavljati u više kvartova i da u jedan kvart može dostavljati više restorana. Takva veza ne može se direktno ostvariti, te iz tog razloga uvodimo pomoćni tj. slabi entitet Dostavlja, koji je sa svakim od jakih entiteta u vezi 1:N (jedan na prema više). Ta se veza kreira tako da se primarni ključevi iz jakih entiteta prenose u slabi entitet kao vanjski ključevi te zajedno čine višekoponentni primarni ključ u slabom entitetu.
Iz modela je zatim vidljiva opcionalnost i kardinalnost svake veze, pa tako vidimo da je veza između primjerice entiteta Provizija i Narudžba opcionalna s jedne strane, a obavezna s druge strane.
Opcionalni dio veze „provizija se odnosi na“ je entitet Narudzba. Opcionalnost u ovom slučaju znači da može postojati stopa provizije koja nije ni na jednoj narudžbi. Obavezni dio veze „narudžba može imati“ je entitet Provizija, što zači da narudžba mora imati neku stopu provizije.

Kardinalnost veze „provizija se odnosi na“ prikazuje da se jedna stopa provizije može odnositi na više narudžbi, dok kardinalnost veze „narudžba može imati“ prikazuje da jedna narudžba može imati samo jednu stopu provizije.

Promatranjem kompleksne veze između entiteta Narudzba i entiteta Meni gdje je uveden slabi (pomoćni) entitet Stavke_narudzbe, vidimo da narudžba mora imati jednu ili više stavaka i da se jedna stavka mora odnositi na jednu i samo jednu (točno jednu) narudžbu. Drugi dio te veze prikazuje kako se jedna stavka narudžbe također mora odnositi na jednu i samo jednu (točno jednu) stavku iz menia, dok može postojati stavka u meniu koja može biti na nula ili više narudžbi, dakle ona ne mora biti ni na jednoj narudžbi.
Na isti način moguće je analizirati i ostale odnose između entiteta na ERA modelu baze podataka.

3.2.2. Relacijski model baze podataka

Relacijski model je linearni prikaz logičkog modela podataka, za razliku od prethodnog ERA modela koji je grafički. Relacija ima dva aspekta: značenje i sadržaj. Značenje se iskazuje relacijskom shemom, a sadržaj skupom vrijednosti tj. slogovima relacije.

Prevođenje ERA modela u Relacijski model je jednoznačno, a provodi se u dva koraka: prevođenje entiteta i prevođenje veza.

Prevođenje entiteta:

- Svaki entitet iz ERA modela baze podataka postaje relacija relacijskog modela baze podataka i to na način da ime entiteta postaje ime relacije.
- Atributi entiteta postaju atributi relacijske sheme.
- Identifikator postaje primarni ključ relacije.

Prevođenje veza:

- Veza 1:1 iskazuje se vanjskim ključem u onoj relacijskoj shemi gdje će poprimiti manje nul-vrijednosti
- Veza 1:M prevodi se na način da odgovarajuća tablica dobiva atribut naziv veze koji je vanjski ključ na drugu tablicu u vezi i taj je atribut definiran kao not null. Vanjski ključ uvijek je u relaciji „na vanjski više“.

- Veza M:N između dva entiteta iskazuje se otvaranjem treće relacijske sheme, čiji je ključ tada dvokomponentni, a sastoji se od primarnih ključeva entiteta koji su u vezi.

U nastavku je prikazan relacijski model baze podataka, koji je napravljen na temelju ranije oblikovanog ERA modela baze podataka.

Primarni ključevi relacije na relacijskom modelu podcrtani su punom linijom, a vanjski ključevi istaknuti su kurzivom

Grad (ID, Naziv)

Kvart (ID, Naziv, *GradID*)

Firma (ID, Naziv, Oib, Email, Telefon, Kontakt_osoba, Napomena)

Restoran (ID, Naziv, Email, Telefon, Fax, *FirmaID*)

Dostavlja (*KvartID, RestoranID*)

Provizija (ID, Naziv, Iznos, Koeficijent)

Pdv (ID, Naziv, Iznos, Koeficijent)

Uloga (ID, Naziv)

Korisnik (ID, Korisnicko_ime, Lozinka, Email, Prezime, Ime, *UlogaID*)

Narudzba (ID, Datum, *RestoranID, KorisnikID, ProvizijaID, PdvID*)

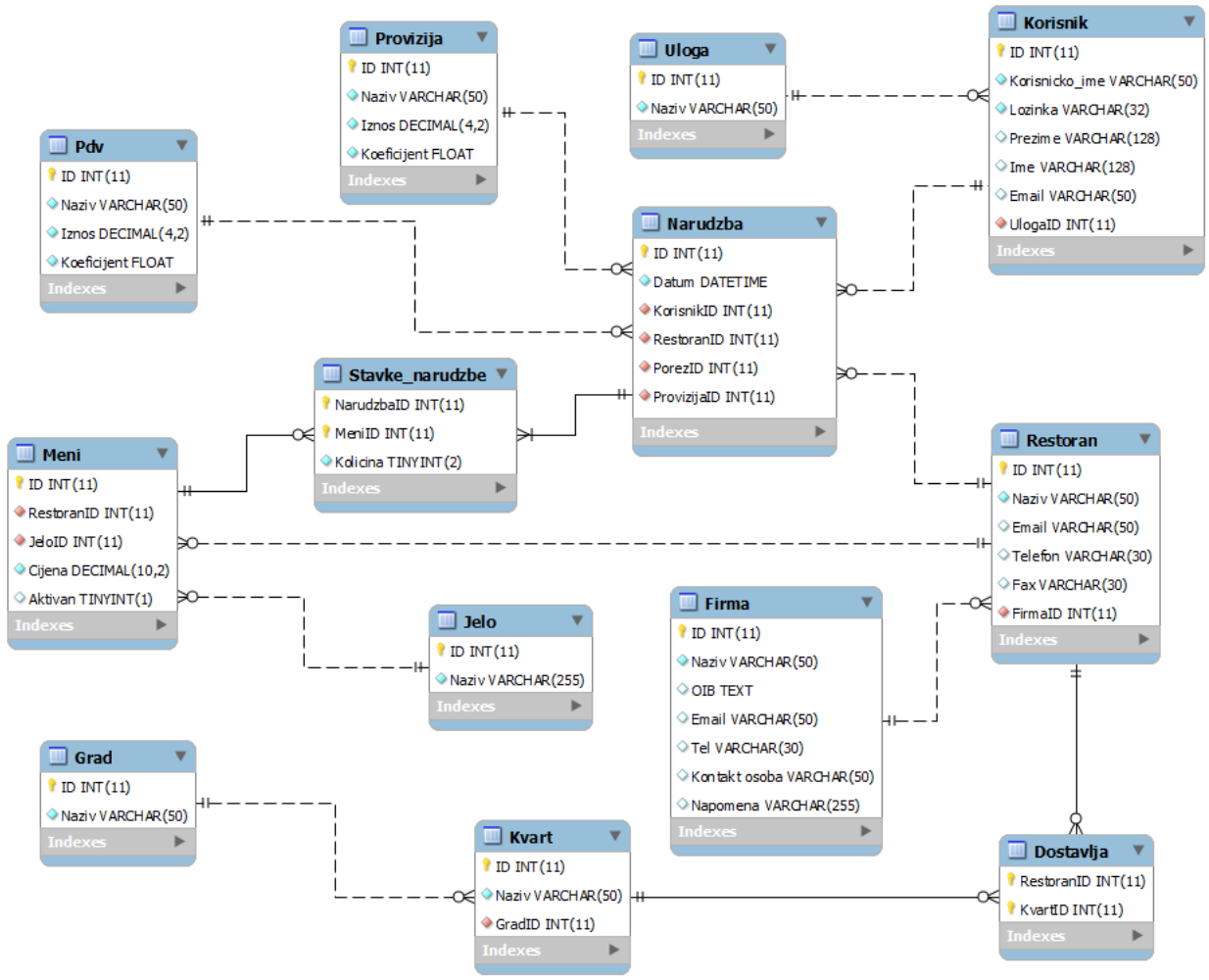
Jelo (ID, Naziv)

Meni (ID, Cijena, Aktivan, *RestoranID, JeloID*)

Stavke_narudzbe (*NarudzbaID, MeniID*)

3.2.3. Dizajn baze podataka u alatu MySQL Workbench

Na temelju znanja iz prethodnih modela, baza podataka je dizajnirana u alatu MySQL Workbench, kako bi se generirala SQL skripta koja će se implementirati u MySQL sustav za upravljanje bazom podataka. Dizajn baze podataka u alatu MySQL Workbench prikazan je na slici 3.2.



