Model baze podataka za potrebe web-dućana

Petričušić, Zvonimir

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Zvonimir Petričušić

MODEL BAZE PODATAKA ZA POTREBE WEB-DUĆANA

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Zvonimir Petričušić

Matični broj: 46253/17-R

Studij: Informacijski sustavi

MODEL BAZE PODATAKA ZA POTREBE WEB-DUĆANA

ZAVRŠNI RAD

Mentor/Mentorica:

Dr. sc. Bogdan Okreša Đurić

Varaždin, kolovoz 2020.

Zvonimir Petričušić

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

---

Sažetak

Tema završnog rada je „Model baze podatka za potrebe web-dućana“. U radu je obrađen teorijski pregled domene baze podataka, PostgreSQL sustav za upravljanje bazama podataka i značajke koje su odabrane za modeliranu bazu podataka. Uz teorijski pregled, završni rad obuhvaća i praktični dio koji se odnosi na modeliranje baze podataka za potrebe web-dućana. Opisane su relacije između tablica, potrebni pogledi kako bi funkcionirala aplikacija te funkcije. Također, kako bi pokazali rad baze podataka, praktični dio obuhvaća i popratnu aplikaciju. Aplikacija je izrađena pomoću HTML-a, CSS-a te PHP-a i phpPgAdmina kako bi rad s bazom podataka bio omogućen.

Ključne riječi: baza podataka; web shop; modeliranje baze podataka; PostgreSQL; aplikacija; PHP; phpPgAdmin; CSS; HTML.

Sadržaj

1. Uvod

U današnjem modernom društvu, baze podataka su postale esencijalni dio svakog poduzeća, obrta, banke itd. Svaki čovjek, koji je tehnološki osviješten, susreće se svakodnevno s aktivnostima koji uključuju bazu podataka. Najbolji primjer toga su web stranice. Svaka stranica koja je dinamična, odnosno sadržaj joj je prilagodljiv, promjenjiv te omogućava interakciju s korisnikom, koristi bazu podataka. Mala i srednja poduzeća su ovisna o bazi podataka radi boljeg upravljanja informacijama. Upravljanje ogromnom količinom podataka može biti zamorno pa čak i nemoguće obraditi na papiru. Lakši put nude baze podataka koje pomažu korisnicima da se izvrši više zadataka odjednom [1]. Samim time se smanjuju troškovi poduzeća, ali baze podataka daju i mnoštvo drugih prednosti. Tako na primjer, možemo pratiti podatke klijenata i na osnovu tih podataka stupiti u kontakt s njima te im poslati informacije o novim proizvodima, ponudama te uslugama.

Baze podataka, nažalost, imaju nekoliko nedostataka. Najizraženiji od njih je šteta ili gubitak podataka koji utječe na sve programe koji koriste tu bazu podataka. Zato je potrebno svakodnevno raditi sigurnosne kopije ako je riječ o poduzećima koji upravljaju s dragocjenim podacima, kao što su na primjer banke. Također, valja napomenuti kako su baze podataka složene, teške i vremenski prilično zahtjevne što se tiče njihovog dizajniranja [2].

Odabrana tema za završni rad glasi „Model baze podataka za potrebe web-dućana“. Razlog odabira ove teme je taj što u današnjem svijetu ,tj. na današnjem tržištu svako poduzeće koje prodaje ili preprodaje proizvod, nudi usluge ne može opstati bez korištenja web-dućana. Većina trgovina ili poduzeća se danas oslanja i na sami web-dućan bez prisustva fizičke trgovine.

Kako bi web-dućan funkcionirao, kako bi se pravilno skladištili podaci o proizvodima, klijentima, računima itd. potrebna je baza podataka. Zapravo, za svaku web stranicu ili aplikaciju, koja nije statična, potrebna je baza podataka. Dakle, potražnja za modeliranje baza podataka u današnje doba je velika. U radu je vidljivo koje su sve tablice te relacije potrebne kako bi baza podataka ispravno funkcionirala. Nadalje, vidljivo je koji su pogledi potrebni da bi aplikacija pravilno dohvaćala podatke iz baze podataka te koje su funkcije korištene, te ograničenja u samim tablicama. Također prikazan je i rad aplikacije kako bi bio ostvaren dojam funkcioniranja izrađenog web-dućana.

2. Opis aplikacijske domene

Na početku je potrebno obraditi teorijski dio baza podataka kako bi bilo razumljivo funkcioniranje izrađene baze podataka, te koji principi i načela moraju biti zadovoljeni. Obrađen je razvojni ciklus kojim se treba voditi prilikom izrade svake baze podataka. Kroz svaki korak potrebno je detaljno proći kako niti jedna komponenta ne bi bila izostavljena. U radu je bilo potrebno detaljno opisati svaku relaciju, vezu i atribut kako bi, uz pomoć relacijskog dijagrama (eng. "Entity relationship diagram", skraćeno "ERD"), bilo moguće shvatiti na koji način funkcionira baza podataka. Veliki dio rada posvećen je izradi aplikacije koja služi za lakše upravljanje bazom podataka. U radu je opisano kako koristiti stvorenu aplikaciju te kako njome navigirati.

U ovom završnom radu implementirana je baza podataka za poslovanje poduzeća koje se bavi preprodajom automobila marke „Mercedes“. Baza podataka je osmišljena tako da se, uz malo dorade, može prenamijeniti za sve ostale vrste proizvoda. Također, aplikacija je rađena tako da administrator može lagano upravljati s njome, odnosno omogućiti dodavanje novih automobila, brisanje istih, dodjeljivanje akcijskih cijena proizvodima te mnoge druge funkcionalnosti koje su potrebne za uspješno vođenje web-dućana. . Aplikacija je izrađena kao web aplikacija pomoću HTML-a, CSS-a te PHP-a. Treba istaknuti programski jezik PHP, koji u savršenoj kombinaciji s HTML-om i CSS-om, omogućuje lagan rad s bazom podataka i jednostavan prikaz i upravljanje informacijama. Tablice, odnosno relacije u bazi podataka su kreirane u minimalnom broju kako bi bilo omogućeno uspješno upravljane i korištenje aplikacije.

3. Popis i opis korištenih alata

3.1. Lucidchart

Lucidchart je platforma koja se koristi kako bi korisnicima omogućila suradnju na crtanju, reviziji i dijeljenju grafikona i dijagrama. Lucidchart je iznimno jednostavan za upotrebu, tako da je idealan kako za poslovanje, tako i za edukaciju (osnovne škole, srednje škole i fakulteti).

Slika 1: Lucidchart logo (Izvor: **www.lucidchart.com**)

Kako bi baza podataka bila ispravno izrađena, te kako bi bile bolje predočene potrebne veze između pojedinih tablica i utvrđeni potrebni atributi i ograničenja koja trebaju biti korišteni najbolje je skicirati, osmisliti ER Dijagram. Pošto PostgreSQL ne nudi tu mogućnost, za kreiranje ER Dijagrama odabran je alat Lucidchart koji je dostupan na adresi www.lucidchart.com. Osim što je ovaj alat besplatan, nudi i najviše mogućnosti za postavljane veza te je takve veze najlakše pročitati. Također, navedeni alat daje predloške za lakše korištenje i kreiranje potrebnih dijagrama.

3.2. phpPgAdmin

PostgreSQL je besplatni sustav za upravljanje bazama podataka koji poštuje ACID principe pri izvođenju transakcija [3]. PhpPgAdmin nam služi za administraciju PostgreSQL poslužitelja baza podataka. On također može raditi sa svim funkcionalnostima kao što su kreiranje tablica, umetanje podataka, postavljanje ograničenja, itd. Također, koristi neke složenije funkcionalnosti kao što su okidači, pogledi i funkcije. Prednost ovog programa je grafičko sučelje koje olakšava rad s bazom podataka. Uz sve to, nudi mogućnost izvoza i uvoza podataka tj. može obraditi jezike kao npr. SQL, XML, XHTML i CSV.

Slika 2: phpPgAdmin logo (Izvor: **www.phppgadmin.sourceforge.net/doku.php**)

Odabran je za ovaj završni rad zbog njegove jednostavnosti spajanja na određenu web stranicu / aplikaciju. Nakon što su kreirane sve potrebne tablice, postavljena sva potrebna ograničenja, izrađene potrebne funkcije te okidači, njegova svrha postaje učitavanje i iščitavanje podataka, ali putem web aplikacije. Samo u krajnjim slučajevima bi se trebalo pristupati bazi podataka preko phpPgAdmina.

3.3. Atom

Za pisanje programskog koda za kreiranje web aplikacije korišten je uređivač teksta pod nazivom „ATOM“. Podržan je na macOS-u, Linuxu te Windows platformama. Atom podržava različite programske jezike i formate datoteka među kojima su HTML, CSS, PHP te SQL koji su bili potrebni za izradu aplikacije. U navedenom uređivaču lako je održavati navigaciju među datotekama te je jako pregledan i lako se koristi. Ima više korisnih funkcija. Neke od njih su: podjela zaslona kako bi usporedili kod, mogućnost pronalaska ključne riječi te trenutačne zamjene.