Slika 3.2. Dizajn baze podataka u alatu MySQL Workbench

SQL dizajn baze podataka, odnosno generirana skripta prikazana je u nastavku.

```
CREATE  TABLE `Grad` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Naziv` VARCHAR(50) NOT NULL ,
  PRIMARY KEY (`ID`)
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;
```

```
CREATE  TABLE `Kvart` (
  `ID` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `Naziv` varchar(50) NOT NULL,
  `GradID` int(11) unsigned NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `GradID` (`GradID` ASC) ,
  CONSTRAINT `Kvart_ibfk_1`
    FOREIGN KEY (`GradID` )
    REFERENCES `Grad` (`ID` )
    ON UPDATE CASCADE
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Firma` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Naziv` VARCHAR(50) NOT NULL ,
  `OIB` TEXT  NULL DEFAULT NULL ,
  `Email` VARCHAR(50)  NULL DEFAULT NULL ,
  `Tel` VARCHAR(30)  NULL DEFAULT NULL ,
  `Kontakt osoba` VARCHAR(50)  NULL DEFAULT NULL ,
  `Napomena` VARCHAR(255)  NULL DEFAULT NULL ,
  PRIMARY KEY (`ID`)
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Restoran` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Naziv` VARCHAR(50)  NOT NULL ,
  `Email` VARCHAR(50)  NULL DEFAULT NULL ,
  `Telefon` VARCHAR(30)  NULL DEFAULT NULL ,
  `Fax` VARCHAR(30)  NULL DEFAULT NULL ,
  `FirmaID` INT(11) UNSIGNED NOT NULL ,
  PRIMARY KEY (`ID`) ,
  INDEX `FirmaID` (`FirmaID` ASC) ,
  CONSTRAINT `Restoran_ibfk_1`
    FOREIGN KEY (`FirmaID` )
    REFERENCES `Firma` (`ID` )
    ON UPDATE CASCADE
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Dostavlja` (
  `RestoranID` INT(11) UNSIGNED NOT NULL ,
  `KvartID` INT(11) UNSIGNED NOT NULL ,
  PRIMARY KEY (`RestoranID`, `KvartID`) ,
  INDEX `KvartID` (`KvartID` ASC) ,
  CONSTRAINT `Dostavlja_ibfk_2`
    FOREIGN KEY (`KvartID` )
    REFERENCES `Kvart` (`ID` )
    ON UPDATE CASCADE,
  CONSTRAINT `Dostavlja_ibfk_1`
    FOREIGN KEY (`RestoranID` )
    REFERENCES `Restoran` (`ID` )
    ON UPDATE CASCADE
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Jelo` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Naziv` VARCHAR(255)  NOT NULL ,
```

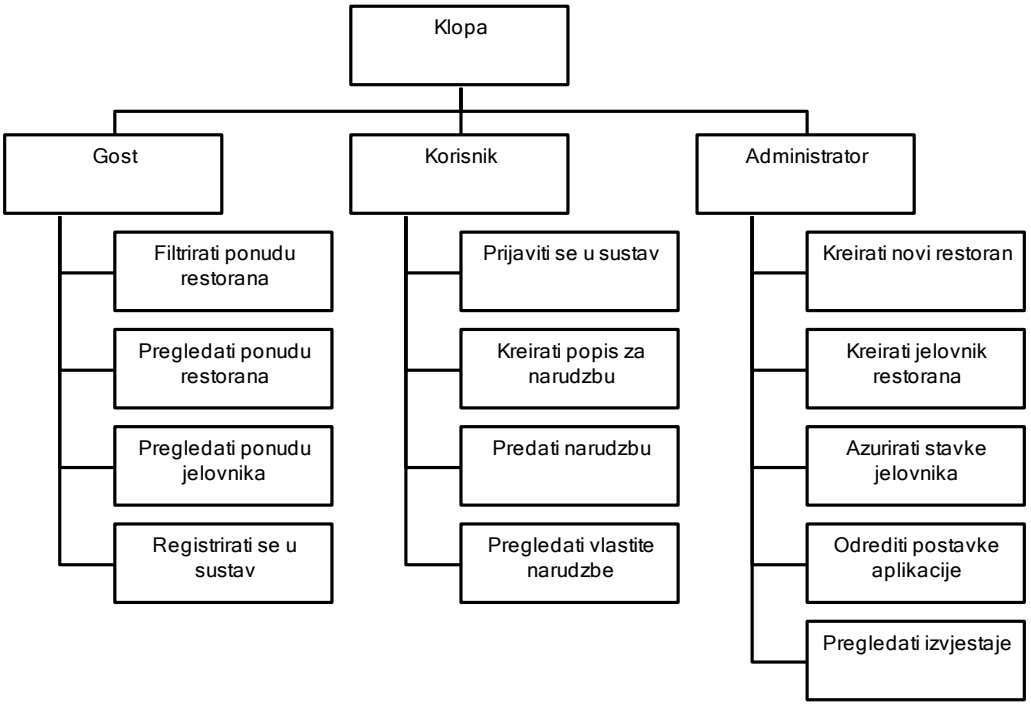
```
  PRIMARY KEY (`ID`)
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Uloga` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Naziv` VARCHAR(50) NOT NULL ,
  PRIMARY KEY (`ID`)
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Korisnik` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Korisnicko_ime` VARCHAR(50)  NOT NULL ,
  `Lozinka` VARCHAR(32)  NOT NULL ,
  `Prezime` VARCHAR(128)  NULL DEFAULT NULL ,
  `Ime` VARCHAR(128)  NULL DEFAULT NULL ,
  `Email` VARCHAR(50)  NULL DEFAULT NULL ,
  `UlogaID` INT(11) UNSIGNED NOT NULL ,
  PRIMARY KEY (`ID`) ,
  INDEX `UlogaID` (`UlogaID` ASC) ,
  CONSTRAINT `Korisnik_ibfk_1`
    FOREIGN KEY (`UlogaID` )
    REFERENCES `Uloga` (`ID` )
    ON UPDATE CASCADE
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Meni` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `RestoranID` INT(11) UNSIGNED NOT NULL ,
  `JeloID` INT(11) UNSIGNED NOT NULL ,
  `Cijena` DECIMAL(10,2) NOT NULL ,
  `Aktivan` TINYINT(1) NULL DEFAULT NULL ,
  PRIMARY KEY (`ID`) ,
  INDEX `RestoranID` (`RestoranID` ASC) ,
  INDEX `JeloID` (`JeloID` ASC) ,
  CONSTRAINT `Meni_ibfk_3`
    FOREIGN KEY (`RestoranID` )
    REFERENCES `Restoran` (`ID` )
    ON UPDATE CASCADE,
  CONSTRAINT `Meni_ibfk_4`
    FOREIGN KEY (`JeloID` )
    REFERENCES `Jelo` (`ID` )
    ON UPDATE CASCADE
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Pdv` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Naziv` VARCHAR(50)  NOT NULL ,
  `Iznos` DECIMAL(4,2) NOT NULL ,
  `Koeficijent` FLOAT NOT NULL ,
  PRIMARY KEY (`ID`)
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Provizija` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Naziv` VARCHAR(50)  NOT NULL ,
  `Iznos` DECIMAL(4,2) NOT NULL ,
```

```
  `Koeficijent` FLOAT NOT NULL ,
  PRIMARY KEY (`ID`)
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Narudzba` (
  `ID` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Datum` DATETIME NOT NULL ,
  `KorisnikID` INT(11) UNSIGNED NOT NULL ,
  `RestoranID` INT(11) UNSIGNED NOT NULL ,
  `PorezID` INT(11) UNSIGNED NOT NULL ,
  `ProvizijaID` INT(11) UNSIGNED NOT NULL ,
  PRIMARY KEY (`ID`) ,
  INDEX `KorisnikID` (`KorisnikID` ASC) ,
  INDEX `RestoranID` (`RestoranID` ASC) ,
  INDEX `PorezID` (`PorezID` ASC) ,
  INDEX `ProvizijaID` (`ProvizijaID` ASC) ,
  CONSTRAINT `Narudzba_ibfk_3`
    FOREIGN KEY (`KorisnikID` )
    REFERENCES `Korisnik` (`ID` )
    ON UPDATE CASCADE,
  CONSTRAINT `Narudzba_ibfk_4`
    FOREIGN KEY (`RestoranID` )
    REFERENCES `Restoran` (`ID` )
    ON UPDATE CASCADE,
  CONSTRAINT `Narudzba_ibfk_5`
    FOREIGN KEY (`PorezID` )
    REFERENCES `Pdv` (`ID` )
    ON UPDATE CASCADE,
  CONSTRAINT `Narudzba_ibfk_6`
    FOREIGN KEY (`ProvizijaID` )
    REFERENCES `Provizija` (`ID` )
    ON UPDATE CASCADE
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;


CREATE  TABLE `Stavke_narudzbe` (
  `NarudzbaID` INT(11) UNSIGNED NOT NULL ,
  `MeniID` INT(11) UNSIGNED NOT NULL ,
  `Kolicina` TINYINT(2) NOT NULL ,
  PRIMARY KEY (`NarudzbaID`, `MeniID`) ,
  INDEX `MeniID` (`MeniID` ASC) ,
  CONSTRAINT `Stavke_narudzbe_ibfk_1`
    FOREIGN KEY (`NarudzbaID` )
    REFERENCES `Narudzba` (`ID` )
    ON UPDATE CASCADE,
  CONSTRAINT `Stavke_narudzbe_ibfk_2`
    FOREIGN KEY (`MeniID` )
    REFERENCES `Meni` (`ID` )
    ON UPDATE CASCADE
) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;
```

3.3. Dekompozicija sustava

„Dekompozicija (raščlanjivanje) osnovna je metoda savladavanja složenosti sustava, kojom se problem rješava postupno, od općeg prema pojedinačnom. Raščlanjivanjem dobivamo hijerarhijski prikaz strukture objekta promatranja, a to znači da se strukture više razine složenosti raščlanjuju na strukture niže razine složenosti.“
[http://arka.foi.hr/PzaWeb/PzaWeb2003_06/dokumentacija/dokumentacija.php#dd]

Dekompozicijski dijagram sustava nam pokazuje aktivnosti koje mogu pojedine vrste korisnika izvoditi na sustavu. Kako su korisnici podjeljeni u tri grupe (administrator, korisnik i gost) tako su im dodjeljene aktivnosti koje mogu izvoditi na sustavu. Sam dekompozicijski dijagram je prikazan na slici 3.3.



Slika 3.3. Dekompozicijski dijagram sustava

Iz dijagrama je vidiljivo kako se sustav sastoji od tri podustava, a to su procesi gosta, procesi korisnika i procesi administratora. Također vidimo da administrator sustava kreira i uređuje ponudu restorana i jelovnika, te da određuje postavke aplikacije, pa prema tome on ima najšire ovlasti u sustavu.

3.4. Funkcionalnost web aplikacije

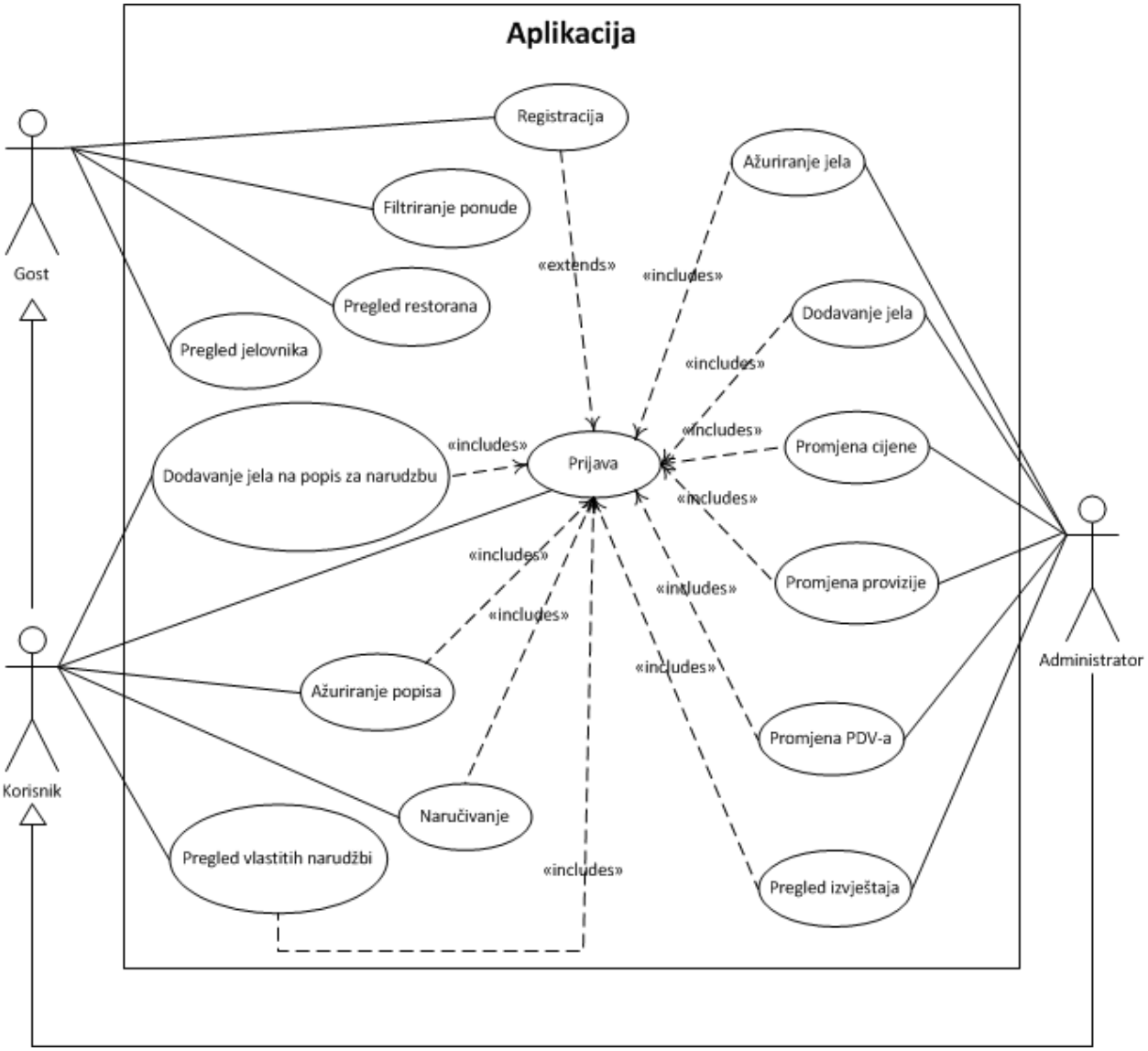
Funkcionalnost web aplikacije prikazana je UML dijagramom slučajeva korištenja (engl. Use-case diagram). Dijagrami slučajeva korištenja koriste se da bi se prikazalo ponašanje sustava, dijelova sustava ili konkretnog razreda na način vidljiv korisniku sustava.

Prema UML-specifikaciji dijagrami slučajeva korištenja sastoje se od učesnika, slučajeva korištenja te veza i odnosa među učesnicima i slučajevima korištenja. Dodatno se na dijagramu slučajeva korištenja može istaknuti granica sustava koji se izgrađuje.

Učesnik predstavlja osobu ili drugi sustav koji je u interakciji sa promatranim sustavom. „Učesnik je netko ili nešto što je u interakciji sa sustavom; tko ili što koristi sustav. Učesnik predstavlja ulogu, a ne individualnog korisnika sustava.“ **[Eriksson, Penker, Lyons, Fado; 2004 ; str.63]**
Dakle učesnik predstavlja skup uloga koje korisnici slučajeva korištenja igraju tijekom njihove interakcije sa slučajevima korištenja.

Slučaj korištenja predstavlja zadatak kojeg izvode učesnici, on predstavlja funkcionalnost koju učesniku pruža modelirani sustav. U UML-u slučaj korištenja definiran je kao skup radnji izvršenih od strane sustava što daje vidljivi rezultat koji je, u pravilu, vrijednost za jednog ili više sudionika ili drugih aktera u sustavu.

Na dijagramu slučajeva korištenja prikazanom na slici 3.4., vidljiva je osnovna funkcionalnost koju treba zadovoljiti web aplikacija razvijena za potrebe ovog rada.



Slika 3.4. Dijagram slučajeva korištenja

Iz dijagrama slučajeva korisnika vidljivo je kako je prikazani sustav u međudjelovanju sa tri učesnika. Učesnici su gost, korisnik i administrator. Učesnici su međusobno povezani vezom generalizacije pri čemu je definicija apstraktnog učesnika dopunjena specifičnijim ulogama izvedenih aktora. Tipično semantičko značenje ove veze je da potomak može raditi sve što roditelj radi, tj. potomci nasljeđuju veze sa slučajevima korištenja od roditelja.

Gost predstavlja neregivanjskog posjetitelja koji može pregledavati ponudu restorana i sastav njihovih jelovnika. Također gost može filtrirati ponudu prema lokaciji dostave kako bi pronašao samo one restorane koji dostavljaju na odabranoj lokaciji. Gost nema ovlasti za naručivanje sve dok ne postane korisnik.

Da bi gost postao korisnik on se mora registrirati u sustav. Nakon registracije i prijave u sustav korisniku se dodjeljuju ovlasti za naručivanje. Nakon što je dobio ovlasti korisnik sada može dodavati željena jela na popis za narudžbu, može mijenjati količinu svakog pojedinog jela na popisu, može uklanjati stavke sa popisa i ponovno ih dodavati. Kada je završio sa sastavljanjem popisa za narudžbu korisnik može isti prosljediti u narudžbu pripadajućem restoranu. Korisnik također može pregledavati sve vlastite narudžbe.
Korisnik je potomak gosta, te on od njega nasljeđuje veze sa slučajevima korištenja, što znači da korisnik također može pregledavati i filtrirati ponudu i sl.

Treći učesnik je administrator. Administrator je potomak korisnika čime nasljeđuje sve njegove veze sa slučajevima korištenja, kao i one koje nasljeđuje korisnik. Iz ovoga se jasno može zaključti kako administrator ima najveće ovlasti u sustavu. Naime on ima ovlasti svih ostalih učesnika uz dodatne ovlasti kao što su: dodavnje restorana, dodavanje jela na jelovnik, promjena cijene i statusa pojedinog jela na jelovniku, promjena postotka provizije na narudžbe, promjena stope poreza na dodanu vrijednost, pregled i filtriranje izvješataja.

Vezom uključivanja (engl. include), na dijagramu povezuju se dva slučaja korištenja kada jedan slučaj korištenja sadrži ponašanje definirano u drugom slučaju korištenja. Ponašanje uključenog slučaja korištenja uključeno je u ponašanje osnovnog slučaja korištenja. Osnovni slučaj korištenja ovisi o ponašanju uključenog slučaja korištenja. Uključeni slučaj korištenja uvijek je potreban kako bi se osnovni slučaj mogao izvršiti. U slučaju da osnovni slučaj korištenja dođe u točku izvođenja u kojoj je potrebno izvesti uključeni obrazac uporabe tada se uključeni obrazac uporabe izvodi u cijelosti. Moguće je da u toku izvođenja nisu zadovoljeni uvjeti pozivanja uključenog obrasca uporabe.