Slika 3: phpPgAdmin logo (Izvor: **www.atom.io**/)

4. Općenito o bazama podataka

Baza podataka je skup povezanih, organiziranih podataka te korisnici (u pravilu) bazu podataka doživljavaju kao skup (povezanih) tablica [4, str. 1]. Iz ove tvrdnje se može zaključiti kako svaka baza podataka sadrži tablice u koje skladištimo i organiziramo podatke. Te tablice trebaju biti povezane na osnovu primarnih i vanjskih ključeva kako bi bila omogućena razmjena podataka između tablica. Cilj je mogućnost korištenja povezanih podataka od strane različitih aplikacija. Bazu podataka također se može opisati kao skladište u koje se spremaju željeni podaci. Kasnije, po potrebi tim podacima se pristupa putem računala odnosno sustava za upravljanje bazom podataka. Baze podataka su strukturirane tako da olakšavaju pohranu, pronalaženje, modificiranje i brisanje podataka zajedno s različitim operacijama obrade podataka [5]. Sustavi za upravljanje bazom podataka izvlače informacije iz baze podataka kao odgovor na upite. Potrebno je napomenuti da je baza podataka skup informacija koji je potrebno organizirati kako bi se omogućio čim lakši pristup [4, str. 1]. Bazom podataka se upravlja dodavanjem, brisanjem ili ažuriranjem podataka. Idealne su za poslovanje nekog poduzeća koje raspolaže s velikom količinom podataka te olakšavaju posao upravljanja tim istim podacima. Baze podataka imaju razne prednosti, a neke od njih su:[2]

- Smanjena suvišnost podataka
- Integritet podataka
- Poboljšan pristup podacima
- Povećana sigurnost podataka
- Smanjeni troškovi unosa, pohrane i pretraživanja podataka

Treba napomenuti da baze podataka imaju i svojih nedostataka, kao što su na primjer troškovi izrade hardvera i softvera. Također, obuka i upoznavanje programera i korisnika s bazom podataka je prilično dugotrajan proces [2].

Razlikuje se više modela baza podataka : [6, str. 4]

- Relacijski
- Hijerarhijski
- Mrežni
- Objektni

4.1. Relacijski model baza podataka

U ovom radu je korišten relacijski model baze podataka. Razlog je taj što gotovo svi sustavi za upravljanjem bazama podataka podržavaju isključivo njega [6, str. 4]. Mrežni i

hijerarhijski modeli najviše su se koristili 60-tih i 70-tih godina 20. stoljeća. Od 80-tih godina pa sve do današnjih dana prevladava relacijski model dok se prijelaz na objektni model još uvijek nije dogodio [6, str. 4]. Relacijski model zasnovan je na relacijama koje su međusobno definirane vezama. Svaka relacija, odnosno tablica sadrži više atributa iz kojih se uzimaju vrijednosti. Svaka relacija mora imati primarni ključ pomoću kojeg identificiramo svakog od ostalih članova relacije. Također, kako bi bile ostvarene veze među relacijama, relacije moraju sadržavati potrebne vanjske ključeve. Prilikom toga potrebno je paziti na referencijalni integritet kako bi veze među relacijama bile važeće [7]. Ovo je izrazito važno radi ukrštenih upita između dvije tablice. Ako tablice nisu pravilno povezane nije moguće dohvatiti potrebne podatke. Kako bi se pristupilo podacima u relacijskoj bazi podataka potrebno je koristiti jezik SQL. Nazivi tablice i nazivi stupaca korisni su za tumačenje značenja vrijednosti u svakom retku te su podaci unutar njega predstavljeni kao skup odnosa [8].

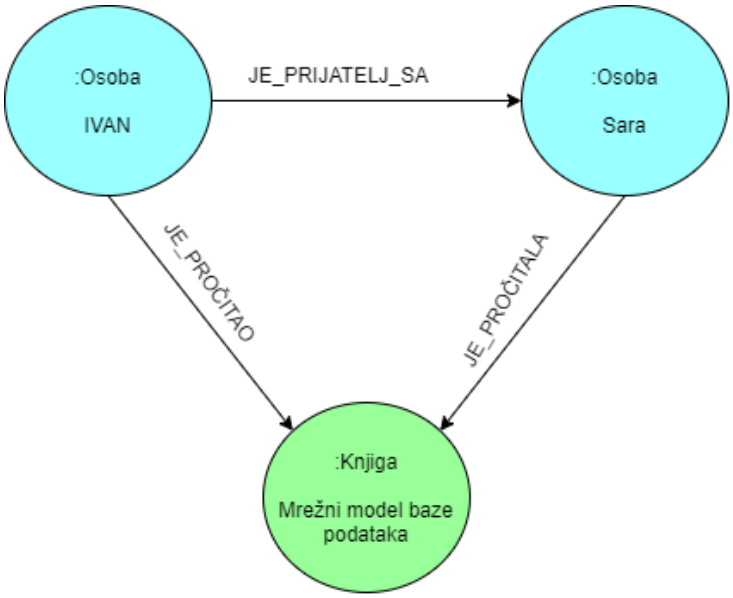
4.1.1. Prednosti korištenja relacijskog modela

Relacijska baza podataka sastoji se od prikladno raspoređenih tablica iz kojih se podacima može upravljati na različite načine, bez potrebe za preuređivanjem cijelog niza tablica baze podataka [9]. Najveća prednost relacijske baze podataka je ta što omogućuje korisniku klasificiranje podataka prema različitim kategorijama. To dalje omogućava korisnicima jednostavan pristup podacima putem upita. Može se, bez ikakvih poteškoća, dohvatiti željena tablica. Također, ukoliko su ispravno povezane, tablice možemo kombinirati pomoću upita za pridruživanje i uvjetnih izraza. Kod ostalih modela baza podataka ovo nije moguće već se isto vrši navigacijom kroz stablo ili putem hijerarhijskog modela [9]. Relacijska baza podataka bavi se samo podacima, a ne i strukturom. To izrazito poboljšava izvedbu modela [8]. Treba napomenuti kako se relacijska baza podataka sastoji od redova i stupaca koje je lako razumijeti odnosno interpretirati njihovo značenje pomoću njihovog imena. Važnu ulogu kod relacijske baze podataka predstavlja integritet podataka koji osigurava ostale karakteristike baze podataka kao što je: jednostavnost upotrebe, preciznost i stabilnost podataka [9]. Relacijska baza podataka smanjuje redundanciju podataka te olakšava implementirati sigurnosne metode, odnosno omogućava kreiranje tablica kao povjerljive, sa kontrolom razine pristupa. To znači da im mogu pristupiti samo određeni korisnici koji imaju dozvolu za pristup.

4.2. Mrežni model baza podataka

Kod mrežnog modela baze podataka, baza je predočena mrežom te se ona sastoji od čvorova ili usmjerenih lukova [6, str. 4]. Čvorovi predstavljaju tipove zapisa, a lukovi veze među tipovima zapisa [6, str. 4]. Podaci su organizirani poput grafa te smiju imati više od jednog roditeljskog čvora [10]. Samim time podaci su više povezani što omogućava lakši i brži pristup. Veza koja se koristi za ovaj model baze podataka je M:N, odnosno više prema više [10]. Samim time teško ga je implementirati i održavati. Danas se koristi pod nazivom graf baze koji prikazuje podatke u obliku grafikona. Sastoji se od entiteta i njihovih odnosa, odnosno oni se ne mapiraju u tablice već korisnici mogu izravno obrađivati podatke prema entitetima i odnosima [11]. To

predstavlja veliku prednost prilikom modeliranja jer se lakše može zaključiti koje relacije su međusobno povezane. Na slici ispod možemo vidjeti primjer povezivanja modela grafa baze.