Primjerice u toku izvođenja slučaja korištenja „Pregled vlastitih narudžbi“ uključeni slučaj korištenja „Prijava“ poziva se u slučaju da korisnik nije prijavljen u sustav, što je prikazano na sljedećem isječku koda.

```php
<?php
	//narudzbe.php
	session_start();
	$_SESSION['back'] =htmlentities($_POSLUŽITELJ['REQUEST_URI']);
	if(!isset($_SESSION['aktivni_korisnik'])) {
		header("Location:login.php");
	}
```

```
        ...
?>
```

Nakon što uključeni slučaj korištenja „Prijava“ bude izveden u cijelosti i korisnik se uspješno prijavi kontrola se vraća osnovnom slučaju korištenja, što je prikazano na sljedećem isječku koda.

```
<?php
        //login.php
        ...
        if (mysql_num_rows($rs) == 1) {
                session_start();
                $page=$_SESSION['back'];
                $row = mysql_fetch_array($rs);
                $_SESSION['aktivni_korisnik'] = $row['ID'];
                $_SESSION['aktivni_korisnik_tip']= $row['ulogaID'];
                if (isset($page) && !empty($page) ) {
                        header ("Location:$page");
                } else {
                        header ("Location:index.php");
                }
        } else {
                $greska = "Neispravni podaci za prijavu!";
        }

        ...
?>
```

Vezom proširenja (engl. extend), na dijagramu se povezuju dva slučaja korištenja pri čemu jedan proširuje funkcionalnost drugog. U ovom primjeru funkcionalnost slučaja korištenja „Prijava“ proširuje se slučajem korištenja „Registracija“. Normalna funkcionalnost slučaja korištenja „Prijava“ podrazumijeva da korisnik ima odgovarajuće podatke za prijavu. Ako korisnik ne posjeduje podatke za prijavu, „Registracija“ proširuje slučaj korištenja „Prijava“ dopuštajući osobi registraciju u sustav kako bi dobila podatke za prijavu. Na sljedećem isjećku koda prikazana je realizacija proširenja.

```
        ...
        <h3>Prijava</h3>
        <p>Prijava je dostupna samo registriranim korisnicima.<br/>
           Ako još niste registrirani, <a href="register.php" class="a">registrirajte
           se OVDJE besplatno.</a>
        </p>
        <form class="unos" action="login.php" method="post" >
                Korisničko ime:
                <input type="text" name="kor_ime">
                Lozinka:
                <input type="password" name="lozinka">
```

```
        <input type="submit" class="submit" name="prijavi" value="Prijavi se" />
    </form>
    ...
```

Zanimljivo je istaknuti da je osnovni slučaj („Prijava“), kompletan i bez proširenja, što je slučaj kada korisnik ima podatke za prijavu.

3.5. Programski kod aplikacije

Aplikacija je razvijena korištenjem PHP programskog jezika, čije su značajke opisane u prvom dijelu ovog rada. Također korišten je JavaScript programski jezik u svrhu postazanja bolje interakcije s korisnikom i AJAX[10] tehnologija kako bi se ažurirali neki dijelovi stranice bez potrebe za ponovnim učitavanjem iste. Programski kod aplikacije sadržan je u datotekama koje su smještene u direktoriju pod imenom „klopa“ koji je glavni „korijenski“ direktorij aplikacije. Struktura direktorija „klopa“ prikazana je na slici 3.5.

Slika 3.5. Struktura direktorija klopa

[10] AJAX (Asynchronous JavaScript and XML)

Eliminiranje dupliciranog koda postignuto je na način da su kreirane datoteke „header.php“, „footer.php“, „configBP.php“ i „funkcije.php“. Prve dvije datoteke odgovorne su za prikaz sadržaja koji je uključen u sve web stranice aplikacije, a to je zaglavlje, podnožje, navigacijski izbornik i dio za prikaz sadržaja. Datoteka „configBP.php“ sadrži funkcije potrebne za rad sa bazom podataka. Pomoću tih funkcija stavra se konekcija na bazu podataka i obavljaju upiti nad bazom podataka.

Datoteka „configBP.php“ definirana je na sljedeći način:

```php
<?php
$BP_poslužitelj = 'localhost';
$BP_bazaPodataka = 'klopa';
$BP_korisnik =  'klopa_2012';
$BP_lozinka = '123456';

$dbc = null;
$db = null;

function otvoriBP() {
	global $dbc;
	global $db;
	global $BP_poslužitelj;
	global $BP_bazaPodataka;
	global $BP_korisnik;
	global $BP_lozinka;

	$dbc = mysql_connect($BP_poslužitelj, $BP_korisnik, $BP_lozinka);
	if(! $dbc) {
		pogreska('Pogreška! ' . mysql_error());
		exit();
	}

	$db = mysql_select_db($BP_bazaPodataka, $dbc);
	if(! $db) {
		pogreska('Pogreška! ' . mysql_error());
		exit();
	}
	mysql_query("set names 'utf8'",$dbc);
}

function izvrsiBP($sql) {
	$rs = mysql_query($sql);
	if(!$rs) {
		echo('Pogreška! ' . mysql_error());
		exit();
	}
	return $rs;
}
function zatvoriBP(){
	global $dbc;
	mysql_close($dbc);
}

?>
```

Sa stajališta funkcionalnosti, datoteka „funkcije.php“ među najvažnijim je datotekama ove aplikacije. Sadržaj ove datoteke čine funkcije koje su odgovorne ponajviše za rad sa popisom za narudžbu. Naime pomoću tih funkcija se dodaju, uklanjaju i ažuriraju stavke popisa za narudžbu. Funkcije također služe za dobivanje raznih podataka iz baze podataka, kao što su cijena jela, naziva jela, naziv restorana, izračun sume narudžbe, te za primjerice zaštitu baze podataka (funkcija fix_string). Sadržaj ove datoteke uključuje se prema potrebi u web stranice aplikacije.

Datoteka „funkcije.php“ definirana je na sljedeći način:

```
<?php
function remove_product($iid){
	$iid=intval($iid);
	$max=count($_SESSION['list']);
	for($i=0;$i<$max;$i++){
		if($iid==$_SESSION['list'][$i]['itemid']){
			unset($_SESSION['list'][$i]);
			break;
		}
	}
	if ($max==1){
		unset($_SESSION['list']);
	} else {
		$_SESSION['list']=array_values($_SESSION['list']);
	}
}

function update_product($iid, $q){
	$iid=intval($iid);
	$q=intval($q);
	$max=count($_SESSION['list']);
	for($i=0;$i<$max;$i++){
		if($iid==$_SESSION['list'][$i]['itemid']){
			$_SESSION['list'][$i]['qty']=$q;
			break;
		}
	}
}

function addtolist($rid,$iid,$q){
	if($rid<1 or $iid<1 or $q<1) return;

	if(is_array($_SESSION['list'])){
		if(restoran_isti($rid)) return "Molimo Vas dovršite započetu narudžbu
prema prvom restoranu";
		if(item_exists($iid)) return "Traženo jelo već postoji na popisu.
Ukoliko želite, promijenite količinu.";
		$max = count($_SESSION['list']);
		$_SESSION['list'][$max]['restid']=$rid;
		$_SESSION['list'][$max]['itemid']=$iid;
		$_SESSION['list'][$max]['qty']=$q;
	} else {
		$_SESSION['list']=array();
		$_SESSION['list'][0]['restid']=$rid;
		$_SESSION['list'][0]['itemid']=$iid;
```

```
                $_SESSION['list'][0]['qty']=$q;
        }
}

function item_exists($iid){
        $iid=intval($iid);
        $max=count($_SESSION['list']);
        $flag=0;
        for($i=0;$i<$max;$i++){
                if($iid==$_SESSION['list'][$i]['itemid']){
                        $flag=1;
                        break;
                }
        }
        return $flag;
}

function restoran_isti($rid){
        $rid=intval($rid);
        $max=count($_SESSION['list']);
        $flag=0;
        for($i=0;$i<$max;$i++){
                if($rid != $_SESSION['list'][$i]['restid']){
                        $flag=1;
                        break;
                }
        }
        return $flag;
}

function getCijena($iid){
        $sql="SELECT Cijena FROM Meni WHERE ID='$iid'";
        $rs=izvrsiBP($sql);
        $row=mysql_fetch_array($rs);
        return $row['Cijena'];
}

function getNazivJela($iid){
        $sql="SELECT j.Naziv
                FROM Meni m, Jela j
                WHERE m.ID = $iid AND m.jeloID = j.ID";
        $rs=izvrsiBP($sql);
        $row=mysql_fetch_array($rs);
        return $row['Naziv'];
}

function getNarudzbaTotal($nID){
        $sql="SELECT SUM(s.kolicina*m.cijena) AS Suma
                FROM stavke_narudzbe AS s, meni AS m
                WHERE s.narudzbaID=$nID AND s.meniID=m.ID";
        $rs=izvrsiBP($sql);
        $row=mysql_fetch_array($rs);
        return $row['Suma'];
}

function getPdvNaziv($pdv) {
        $sql="SELECT Naziv FROM Pdv WHERE ID='$pdv'";
        $rs=izvrsiBP($sql);
        $row=mysql_fetch_array($rs);
        return $row['Naziv'];
}
```

```
function getProvizijaNaziv($provizija) {
	$sql="SELECT Naziv FROM Provizija WHERE ID='$provizija'";
	$rs=izvrsiBP($sql);
	$row=mysql_fetch_array($rs);
	return $row['Naziv'];
}

function getRestNaziv($restID) {
	$sql="SELECT Naziv FROM Restorani WHERE ID=$restID";
	$rs=izvrsiBP($sql);
	$row=mysql_fetch_array($rs);
	return $row['Naziv'];
}

function mysql_fix_string($string){
	if (get_magic_quotes_gpc()){
		$string = stripslashes($string);
	}
	$string = mysql_real_escape_string($string);
	return $string;
}
?>
```

Među važnim datotekama valja istaknuti i datoteku „narudzba.php“ koja je odgovorna za unos podataka o narudžbama u bazu podataka. U toj datoteci primijenjena jedna od najvažnijih zanačajki InnoDB tablica, a to su transakcije. Transakcija je korištena kako bi se osiguralo da u bazi podataka ne postoji narudžba koja ili nema stavaka ili nema sve potrebne stavke upisane u pripadajuču tablicu „Stavke_narudzbe“.