Slika 4: Graf baze podataka (Izvor: **https://neo4j.com/developer/guide-data-modeling/**)

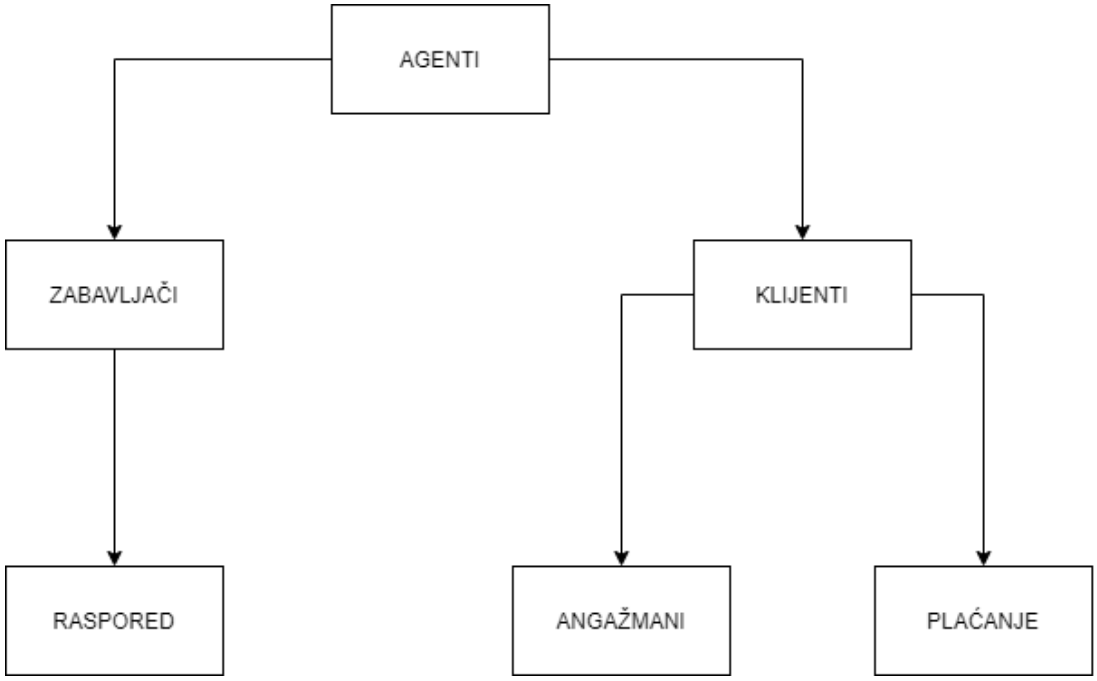
4.3. Objektni model baza podataka

U ovom su modelu scenariji predstavljeni kao objekti koji su grupirani zajedno i povezani sa različitim objektima. Tim objektima se dodjeljuju svojstva i vrijednosti, a podaci se strukturiraju na način kako bi bili svestraniji od jednostavnog popisa [12]. Objektni model podataka pomaže u ispunjavanju većih ciljeva oko velikih podataka i analitike. Objektni model podataka predstavlja podatke koji su sastavljeni od modula. Pomoću njih se kombiniraju podaci i postupci koji rade sa podacima [12]. Objektni model podataka ima raznih prednosti, kao što su svojstva nasljeđivanja zbog kojih možemo ponovno koristiti atribute i funkcionalnosti. Ako zatreba neka nova značajka, može se dodati nova klasa, koja je naslijeđena od roditeljske klase, te joj dodati nove značajke [12]. Ova svojstva pomažu da baza podataka objektnog modela bude izrazito fleksibilna u slučaju bilo kakvih promjena. Treba napomenuti kako ovaj model nije u potpunosti razvijen te nije cjelovit za upotrebu u sustavima baza podataka. Također, objektni model baze podataka treba zamisliti kao pristup rješavanja problemu, a ne kao tehnologiju [12]. Na osnovu toga nije ga moguće implementirati u sustav za upravljanje bazama podataka.

4.4. Hijerarhijski model baza podataka

Kao što mu i samo ime govori, hijerhijski model baze podataka organizira podatke u hijerarhiju. Hijerarhija počinje sa korijenskim podatkom te se dalje širi poput stabla, dodajući podređene čvorove nadređenim čvorovima [10]. Iz toga je vidljivo da je veza koja se koristi je 1:M, odnosno jedan naprema više. To predstavlja problem ukoliko nam je potrebna veza M:N, odnosno više prema više. Hijerarhijski model nije fleksibilan, jer dodavanje novih odnosa

može rezultirati velikim promjenama postojeće strukture, što znači da se moraju mijenjati i sve postojeće aplikacije [10]. Dijagram u nastavku prikazuje tipičnu hijerarhijsku strukturu baze podataka.



Slika 5: Hijerahijski model (Izvor: **https://www.tutorialspoint.com/Hierarchical-Database-Model/**)

4.5. PostgreSQL

SQL je jezik koji omogućuje rad s bazama podataka, odnosno SQL je lingua franca u svijetu baza podataka [4, str. 1]. Pomoću ovog jezika moguće je kreiranje tablica, funkcija, okidača, vršenje upita, unosa i mnogih drugih funkcionalnosti. Neki od poznatijih sustava za upravljanje bazom podataka jesu: MySQL, SQL Server, PostgreSQL, Oracle... Za izradu ove baze podataka korišten je PostgreSQL. Razlog tome je što PostgreSQL nudi program zvan phpPgAdmin koji služi kao uslužni program za PostgreSQL bazu podataka. Omogućuje povezivanje web aplikacije s bazom podataka, što znači da ima mogućnost unosa, brisanja ili ažuriranja podataka u tablicama putem web stranice, također još jedan od razloga zašto je odabran PostgreSQL je taj što je jako jednostavan za korištenje te je idealan za početnike koji se tek upoznaju s radom baze podataka.

PostgreSQL ima visoku reputaciju zbog njegove arhitekture, pouzdanosti, integriteta podataka, proširivošću. PostgreSQL je dostupan na svim operacijskim sustavima te posjeduje ACID osobine [3]. Također valja napomenuti kako je PostgreSQL besplatan alat koji je vrlo proširiv.

5. Razvojni ciklus baze podataka

Uvođenje baze podataka u neku ustanovu predstavlja složeni zadatak koji zahtijeva primjenu pogodnih metoda i alata te timski rad stručnjaka raznih profila [6, str. 8] . Sami razvojni ciklus baze podataka je podijeljen u pet aktivnosti, a to su:[6, str. 8-11]

- Utvrđivanje i analiza zahtjeva
- Projektiranje
- Implementacija
- Testiranje
- Održavanje

Odluka za izradu baze podataka se uvijek veže za potrebu nekog poduzeća da bilježi i prati podatke koji su njemu bitni. Važno je ne usredotočiti se na neposredni zadatak, nego na razumijevanje podatka koji služe za podržavanje tog zadatka i druge zadatke koji će eventualno koristiti bazu podatka [13, str. 9]. Prema tome, važno je proći svaki od razvojnih ciklusa baze podataka, od utvrđivanja i analize zahtjeva do samog održavanja.

5.1. Utvrđivanje i analiza zahtjeva

Prilikom izrade svake baze podataka prva stvar je prepoznati zahtjeve, podatke, tokove informacija poduzeća za koje se razvija baza podataka te ustanoviti koje su veze među njima. Ovo je jako bitan korak, te mu je važno posvetiti vrijeme kako bi se zaključilo na koji način je najjednostavnije razviti bazu podataka i tako izbjeći redundanciju i nekonzistentnost [6, str. 8-9]. Najlakši način za uvrđivanje i analizu zahtjeva poduzeća je intervju s osobljem, korisnicima te proučavanje trenutačnog softvera. Važno je stvoriti početnu ideju baze podataka te osmisliti kako ju uskladiti s aplikacijom. Najlakši način za navedeno je korištenje dijagrama. U procesu je važno ustanoviti kako će korisnici komunicirati sa sustavom. Nakon što se jasno postave ciljevi sustava potrebno je oblikovati model kako bi se dobila šira slika problema. Prilikom izrade ove baze podataka, proveden je intervju s poduzećem „Magis tim d.o.o.“ koje se bavi online prodajom naljepnica, ukrasnih jastučića, šalica itd. te je prepoznat kao idealan primjer onoga što bi sve jedan web-dućan trebao imati (kategorije, akcijske cijene, online kupovinu, pregled računa, sučelje za administratora i moderatora ...). Također, napravljen je pregled velikog broja web-dućana koji se bave prodajom novih i rabljenih automobila kako bi se ustanovilo koji su atributi potrebni i time bile dostupne sve potrebne informacije za buduće korisnike.

5.2. Projektiranje

Nakon što su utvrđeni i analizirani zahtjevi poduzeća, baza podataka mora biti oblikovana prema zadanoj specifikaciji. Dakle, valja prepoznati koji su entiteti, atributi potrebni kako

bi se kasnije omogućilo ispravno povezivanje tablica te pružanje svih potrebnih informacija budućem korisniku. Projektiranje je složena aktivnost, koja se obično dijeli na tri faze koje slijede jedna iza druge. A to su:[6, str. 9]

- Projektiranje na konceptualnoj razini
- Projektiranje na logičkoj razini
- Projektiranje na fizičkoj razini

5.2.1. Projektiranje na konceptualnoj razini

Prva faza projektiranja je sastavljena od entiteta, veza i atributa, te joj je glavni cilj stvoriti konceptualnu shemu baze podataka. Svrha konceptualne sheme nije izrada baze podataka. Razlog je taj što joj nedostaje previše podataka i veza kako bi se isto ostvarilo. Njena svrha je da predoči svakome korisniku, buduću bazu podataka te kako bi ona trebala funkcionirati.

5.2.2. Projektiranje na logičkoj razini

Glavni rezultat druge faze projektiranja je logička shema, koja je u slučaju relacijskog modela sastavljena od relacija odnosno tablica [6, str. 9]. U toj fazi predočena je povezanost tablica baze podataka. Kao najbolji primjer toga može poslužiti ER dijagram baze podataka za potrebe web-dućana. Vidljivo je koje su pojedinačne tablice povezane te koje su veze korištene među njima.