Datoteka „narudzba.php“ definirana je na sljedeći način:

```
<?php
session_start();
if(!isset($_SESSION['aktivni_korisnik'])) {
	header("Location:login.php");
}

include ("header.php");
include ("funkcije.php");
include ("configBP.php");
otvoriBP();

if (isset($_POST['naruceno']) && isset($_SESSION['list'])) {

	$restoran=$_POST['restoran'];
	$korisnik=$_POST['korisnik'];
	//ARRAY POST VARIABLES
	$jelo = $_POST['jelo'];
	$kolicina = $_POST['kolicina'];
	$cnt = count($jelo);
	for($i=0;$i<$cnt;$i++){
		update_product($jelo[$i],$kolicina[$i]);
```

```
        }
        //PARSING CONFIG FILE
        $configFile = 'postavke.ini';
        $ini_array = parse_ini_file($configFile);
        $pdv = $ini_array ['pdv'];
        $provizija = $ini_array ['provizija'];
        //STARTING TRANSACTION
        $sql="SET AUTOCOMMIT=0";
        $rs = izvrsiBP($sql);
        $sql= "START TRANSACTION";
        $rs = izvrsiBP($sql);
        //FIRST STATEMENT
        $sql= "INSERT INTO Narudzbe
                    (Datum, KorisnikID, RestoranID, PorezID, ProvizijaID)
                    VALUES
                    (now(), '$korisnik', '$restoran','$pdv','$provizija')";
        $rs = izvrsiBP($sql);
        $narudzba = mysql_insert_id();
        $max = count($_SESSION['list']);
        if ($max > 0) {
              for($i=0;$i<$max;$i++){
                    $jelo=$_SESSION['list'][$i]['itemid'];
                    $kolicina=$_SESSION['list'][$i]['qty'];
                    $sql= "INSERT INTO Stavke_narudzbe VALUES ('$narudzba', '$jelo',
'$kolicina')";
                    $rs = izvrsiBP($sql);
              }
              $sql="COMMIT";
              $rs = izvrsiBP($sql);
              $poruka="Vaša narudžba uspješno je obavljena.";
        } else {
              $sql="ROLLBACK";
              $rs = izvrsiBP($sql);
        }
        unset($_SESSION['list']);
        echo "<div class='success'>$poruka</div>";
}
Include ("footer.php");
?>
```

Snaga ove aplikacije leži u njenoj prilagodljivosti novonastalim promjenama stope provizije, stope PDV-a i promjenama cijene jela. Za ove karakteristike odgovorna je dijelom baza podataka koja omogućava promjenu cijene i statusa artikla i pohranu trenutnih vrijednosti postavki aplikacije, a drugim dijelom datoteka „postavke.php“ koja pohranjuje trenutne postavke PDV-a i provizije u konfiguracijsku datoteku.

Datoteka „postavke.php“ definirana je na sljedeći način:

```
<?php
session_start();
$_SESSION['back'] =htmlentities($_POSLUŽITELJ['REQUEST_URI']);
if(!isset($_SESSION['aktivni_korisnik']) || ($_SESSION['aktivni_korisnik_tip'] !=1))
{
```

```
	header("Location:login.php");
}

include('funkcije.php');
include('configBP.php');
otvoriBP();

$configFile = 'postavke.ini';

$potvrda = "";

if (!isset($_POST['postavi'])) {

	$ini_array = parse_ini_file($configFile);
	$pdv = $ini_array ['pdv'];
	$provizija = $ini_array ['provizija'];
	$pdvNaziv = getPdvNaziv($pdv);
	$provizijaNaziv = getProvizijaNaziv($provizija);
}

if (isset($_POST['postavi'])) {

	$pdv = $_POST['pdv'];
	$provizija = $_POST['provizija'];
	$pdvNaziv = getPdvNaziv($pdv);
	$provizijaNaziv = getProvizijaNaziv($provizija);

	$ini_array = array(
		'pdv' => $pdv,
		'provizija' => $provizija,
		);

	// open and lock configuration file for writing
	$fp = fopen($configFile, 'w+') or die('ERROR: Cannot open configuration file
for writing');
	flock($fp, LOCK_EX) or die('ERROR: Cannot lock configuration file for
writing');

	// write each configuration value to the file
	foreach ($ini_array  as $key => $value) {
		if (trim($value) != '') {
			fwrite($fp, "$key=$value\n") or die('ERROR: Cannot write [$key]
to configuration file');
		}
	}
	// close and save file
	flock($fp, LOCK_UN) or die ('ERROR: Cannot unlock file');
	fclose($fp);
	$potvrda = "Postavke aplikacije uspješno su zapisane.";
}

include ("header.php");
?>
<h3>Postavke</h3>
<form class="unos" method="post" action="postavke.php" name="setupform">
	<label for="pdv">Odaberi stopu PDV-a :</label>
	<select name="pdv">
		<option value="">----------------------------</option>
<?php
		$sql = "SELECT ID, Naziv FROM Pdv";
		$rs = izvrsiBP($sql);
```

```
            while(list($id, $naziv) = mysql_fetch_array($rs)) {
                  echo "<option value='$id'";
                  if($id == $pdv) {
                        echo "selected='selected'";
                  }
                  echo">$naziv</option>";
            }
?>
      </select>
      <label for="provizija">Odaberi postotak provizije:</label>

      <select name="provizija">
            <option value="">----------------------------</option>
<?php
            $sql = "SELECT ID, Naziv FROM Provizija";
            $rs = izvrsiBP($sql);
            while(list($id, $naziv) = mysql_fetch_array($rs)) {
                  echo "<option value='$id'";
                  if($id == $provizija) {
                        echo "selected='selected'";
                  }
                  echo">$naziv</option>";
            }
?>
      </select>
      <input type="submit" class="submit" name="postavi" value="Postavi" />
</form>
<?php
if ($potvrda != "") {
      echo "<br/><div class='success'>$potvrda</div>";
}
include ('footer.php');
?>
```

Kao snagu aplikacije valja istaknuti i izvještaje koji administrratoru aplikacije daju uvid u ostvarenu proviziju, broj narudžbi, prosječnu narudžbu i sl. Izvještaje je moguće filtrirati, a za njihovu realizaciju odgovorna je datoteka „izvjestaji.php“ koja između ostalih sadrži sljedeće upite:

```
// OSNOVNI UPIT ZA STAVKE
$sql_list = "SELECT n.ID, n.datum AS datum,
            k.korisnicko_ime AS korisnik, r.naziv AS restoran
            FROM narudzbe AS n, korisnici AS k, restorani AS r
            WHERE n.restoranID=r.ID AND n.korisnikID=k.ID";


// OSNOVNI UPIT ZA SUME
$sql_sum = "SELECT SUM(s.kolicina*m.cijena) AS Suma,
            SUM(s.kolicina*m.cijena*pdv.koeficijent) AS Porez,
            SUM(s.kolicina*m.cijena*p.koeficijent) AS Provizija,
            AVG(s.kolicina*m.cijena) AS Prosjek
            FROM narudzbe AS n, stavke_narudzbe AS s, meni AS m, provizija AS p, Pdv
            WHERE n.id=s.narudzbaID AND s.meniID=m.ID
            AND n.provizijaID=p.ID AND n.porezID=pdv.ID";
```

```php
// FILTER PO RESTORANU
if (isset($_POST['restoran'])  && !empty($_POST['restoran'])) {
      $restoran = $_POST['restoran'];
      $sql_sum.= " AND n.restoranID =$restoran";
      $sql_list.=" AND n.restoranID =$restoran";
}

// FILTER NARUDŽBA OD
if (isset($_POST['narudzba_od']) && strlen($_POST['narudzba_od'] > 0) &&
!empty($_POST['narudzba_od'])) {
      date_default_timezone_set("Europe/Zagreb");
      $narudzba_od =  date ('Y-m-d', strtotime($_POST['narudzba_od']));
      $sql_sum .= " AND DATE(n.datum) >= '$narudzba_od'";
      $sql_list.=" AND DATE(n.datum) >= '$narudzba_od'";
}

// FILTER NARUDŽBA DO
if (isset($_POST['narudzba_do']) && strlen($_POST['narudzba_do'] > 0)) {
      date_default_timezone_set("Europe/Zagreb");
      $narudzba_do =  date ('Y-m-d', strtotime($_POST['narudzba_do']));
      $sql_sum .= " AND DATE(n.datum) <= '$narudzba_do'";
      $sql_list.= " AND DATE(n.datum) <= '$narudzba_do'";
}
```

4. Korisničke upute

U ovom dijelu rada dane su upute korisnicima kako koristiti sustav, odnosno aplikaciju za online naručivanj „klopa“. Sve što korisnik mora imati od programske potpore za rad sa aplikacijom je internet preglednik i vezu na internet.

4.1. Početna stranica

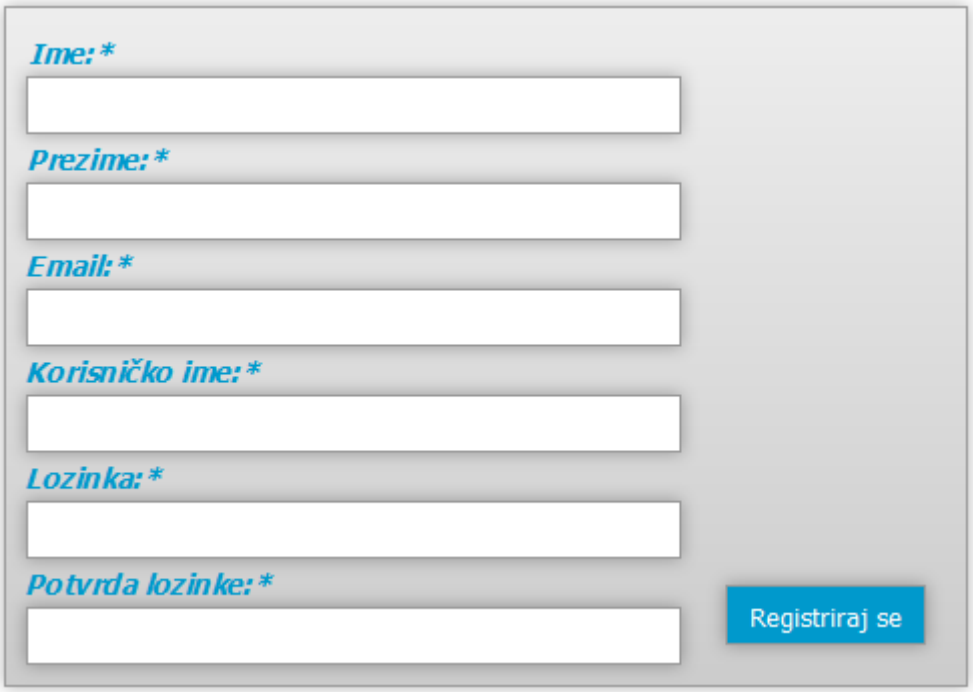
Aplikaciji se pristupa putem adrese http://localhost/klopa/ nakon čega se otvara početna stranica aplikacije prikazana na slici 4.1.

Slika 4.1. Početna stranica aplikacije

Na ovoj stranici moguće je filtrirati ponudu prema željenoj lokaciji dostave.

4.2. Registracija u sustav

Za naručivanje putem aplikacije neophodna je registracija i prijava u sustav. Registracija u sustav obavlja se putem obrazca prikazanog na slici 4.1.

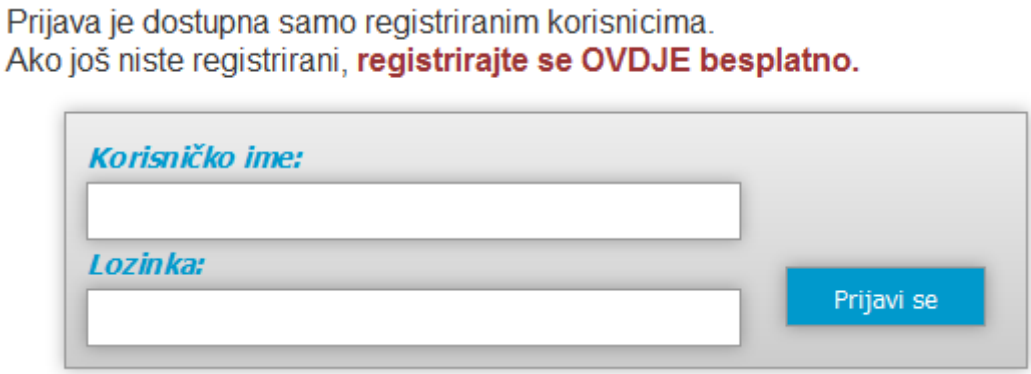


Slika 4.2. Registracija u sustav

Nakon uspješne registracije korisnika se automatski prijavljuje u sustav i preusmjerava na početnu stranicu.

4.3. Prijava u sustav

Registrirani korisnici prijavljuju se u sustav putem obrazca prikazanog na slici 4.2.



Slika 4.3. Prijava u sustav

Nkon uspješne prijave korisniku se dodjeljuju prava za naručivanje i preusmjerava ga se na početnu stranicu ili na stranicu na kojoj je bio prije prijave u sustav.

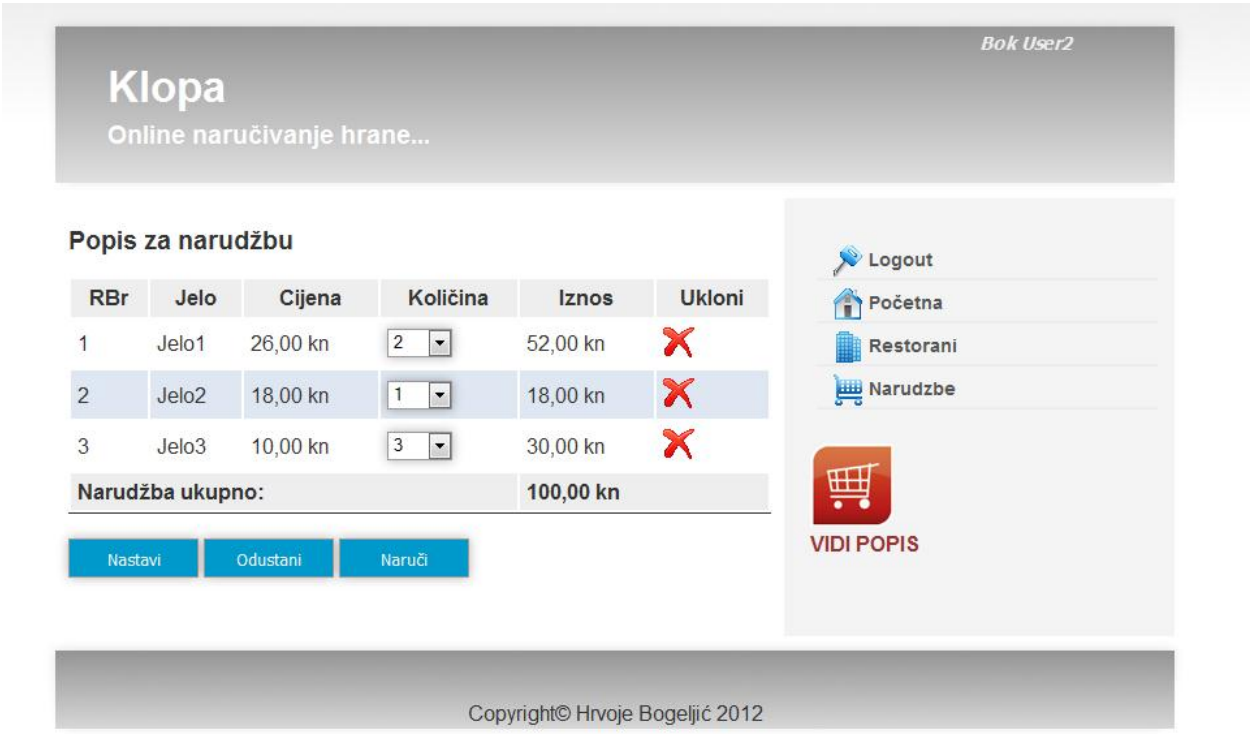
4.4. Naručivanje

Nakon što je filtrirao ponudu prema lokaciji dostave korisniku se nudi popis restorana iz kojih može naručivati. Nakon što je odabrao neki od mogućih restorana, klikom na znak plus može dodavati jela sa jelovnika na popis za narudžbu. Jelovnik Restorana1 sa opcijom dodavanja jela na popis prikazan je na slici 4.4.

Slika 4.4. Dodavanje jela na popis za narudžbu

Nakon što je dodao pojedino jelo korisnika se prusmjerava na popis za narudžbu, gdje mu se nude opcije odustajanja od narudžbe, promjene količine jela, uklanjanje jela sa popisa i nastavka narudžbe, odnosno dodavanje drugih jela na popis za narudžbu.