5.2.3. Projektiranje na fizičkoj razini

Fizičku shemu baze podataka treba shvatiti kao programski kod koji služi za izradu tablica, pogleda, upita, funkcija. Dakle, ostvaruju se putem SQL naredbi te omogućuju rad s bazom podataka.

5.3. Implementacija

Implementacija podrazumijeva pokretanje SQL naredbi. Samim time stvaraju se potrebne tablice i veze među njima te se ostvaruje rad baze podataka. Također korisnicima treba omogućiti sučelje, obrasce putem kojeg će unositi tražene podatke. Za izradu sučelja treba koristiti forme ili web stranicu. Nakon što je ostvarena veza s bazom podataka, podatke koje upisuje korisnik unose se izravno u tablice, ovisno o kojim podacima je riječ. Također vrlo je važno omogućiti izvještaje baze podataka koji će prikazivati željene podatke. To se vrši putem upita. Ako je baza podataka ispravno povezana omogućen je pristup svim potrebnim podatcima na osnovu veza između ključeva tablica. Bazu podataka je potrebno ispuniti podacima koji će se kasnije koristiti za testiranje. Važno je napomenuti kako se u ovoj fazi ne unose pravi podatci već samo testni podaci. Tablice se popunjavaju s obzirom na njihove primarne i vanjske ključeve te na osnovu veza između njih.

5.4. Testiranje

Prilikom testiranja korisnici rade s kreiranom bazom podataka te isprobavaju sve njezine funkcionalnosti. Kod testiranja je važno praćenje svih prepoznatih grešaka u radu sustava te prijavljivanje istih timu odgovornom za njihovo ispravljanje. Ovo je često dugotrajan proces jer praksa nalaže ponvaljanje testiranja iznova ako se uoči greška. Pogreške u prijašnjim aktivnostima imaju teže posljedice jer se provlače i kroz sljedeće aktivnosti pa samim time zahtijevaju više truda da se poprave [6, str. 10]. Zato je važno detaljno pregledati svaku navedenu aktivnost da li je došlo do propusta. Prilikom testiranja važno je pratiti i bilježiti performanse sustava te ako nisu zadovoljavajuće potrebno je poboljšati iste. Iznimno je bitna i zadovoljavajuća brzina učitavanja, te privlačan dizajn i sadržaj. Potrebno je također obratiti pažnju da navigacija bude jasna i nedvosmislena, odnosno da bude shvatljiva za svakog korisnika.

6. Specifikacija zahtjeva

Kako bi se ostvarilo što bolje poslovanje i produktivnost poduzeća putem web-dućana, treba utvrditi i analizirati zahtjeve poduzeća za kojeg se web-dućan izrađuje. Ostvaren je kontakt s više poduzeća koji se bave online prodajom proizvoda, te kao idealan primjer uzeto je poduzeće „Magis Tim d.o.o.“ koje se bavi proizvodnjom i online prodajom naljepnica, ukrasnih jastučića i šalica. Putem kratkog intervjua definirane su sve komponente koje web-dućan mora sadržavati te su ustanovljeni sljedeći elementi.

6.1. Kategorije proizvoda

Kako automobili mogu pripadati više kategorija i više modela bilo je potrebno implementirati ovu komponentu kako bi se klijentima omogućilo lakše snalaženje i navigaciju na web stranici odnosno web-dućanu. Stranica je podijeljena na četiri kategorije: SUV, limuzine, karavani te compact automobili. Svaka od kategorija posjeduje više vrsta modela, te jedan model može pripadati više kategorija.

6.2. Prijava i registracija

Važna stavka svakog web-dućana je bilježenje korisnika koji koriste njegove usluge. Dakle, kako bi se ostvarila mogućnost da korisnik prati svoje račune, kupuje proizvode te postavlja upite i ostavlja recenzije na određene proizvode potrebno ih je bilježiti u bazu podataka, a to je najlakše ostvariti putem prijave i registracije.

6.3. Kontakt

Kako bi korisnik mogao biti u stalnom kontaktu s prodavačem, neovisno o tome dali je neregistriran ili registriran potrebno je implementirati stranicu na kojoj će korisnik postavljati upite. Prilikom toga korisnik upisuje i email adresu na koju će moderator ili administrator odgovoriti u što kraćem vremenskom roku.

6.4. Računi

Korisnik, ako izvršava kupnju automobila mora na pregled imati i račun koji mora platiti kako bi njegov proizvod mogao biti dostavljen. Korisnik također može vidjeti da li je njegova uplata potvrđena od strane administratora.

6.5. Sučelje za administratora ili moderatora

Sučelje koje nije vidljivo običnome korisniku, već samo moderatoru ili administratoru. Putem navedenog sučelja ovlaštenici imaju mogućnost upravljanja cijelom bazom podataka ili web stranicom. Tu se podrazumijeva: promjena naslovne slike, dodjela akcijskih cijena proizvoda, postavljanje novog proizvoda, kreiranje nove kategorije itd.

Uz sve komponente koje su nabrojane bilo je potrebno proučiti web-dućane koji se također bave prodajom vozila te na osnovu toga izvući najbitnije elemente koji su potrebni za opis proizvoda. Pri tome su neizbježni sljedeći elementi: cijena, vrsta motora, model, prijeđeni kilometri, opis, slika te eventualno akcijska cijena.

7. Popis i opis tablica, atributa i veza

7.1. Tablica "kategorija"

Tablica 1: Tablica kategorija

kategorija	
PK	kategorija_id: SERIAL
	naziv: CHARACTER(45), NOT NULL

Kao što je već opisano u poglavlju „Opis aplikacijske domene“, ova baza podataka služi za prodaju odnosno preprodaju rabljenih automobila marke „Mercedes“. Svrha tablice *"kategorija"* je da odredi kojoj kategoriji pripada određeni model kako bi uspješno bio ispisan na web stranici na ispravnom mjestu. Točnije rečeno, web stranica je osmišljena tako da se npr. limuzine ispisuju na posebnoj stranici, karavani također na posebnoj stranici itd. Zbog toga je bilo potrebno dodati tablicu *„kategorija“* koja se sastoji od primarnog ključa *„kategorija_id“* , tipa serial te atribut *„naziv“*, tipa CHARACHTER(45) koji ne smije imati praznu vrijednost. Svaka kategorija mora imati naziv. Tablica *„kategorija“* nema vanjski ključ na nijednu drugu tablicu već se na nju veže tablica *„model“*.

7.2. Tablica model

Tablica 2: Tablica model

model	
PK	model_id: SERIAL
	naziv: CHARACTER(45), NOT NULL
FK	id_kategorija: INTEGER, NOT NULL

Tablica *„model“*, slično i kao tablica *„kategorija“* se sastoji od primarnog ključa, *„model_id“*, tipa serial te atributa *„naziv“* tipa character(45) čija vrijednost ne smije biti prazna. Tablica *„model“* također sadrži vanjski ključ *"id_kategorija"* i označava vezu 1:M, odnosno kategorija može imati više modela, a model može pripadati samo jednoj kategoriji. Tablica je namijenjena tako da se ostvari još jedna podjela vozila kako bi se omogućilo lakše pretraživanje korisniku.

7.3. Tablica "proizvodi"

Primarni ključ tablice je *„proizvod_id“* tipa serial. Atributi koje sadrži tablica *„proizvodi“*, a valja ih istaknuti jesu dva vanjska ključa, *„id_model“* koji se povezuje na primarni ključ tablice *„model“* te označava da model može imati više proizvoda, dok proizvod može pripadati samo jednom modelu. Isti način povezivanja je korišten i s tablicom „akcija“ samo preko vanjskog

Tablica 3: Tablica proizvodi

proizvodi	
PK	proizvod_id: SERIAL
	cijena: NUMERIC, NOT NULL
	opis: TEXT, NOT NULL
	vrsta_motora: CHARACTER(45), NOT NULL
	slikanaziv: CHARACTER(45), NOT NULL
	kilometri: INTEGER, NOT NULL
FK	id_model: INTEGER, NOT NULL
	godiste: INTEGER, NOT NULL
	snagamotora: CHARACTER(45), NOT NULL
FK	id_akcija: INTEGER

ključa *"id_akcija"*. Ostali atributi tablice „proizvod“ označavaju tj. opisuju sami proizvod te su neizostavni podaci o istome. Oni se mogu saznati putem istraživanja različitih web-dućana koji se bave prodajom novih i rabljenih automobila te su itekako potrebni kako bi klijentu dali cijelu sliku proizvoda koji želi kupiti.

7.4. Tablica "korisnik"

Primarni ključ ove tablice je *„korisnik_id“* koji je tipa serial. Tablica *„korisnik“* sadrži sve podatke koji su potrebni kako bi se upravljalo računom, kupovinom i ostalim mogućnostima korisnika. Prilikom registracije korisnik mora ispuniti sva polja osim *„email_potvrda“* i *„blokiran“* koja se naknadno popunjavaju kada korisnik potvrdi email ili zablokira korisnički račun. Nakon što korisnik uspješno unese podatke u bazu podataka putem forme, stupac *„email_potvrda“* je netočna vrijednost. Sve dok je tako registracija nije valjana, odnosno korisniku preostaje još jedan korak, a to je potvrda email adrese koju može ostvariti putem linka koji je poslan na njegovu email adresu. Ovo je potrebno izvršiti da se provjeri da li klijent uistinu posjeduje navedenu email adresu. Stupac *„blokiran“* je također netočna vrijednost. Sve dok je tako korisnik ima mogućnost prijave. Korisnik zablokira korisnički račun ako unese krivu lozinku više od tri puta ili ako ga administrator blokira. Također implementiran je stupac koji ima početnu vrijednost koja, u tablici „uloga“ označava ulogu registriranog korisnika. Valja spomenuti stupce *„korisnickoime“* i *„email“* koje moraju biti jedinstvene vrijednosti. Razlog je taj što se preko korisničkog imena saznaju ostali podatci o korisniku koji su potrebni. Zapravo korisničko ime funkcionira kao jedna vrsta identifikacije. Potrebno je da email bude jedinstven kako ne bi došlo do beskonačnog kreiranja računa istog korisnika s istim emailom. Eventualno, email služi još da se klijentu pošalje odgovor na upite ili npr. katalog cijena.

7.5. Tablica "upit"

Primarni ključ ove tablice je *„upit_id“* koji je tipa serial. Ova tablica služi kako bi se omogućilo praćenje upita korisnika kojeg pošalju putem stranice odnosno forme. Valja istaknuti

Tablica 4: Tablica korisnik

korisnik	
PK	korisnik_id: SERIAL
	ime: CHARACTER(45), NOT NULL
	prezime: CHARACTER(45), NOT NULL
	email: CHARACTER(45), NOT NULL, UNIQUE
	telefon: CHARACTER(45), NOT NULL
	korisnickoime: CHARACTER(45), NOT NULL, UNIQUE
	lozinka: CHARACTER(45), NOT NULL
	adresa: CHARACTER(45), NOT NULL
	postanskibroj: CHARACTER(45), NOT NULL
	mjesto: CHARACTER(45), NOT NULL
	email_potvrda: BOOLEAN
	blokiran: BOOLEAN
	id_uloga: INTEGER, NOT NULL

Tablica 5: Tablica upit

upit	
PK	upit_id: SERIAL
	naslov: CHARACTER(45), NOT NULL
	upit: TEXT, NOT NULL
FK	id_korisnik: INTEGER, NOT NULL

vanjski ključ *„id_korisnik“* koji se referencira na primarni ključ tablice *„korisnik“*. On služi kako bi znali koji korisnik je postavio upit te eventualno saznali njegov email.

7.6. Tablica "akcija"

Tablica 6: Tablica akcija

akcija	
PK	akcija_id: SERIAL
	iznos: INTEGER, NOT NULL
	pocetak: DATE, NOT NULL
	kraj: DATE, NOT NULL

Primarni ključ ove tablice je *„akcija_id“* tipa serial. Tablica *„akcija“* sadrži stupce:

- - *„iznos“*, koji je tipa integer, kako bi bio definiran popust na određeni proizvod. Na primjer, 30 označava popust od 30%.
- - *„pocetak“*, tipa DATE, koji označava vremenski početak akcije.
- - *„kraj“*, također tipa DATE, koji označava vremenski kraj akcije.

Tablica *"akcija"* služi za prikazivanje proizvoda koji su pod akcijskom cijenom. To omogućuje veza između navedene tablice i tablice *„proizvod“*. Ti proizvodi se prikazuju samo na početnoj stranici te je omogućeno izračunavanje nove cijene putem PHP-a

7.7. Tablica "dnevnik"

Tablica 7: Tablica dnevnik

dnevnik	
PK	dnevnik_id: SERIAL
FK	radnja_id: INTEGER, NOT NULL
	datumvrijeme: TIMESTAMP, NOT NULL
FK	korisnik_id: INTEGER, NOT NULL

Primarni ključ tablice *„dnevnik“* je *„dnevnik_id“* koji je tipa serial. Tablica *„dnevnik“* služi kako bi se bilježila svaka radnja koju napravi registrirani korisnik, moderator ili administrator. Tome služi vanjski ključ *„korisnik_id“* koji je referenciran na primarni ključ tablice *„korisnik“*. Također, implementiran je još jedan vanjski ključ, *„radnja_id“* kako bi se znalo o kojoj je radnji riječ, odnosno da li je ta radnja uređivanje, brisanje, umetanje... Uz to, također je implementiran i stupac *„datumvrijeme“* kojemu je početna vrijednost trenutačni datum i vrijeme. Sva polja ove tablice se automatski popunjavaju. ID dnevnika se automatski popunjava, dok se ID radnje popunjava na osnovu radnje koja se odvija. Ako je radnja umetanja sustav će prepoznati tu radnju u tablici *„radnja“* te odabrati njeni ID i tako popuniti stupac *„radnja_id“*. Datum i vrijeme se također automatski popunjava zahvaljujući ugrađenoj PHP funkciji *"CURRENT_TIMESTAMP()"*. Ovisno o kojem korisniku je riječ, sustav će preuzeti njegov ID te ga umetnuti u odgovarajući stupac.

7.8. Tablica "radnja"

Tablica 8: Tablica radnja

radnja	
PK	radnja_id: SERIAL
	naziv: CHARACTER(45), NOT NULL

Tablica koja se sastoji od dva stupca, primarni ključ *"radnja_id“* te stupca *„naziv“* koji služi kako bi označili o kojoj je radnji riječ. Tablica uvelike olakšava posao prilikom popunjavanja tablice *„dnevnik“* jer su sve moguće radnje unaprijed definirane. Na osnovu korisnikove radnje može se saznati i radnja iz navedene tablice te se ona bilježi u dnevnik, automatski.

Tablica 9: Tablica racun

racun	
PK	racun_id: SERIAL
FK	id_proizvod: INTEGER, NOT NULL
FK	id_korisnik: INTEGER, NOT NULL
FK	id_placanje: INTEGER, NOT NULL
	cijena: NUMERIC, NOT NULL
	PDV: INTEGER, NOT NULL
	ukupna_cijena: NUMERIC, NOT NULL
	placeno: BOOLEAN
	potvrdenoplacanje: BOOLEAN
	isporuceno: BOOLEAN
	datumstavaranja: DATE, NOT NULL

7.9. Tablica "racun"

Primarni ključ tablice je *„racun_id“* koji je tipa serial. Tablica ima tri vanjska ključa *„id_proizvod“*, *„id_korisnik“*, *„id_placanje“* koji se vežu redom na tablice *„proizvod“*, *„korisnik“* te na tablicu *„placanje“*. Svrha im je kako bi saznali o kojem je proizvodu, korisniku odnosno kupcu riječ, te koji način plaćanja koristi trenutačni korisnik. Tablica također ima stupce *„potvrdenoplacanje“* te *„isporuceno“* koje mora popuniti administrator, odnosno označiti kao točnu vrijednost kada je proizvod isporučen kupcu te kada je vidljiva uplata. Tablica ispisuje podatke o plaćenim i neplaćenim računima određenog kupca na stranici *„Moji računi“*. Preko te stranice korisnik vrši i plaćanje te može vidjeti da li je njegova kupnja potvrđena. Potrebno je istaknuti stupac *„PDV“*. Za njegov izračun napravljena je posebna funkcija. Korišteni PDV je 25%. Nakon što je izračunat PDV, ta vrijednost se pridruživa cijeni i tako se upotpunjava stupac *„ukupna_cijena“*.

7.10. Tablica "placanje"

Tablica 10: Tablica placanje

placanje	
PK	placanje_id: SERIAL
	naziv: CHARACTER(45), NOT NULL

Tablica koja se sastoji od dva stupca, primarni ključ *„placanje_id“* te stupca *„naziv“* koji služi kako bi se utvrdilo o kojem je plaćanju riječ. Sustav podrazumijeva dvije vrste plaćanja, plaćanje pouzećem te plaćanje karticom. Korisniku se nudi mogućnost odabira između te dvije opcije te se na osnovu toga odabira saznaje ID plaćanja i upisuje se u tablicu *"racun"*. Na primarni ključ tablice veže se vanjski ključ tablice *„racun“* i tako se ostvaruje veza među te dvije tablice koja je 1:M, odnosno svaka vrsta plaćanja može pripadati više računa, dok račun može imati samo jednu vrstu plaćanja.