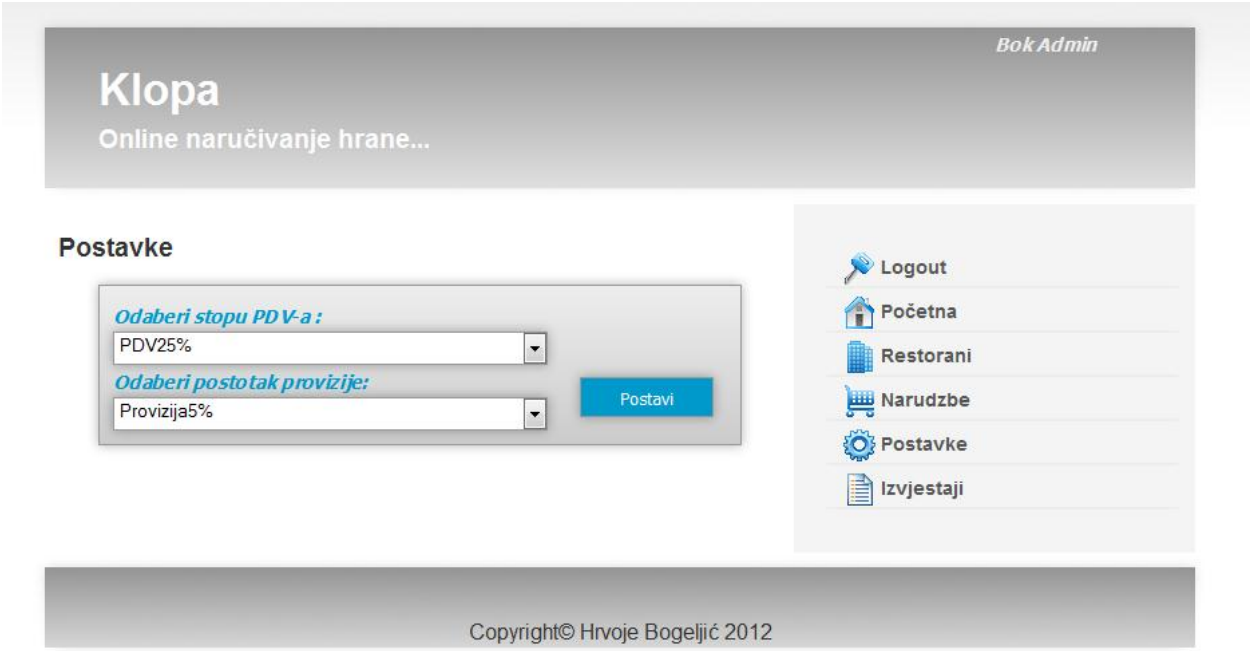
Na slici 4.5. prikazan je popis za narudžbu sa pripadajućim opcijama za upravljanje popisom.



Slika 4.5. Upravljanje popisom za narudžbu

4.5. Promjena postavki aplikacije

Administrator je korisnik u sustavu koji ima ovlasti za promjenu postavki aplikacije. Izgled obrazca za promjenu postavki aplikacije prikazan je na slici 4.6.



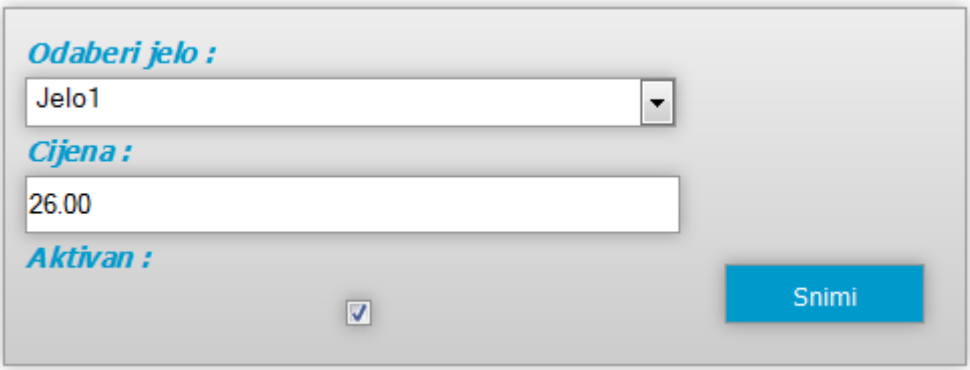
Slika 4.6. Upravljanje postavkama aplikacije

4.6. Upravljanje jelovnikom

Upravljanje jelovnikom također je uloga administratora sustava. Pod upravljanje jelovnikom spada dodavanje jela na jelovnik, zatim promjena cijene jela i promjena statusa jela. Opcije administratora za upravljanje jelovnikom prikazane su na slici 4.7.

Slika 4.7. Opcije adminstratora za upravljenje jelovnikom

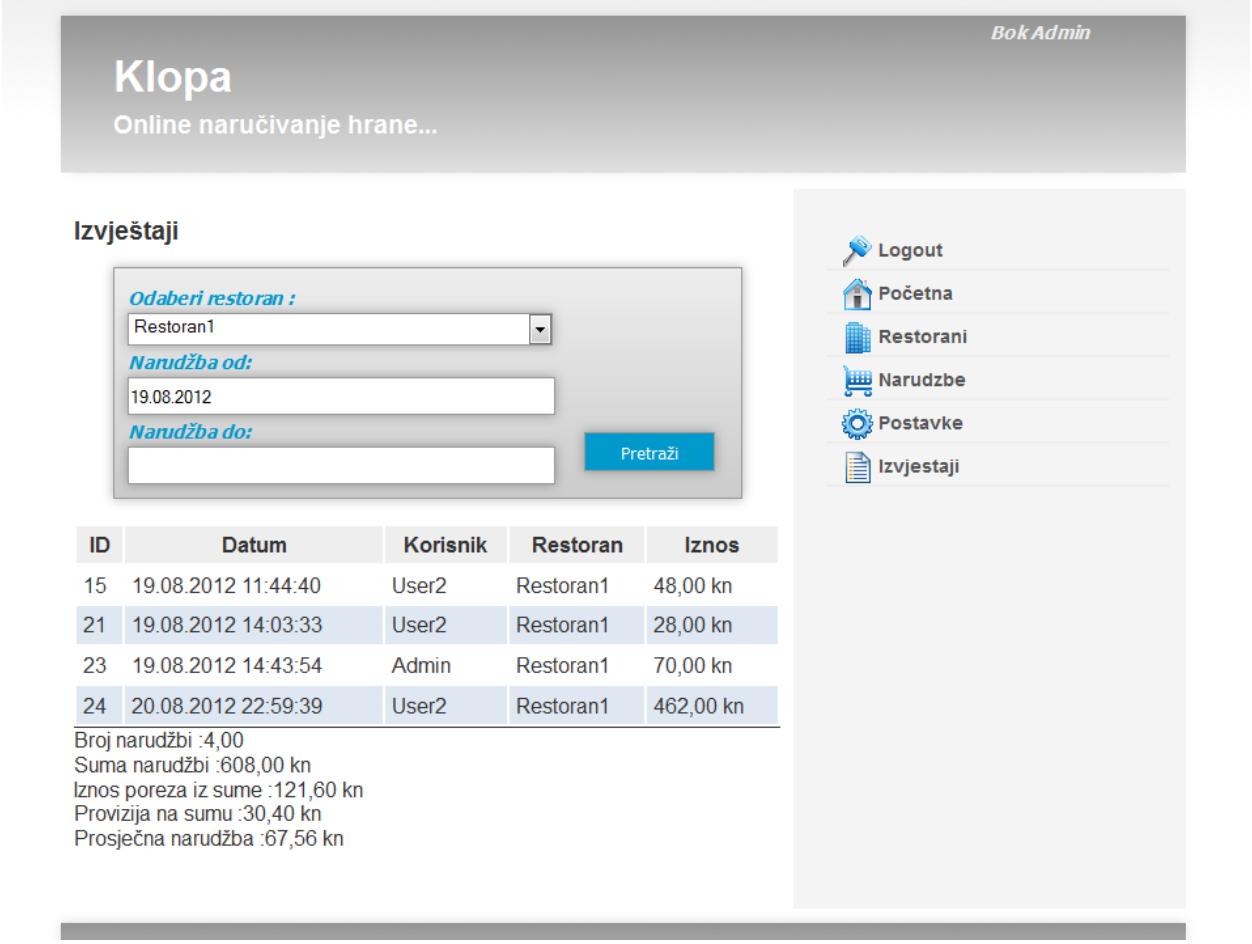
Sve promjene na jelovniku obavljaju se putem obrazca prikazanog na slici 4.8.



Slika 4.8. Obrazac za upravljanje jelovnikom

4.7. Pregled izvještaja

Pregled izvještaja također je uloga administratora sustava. Administrator generira izvještaj putem obrazca prikazanog na slici 4.9.