7.11. Tablica "osvrt"

Tablica 11: Tablica osvrt

osvrt	
PK	osvrt_id: SERIAL
FK	id_proizvod: INTEGER, NOT NULL
	ocjena: INTEGER, NOT NULL
FK	id_korisnik: INTEGER, NOT NULL

Primarni ključ tablice je *„osvrt_id“* tipa serial. Tablica također ima dva vanjska ključa koji se referenciraju na primarni ključ tablice *„proizvodi“* i *„korisnik“*, a služe kako bi se omogućio pregled koji korisnik je kojem proizvodu dao koju ocjenu. Veze su tipa 1:M odnosno proizvod može imati više osvrta od strane više korisnika, dok osvrt pripada samo jednom korisniku i proizvodu. Za ovu aplikaciju je izrađena posebna HTML forma pomoću koje korisnik unosi ocjenu za odabrani proizvod. Jako je bitno pratiti zadovoljstvo klijenata kako bi se omogućilo pružanje najboljih usluga i proizvoda. Zato je potrebna tablica *„osvrt“*.

7.12. Tablica "uloga"

Tablica 12: Tablica uloga

placanje	
PK	uloga_id: SERIAL
	naziv: CHARACTER(45), NOT NULL

Tablica koja se sastoji od primarnog ključa *"uloga_id"* tipa SERIAL te stupca *"naziv"* koja i ujedno pobliže opisuje ulogu koju korisnik može posjedovati. Na sustavu dostupne su 4 uloge:

- Neregistrirani korisnik
- Registrirani korisnik
- Moderator
- Administrator

Svaka od uloga ima svoje funkcije, mogućnosti i zabrane. Na tablicu se veže tablica *"korisnik"* putem vanjskog ključa vezom 1:M. Uloga kod tablice *"uloga"* najviše služi mogućnosti prijave, odnosno da bi se utvrdilo o kojemu je korisniku riječ te koje mogućnosti on sve posjeduje na sustavu.

8. ER dijagram

Pošto phpPgAdmin nema opciju izrade ER dijagrama, ER dijagram je izrađen u online alatu za izradu dijagrama pod naziv Lucidchart. Izrada ER dijagram je temelj kreiranja svake baze podataka. Na ovom ER dijagramu je vidljivo 12 tablica, svaka sa svojim atributima. Neke tablice su složenije neke jednostavnije, ovisno o njihovoj ulozi. Pretežno je korištena veza 1:M odnosno jedan naprema više te nije bilo potrebe za korištenjem pomoćnih tablica. Iste te tablice, sa zadanim ograničenjima, atributima i vezama implementirane su i u phpPgAdminu, tj. kreirana je baza podataka na osnovu ovog ER dijagrama. Nešto više o samim tablicama, njihovim atributima i vezama izneseno je u idućem poglavlju.

Slika 6: ER dijagram

9. Implementacija

Nakon što je kreiran ER dijagram te je baza podataka i logički povezana potrebno je krenuti s njenom implementacijom. Kao što je već rečeno, baza podataka se sastoji od 12 tablica koje je bilo potrebno kreirati putem SQL naredbi. Potrebno je obratiti pozornost na redoslijed kreiranja tablica odnosno treba imati na umu kako su tablice povezane (nije moguće povezati vanjski ključ na primarni ako taj primarni ključ ne postoji).

9.1. Kreiranje tablica

Prvo je izrađena tablica *„kategorija“* pošto ne sadrži niti jedan vanjski ključ. To se izvodi putem sljedeće SQL naredbe.

```
CREATE TABLE kategorija (kategorija_id SERIAL PRIMARY KEY, naziv CHARACTER(45));
```

Nadalje, kreirana je tablica *„model“* koja sadrži vanjski ključ koji se povezuje na tablicu *„kategorija“*.

```
CREATE TABLE model (model_id SERIAL PRIMARY KEY, naziv CHARACTER(45), id_kategorija
INTEGER REFERENCES kategorija(kategorija_id));
```

Kako bi bilo omogućeno kreiranje tablice *„proizvodi“* treba kreirati još jednu tablicu pod nazivom *„akcija“*. Razlog je taj što tablica *„proizvod“* sadrži dva vanjska ključa od kojih je jedan *„id_akcija“* koji se povezuje na primarni ključ tablice *„akcija“*.

```
CREATE TABLE akcija (akcija_id SERIAL PRIMARY KEY, iznos INTEGER NOT NULL, pocetak
    DATE
NOT NULL, kraj DATE NOT NULL);
```

Na redu je tablica *„proizvodi“*. Kao što je već prije spomenuto, tablica se sastoji od dva vanjska ključa, primarnog ključa te dosta atributa. SQL naredba za kreiranje te tablice glasi:

```
CREATE TABLE proizvodi
(proizvod_id SERIAL PRIMARY KEY, cijena NUMERIC NOT NULL, opis TEXT NOT NULL,
vrsta_motora CHARACTER (45) NOT NULL, slikanaziv CHARACTER(45) NOT NULL,
kilometri INTEGER NOT NULL, id_model INTEGER NOT NULL REFERENCES model(model_id),
id_akcija INTEGER REFERENCES akcija(akcija_id),
vrstamotora CHARACTER(45) NOT NULL, godiste INTEGER NOT NULL,
snagamotora CHARACTER(45) NOT NULL);
```

Na gore navedenim primjerima vidljivo je da je potrebno prvo kreirati tablice koje sadrže primarne ključeve na koje se povezuju druge tablice putem vanjskih ključeva. Preostalo je kreirati još tri takve tablice, a to su *„placanje“*, *„uloga“*, *„radnja“*. SQL naredbe za kreiranje tih tablica glase:

```
CREATE TABLE uloga (uloga_id SERIAL PRIMARY KEY, naziv CHARACTER(45) NOT NULL);

CREATE TABLE  placanje (placanje_id SERIAL PRIMARY KEY, naziv CHARACTER(45) NOT NULL
    );
```

```
CREATE TABLE  radnja (radnja_id SERIAL PRIMARY KEY, naziv CHARACTER(45) NOT NULL);
```

Za kompletnu bazu podataka preostaje izraditi još pet tablica, *„racun“*, *„korisnik“*, *„dnevnik“* , *„osvrt“* i *upit*. No, i ovdje valja obratiti pozornost na redoslijed kreiranja tablica. Iz ER dijagrama vidljivo je da tablica *„dnevnik“* sadrži vanjski ključ na tablicu *„korisnik“*. Stoga je prvo potrebno kreirati tablicu *„korisnik“*. SQL naredbe za kreiranje ovih tablica glase :

```
CREATE TABLE  korisnik (korisnik_id SERIAL PRIMARY KEY, ime CHARACTER (45) NOT NULL,
prezime CHARACTER (45) NOT NULL, email CHARACTER (45) NOT NULL UNIQUE,
telefon CHARACTER (45) NOT NULL, korisnickoime CHARACTER (45) NOT NULL UNIQUE,
lozinka CHARACTER (45) NOT NULL, adresa CHARACTER (45) NOT NULL,
postanskibroj CHARACTER (45) NOT NULL, mjesto CHARACTER (45) NOT NULL,
email_potvrda BOOLEAN, blokiran BOOLEAN,
id_uloga INTEGER NOT NULL REFERENCES uloga(uloga_id) );

CREATE TABLE racun (racun_id SERIAL PRIMARY KEY,
id_proizvod INTEGER REFERENCES proizvod(proizvod_id),
id_korisnik INTEGER REFERENCES korisnik(korisnik_id),
id_placanje INTEGER REFERENCES placanje(placanje_id),
cijena NUMERIC NOT NULL, PDV INTEGER NOT NULL, placeno BOOLEAN NOT NULL,
isporuceno BOOLEAN NOT NULL, datumstvaranja DATE NOT NULL);

CREATE TABLE dnevnik (dnevnik_id SERIAL PRIMARY KEY, radnja_id INTEGER NOT NULL,
datumvrijeme TIMESTAMP NOT NULL,
id_korisnik INTEGER NOT NULL REFERENCES korisnik(korisnik_id));

CREATE TABLE osvrt (osvrt_id SERIAL PRIMARY KEY,
id_proizvod INTEGER NOT NULL REFERENCES proizvodi(proizvod_id),
id_korisnik INTEGER NOT NULL REFERENCES korisnik(korisnik_id),
ocjena INTEGER NOT NULL);

CREATE TABLE upit (upit_id SERIAL PRIMARY KEY,
naslov CHARACTER(45) NOT NULL, upit TEXT NOT NULL,
id_korisnik INTEGER NOT NULL);
```

9.2. Unos podataka za potrebe testiranja

Za potrebe testa pripremljeni su podaci za svaku tablicu. Unos tih podataka također je izvršen preko SQL naredbe „INSERT INTO“. Primjer toga prikazan je putem unosa u sljedeće tablice: *„kategorija“*, *„model“*, *„proizvodi“*. Razlog što su odabrane ove tri tablice je što su međusobno povezane te se najbolje može prikazati redoslijed kojim se trebaju unositi podatci.

```
INSERT INTO kategorija (naziv) VALUES ('limuzina');
```