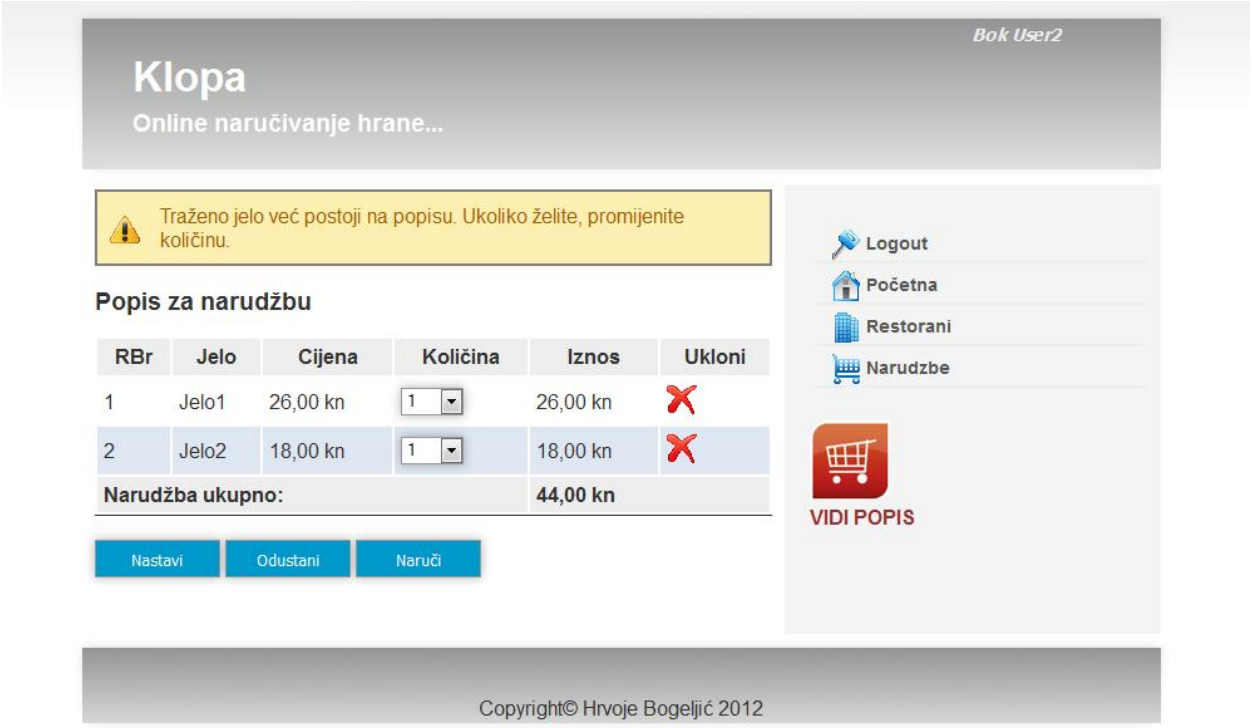
Slika 4.9. Generiranje izvještaja

5. Korisnička perspektiva apliacije

Za rad sa aplikacijom napravljeno je korisničko sučelje sa namjerom da bude jednostavno za uptrebu i da omogući navigaciju u aplikaciji. Sučelje aplikacije je konzistentno tijekom upotrebe i sastoji se od zaglavlja, podnožja, navigacijskog izbornika i dijela za prikaz sadržaja. Elementi sučelja prikazani su na slikama iz prethodnog poglavlja.

Pri korištenju aplikacije korisnika se čitavo vrijeme usmjerava i izvještava o posljedicama njegovog djelovanja. Korisnika je bitno obavijestiti o tim posljedicama jer u protivnome on pogađa šta se dešava, a korisnik ćesto pogodi krivo.

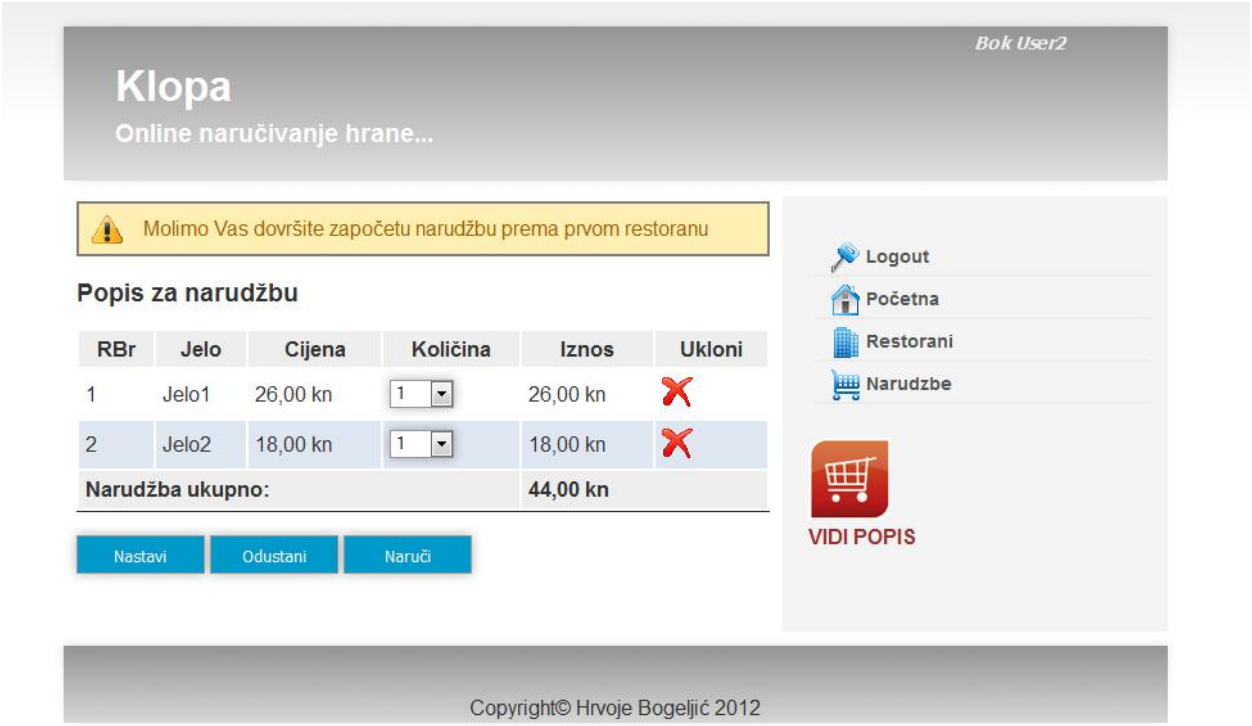
Primjer obavještavanja korisnika prikazan je slikom 5.1. u slučaju kada korisnik pokuša dodati jelo koje se već nalazi na popisu za narudžbu.



Slika 5.1. Obavještavanje korisnika1

Na slici 5.1. također se vidi opcija pregleda popisa klikom na ikonu košarice. Ova opcija generira se čim korisnik doda neku stavku na popis, a služi tome da korisnik u svakom trenutku, neovisno o tome na kojoj se stranici aplikacije nalazio ima mogučnost pregleda popisa i eventualnog nastavka narudžbe. Opcija nastavi narudžbu uvijek vodi na jelovnik onog restorana na koji se odnose stavke sa popisa, iz razloga što jedna narudžba može biti upućena samo

jednom restoranau, odnosno na jednom popisu mogu se nalaziti stavke jelovnika samo jednog restorana. U slučaju da korisnik pokuša dodati stavku sa jelovnika nekog drugog restorana biti će obaviješten porukom prikazanom na slici 5.2.



Slika 5.2. Obavještavanje korisnika2

6. Kritički prikaz

Ovom aplikacijom prikazana je osnovna funkcionalnost koju treba zadovoljiti jedan sustav za online naručivanje ovakve koncepcije. Izrada sustava temeljila se na upotrebi tehnologija prezentiranih u prvom dijelu rada. Nakon prezentacije navedenih tehnologija, u drugom dijelu rada pristupilo se i samoj izradi sustava. Prvo je dizajnirana baza podataka, pri čemu se vodilo računa o prilagodljivosti aplikacije, nakon čega je razvijen i sam programski dio sa naglaskom na upravljanje popisom za narudžbu.

U nastavku rada pažnja je posvečena korisniku sustava. Naime pružene su upute korisnicima za rad sa aplikacijom, te je ista sagledana iz perspektive korisnika.

Ovakvi sustavi podložni su čestim promjena, pa bi iz tog razloga u budućnosti aplikaciju bilo dobro razviti na objektno-orijentirani način, primjenom MVC[11] arhitekture. Na taj način odvojilo bi se logiku od prezentacije, te bi se promjene implementirale bez opasnosti narušavanja logike aplikacije.

[11] Model–View–Controller (MVC) je naziv koji se često koristi u softverskom inženjeringu. Namjenjen je izoliranju programske logike (business logic) od ulaza (input) i prezentacijskog djela, omogućava nezavisan razvoj, testiranje i održavanje određene aplikacije.

7. Literatura

1. http://en.wikipedia.org/wiki/Apache_HTTP_Poslužitelj, Dostupno 03.08.2012.
2. http://www.w3schools.com/ajax/default.asp, Dostupno 19.07.2012.
3. Galić D. (2010). *Moj OpenSource projekt na temu „Forum“(završni rad)*
4. Brumec J. (2008/2009). *Modeliranje procesa i aplikacija (Prezentacije sa Moodle sustava).* Dostupno 5.8.2012. na http://elfarchive.foi.hr/11_12/course/view.php?id=146
5. Sheldon R., Moes G. (2005). *Beginning MySQL.* Indianapolis: Wiley
6. Beighley L., Morrison M. (2009). *Head First PHP & MySQL.* Sebastopol: O'Reilly
7. Vaswani V. (2009). *A Beginner's Guide.* SAD: The McGraw-Hill Companies
8. Eriksson H.-E., Penker M., Lyons B., Fado D. (2004). *UML2 Toolkit.* Indianapolis: Wiley
9. Halpin T., Morgan T. (2008). *Information modelin and relational databases (second edition).* Morgan Kaufmann
10. Paunović V., Tomić S. (2006*). PHP (Priručnik uz seminar)*