Može se primjetiti da se prvo treba odabrati naziv tablice i stupci u koje se umeću vrijednosti, a potom unijeti vrijednost. Valja napomenuti da ako se unose tekstualne vrijednosti tu istu vrijednost treba staviti pod navodnike. Brojčane vrijednosti ne treba stavljati pod navodnike.

```
INSERT INTO model(naziv,id_kategorija) VALUES ('S klasa', 1);
```

Unesen je model pod nazivom „S klasa“ koji pripada kategoriji s ID-om pod brojem 1 odnosno pripada kategoriji limuzina.

```
INSERT INTO proizvodi(cijena, opis, vrsta_motora, slikanaziv, kilometri,
id_model,godiste,snagamotora)
VALUES ( 100000, 'Jako povoljna cijena', 'benzin','sklasa.jpg', 100000, 1, 2000,'100
    kW');
```

9.3. Funkcije

U ovoj bazi podataka napravljene su tri funkcije radi same demonstracije. Sljedeća tri primjera su od velike koristi jer se lakšim putem može doći do željenog podataka. Najbolje je izdvojiti funkciju za izračun ukupne cijene koja uvelike koristi prilikom kupnje novog proizvoda gdje funkcija izračunava ukupnu cijenu na osnovu cijene i PDV-a. Također, potrebno je napomenuti kako funkcije "brojDostupnihVozila()" i "minKilometri()" služe kao jedna vrsta pretraživanja za potrebe administratora i krajnjeg korisnika. Za ovu aplikaciju nije potrebno kreirati mnoštvo funkcija putem SQL-a, jer se mogu ostvariti putem PHP-a ili JavaScripta prilikom izrade aplikacije.

9.3.1. Broj dostupnih vozila

```
create function brojDostupnihVozila()
returns int
language plpgsql
as
$$
declare
   brojac integer;
begin
   select count(*)
   into brojac
   from proizvodi
   where id_model=7;

   return brojac;
end;
$$;
```

Funkcija koja govori koliko je dostupnih vozila na lageru. Vodi se prema modelu vozila. Dakle, koliko je vozila zadanog modela na skladištu toliki će se broj prikazati korisniku. Kod ove funkcije nema nikakvih argumenata.

9.3.2. Izračun ukupne cijene

```
create function ukupnaCijena(proizvod_ide integer)
returns int
language plpgsql
as
$$

declare
```

```
   brojac  numeric;
begin


 select cijena+(cijena*0.25) into brojac from proizvodi where proizvod_id=
     proizvod_ide;

return brojac;
end;
$$;
```

Funkcija koja se koristi za izračunavanje ukupne cijene proizvoda, dakle cijena + PDV. PDV je računat 25%. Ova funkcija ima jedan argument, a to je ID proizvoda koje korisnik želi kupiti, te na osnovu njega se saznaje cijena i na kraju ukupna cijenu.

9.3.3. Najmanji broj prijeđenih kilometara

```
create or replace function minKilometri()
returns int
language plpgsql
as
$$

declare
   brojac integer;
begin
   select proizvod_id
   into brojac
   from proizvodi where kilometri =  ( select min(kilometri) from proizvodi );
   return brojac;
end;
$$
;
;
```

Funkcija koja daje automobil s najmanjim brojem prijeđenih kilometara. Služi kao jedna vrsta filtriranja za korisnika. Kod ove funkcije nema argumenata.

9.4. Upiti

9.4.1. Registracija

```
$unos =pg_query($con,"insert␣into␣korisnik(ime,prezime,email,telefon,
    korisnickoime,lozinkasha1,adresa,postanskibroj,mjesto)␣values␣('".$ime."
    ',␣'".$prezime."',␣'".$email."',␣'".$telefon."'␣,␣'".$korIme."','".sha1(
    $lozinkaSHA1)."',␣'".$adresa."'␣,␣'".$postBroj."','".$mjesto."')");
```

Upit koji služi za unos podataka o korisniku prilikom registracije. Podaci o korisniku se prikupljaju putem HTML forme, i onda se te vrijednosti pohranjuju u varijable koje se kasnije

koriste za unos u bazu podataka. Prije svega treba ostvariti vezu s bazom podataka. PHP podržava SQL upite te se izvršavaju putem ugrađene PHP funkcije pg_query() koji mora sadržavati vezu baze podataka te sami upit koji treba izvršiti. Neovisno radi li se o umetanju, ažuriranju ili brisanju uvijek koristimo pg_query().

9.4.2. Dodjela akcije

```
$update=pg_query($con,"UPDATE␣proizvodi␣SET␣id_akcija␣=␣".$upit['akcija_id'
    ]."␣WHERE␣proizvod_id=".$proizvod_id);
```

Upit koji služi za dodjeljivanje akcije određenom proizvodu. Kako bi se to postiglo treba znati o kojem je proizvodu i akciji riječ. Nakon toga se koristi SQL upit tipa UPDATE koji se koristi za ažuriranje te odabranom proizvodu dodjeljuje akcijsku cijenu. Preko HTML forme dostupni su svi popusti odnosno akcije koji se dodijeljuju proizvodima. Nakon što moderator ili administrator odabere proizvod i akciju vrši se ažuriranje tablice „proizvodi“ na onome proizvodu za koji je ID odabran te mu se dodjeljuje odabrani ID akcije.

9.4.3. Odabir kategorije automobila

```
$con=pg_connect("host=localhost␣dbname=webshop␣user=postgres␣password=182ozux33");
        $sql = "SELECT␣*␣from␣proizvodi␣join␣model␣as␣mo␣on␣id_model=mo.model_id␣
            join␣kategorija␣as␣k␣on␣mo.id_kategorija=k.kategorija_id␣where␣k.
            kategorija_id␣=␣4";
```

Dobar primjer upita gdje je potrebno koristiti pridruživanje odnosno spajanje podataka iz dodatne dvije tablice. Ovaj primjer dohvaća sve proizvode iz tablice proizvodi, koji su kategorije 4. Kako bi ovaj upit funkcionirao moralo je biti ostvareno povezivanje triju tablica, *„proizvodi“*, *„model“*, *„kategorija“*. Svrha ovog upita je da se ispišu svi podaci proizvoda na stranici kategorije kojoj on pripada. Dakle, ako je *„kategorija_id“* = 4 koja u tablici *"kategorija“* označava limuzine, gore navedeni upit će nam ispisati sve proizvode koje pripadaju toj kategoriji. Stranica je osmišljena tako da svaka kategorija ima svoju stranicu.

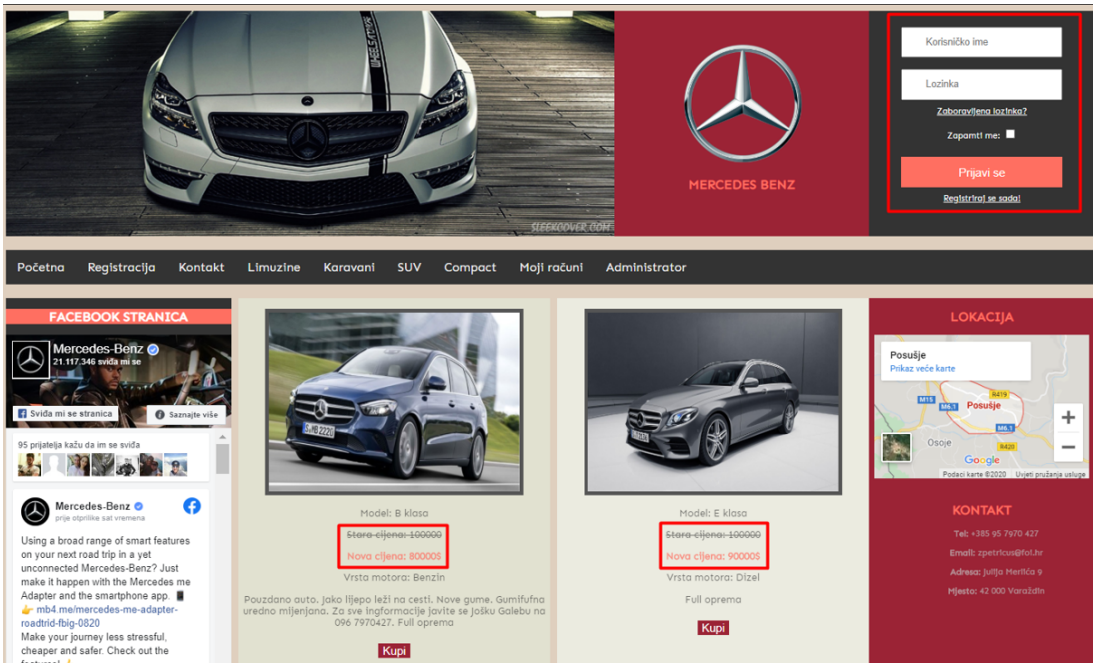
10. Izrada i korištenje aplikacije

Za izradu ove aplikacije korišten je program ATOM te sljedeći jezici:

- HTML
- CSS
- PHP
- JavaScript

Također korišten je i alat pod nazivom phpPgAdmin kako bi se ostvario rad s bazom podataka. Za potrebe aplikacije su korištene sve tablice iz baze podataka te je njihov rad ostvaren s velikim brojem php skripti. Web aplikacija funkcionira tako da koristi podatke koji se nalaze u bazi podataka te na osnovu njih prikazuje određene proizvode ili ostale podatke.

10.1. Početna stranica



Slika 7: Početna stranica

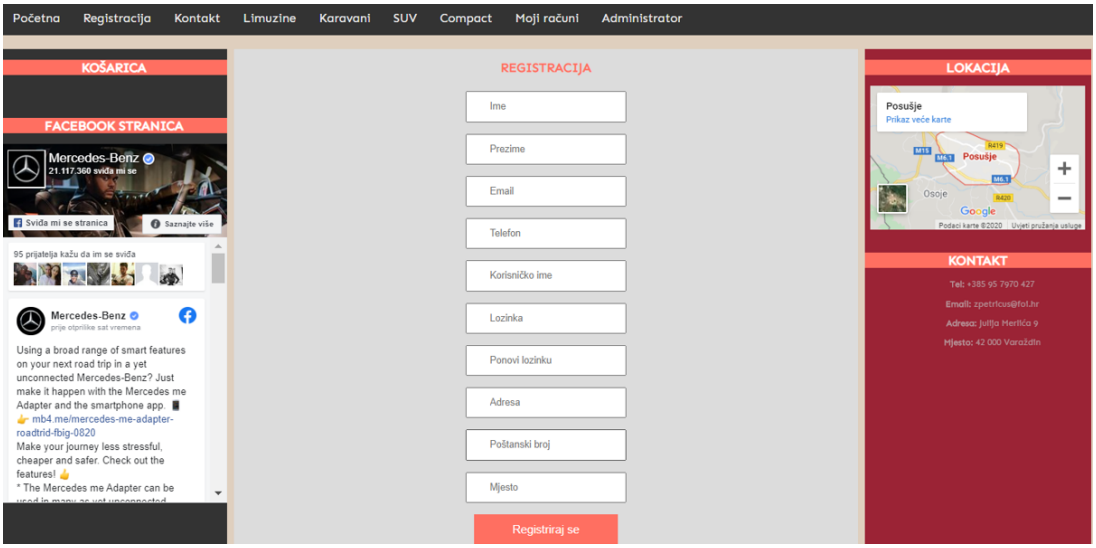
Na početnoj stranici se prikazuju samo oni proizvodi koji su pod akcijskom cijenom te je na osnovu te akcije odnosno popusta izračunata nova cijena. To je napravljeno pomoću idućeg upita.

```
$sql = "SELECT proizvod_id from proizvodi where id_akcija is not null";
```

Na početnoj stranici također se nudi mogućnost prijave te na osnovu nje sustav zaključuje da li je korisnik koji se prijavljuje moderator, administrator ili obični korisnik. Svaka vrsta

korisnika ima određene funkcije. Neregistrirani korisnik može pristupiti svim stranicama kao i registrirani korisnik, ali ne može vršiti kupnju te samim time nema pristup svojim računima. Administrator, kao najjača uloga, ima pristup svim skriptama web aplikacije. Web aplikacija je implementirana tako da administrator kroz korisničko sučelje može upravljati bazom podataka, blokirati korisnika, dodavati novi proizvod, model, kategoriju, kreirati akcijsku cijenu, dodijeliti akciju, promijeniti naslovnu sliku, dodijeliti ulogu administratora ili moderatora te ostale funkcionalnosti potrebne za uspješno funkcioniranje web-dućana.

10.2. Registracija



Slika 8: Registracija

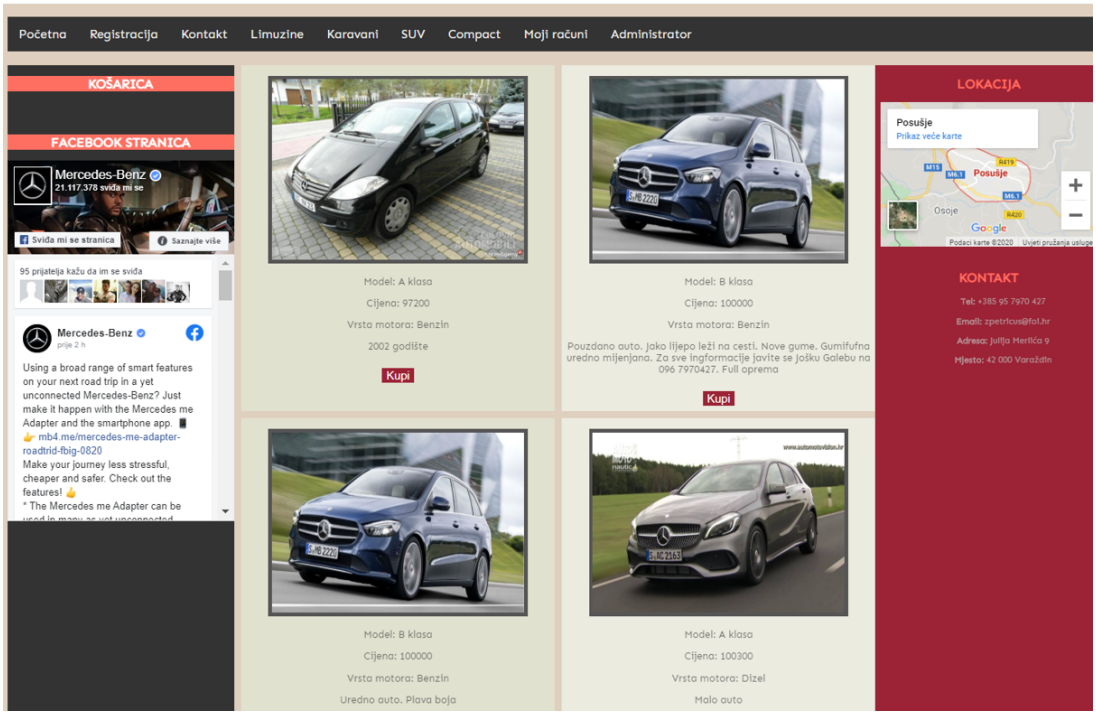
Dolaskom na stranicu registracija prikazuje se HTML forma koju korisnik ispunjava te samim time ostvaruje registraciju. Prilikom toga, automatski mu se dodjeljuje uloga vrijednosti 1, koja u tablicu uloga označava registriranog korisnika. Kako bi se uspješno izvršila registracija korisnik mora potvrditi email adresu i samim time će mu biti omogućena prijava s vlastitim podacima.

10.3. Kategorije

Automobili su podjeljeni na 4 kategorije:

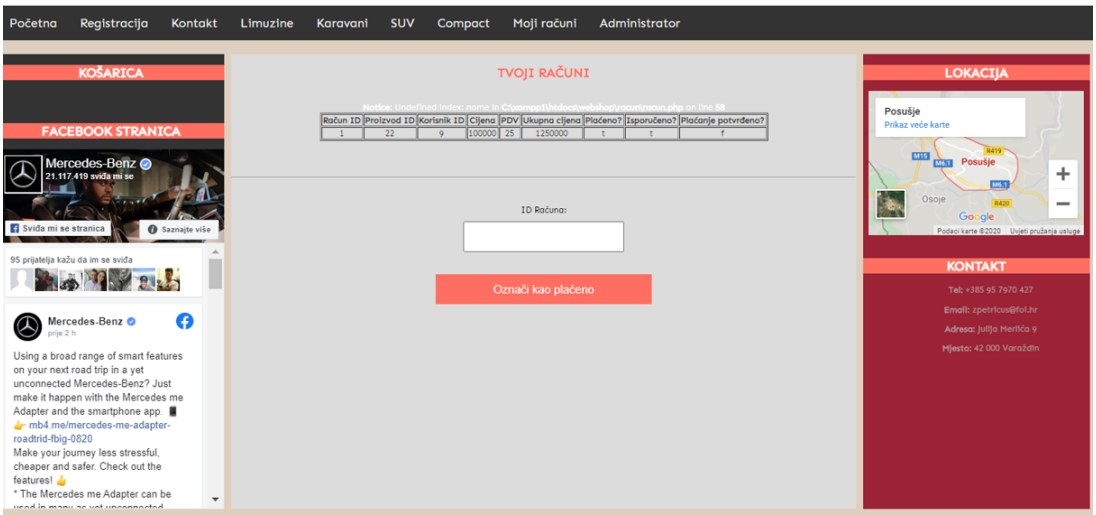
- Limuzine
- SUV
- Karavani
- Compact

Klikom na bilo koju od tih kategorija prikazuju se proizvodi, odnosno automobili koji pripadaju toj kategoriju. Tako naprimjer klikom na kategoriju Compact prikazuje se samo A klasa i B klasa.



Slika 9: Kategorije

10.4. Moji računi

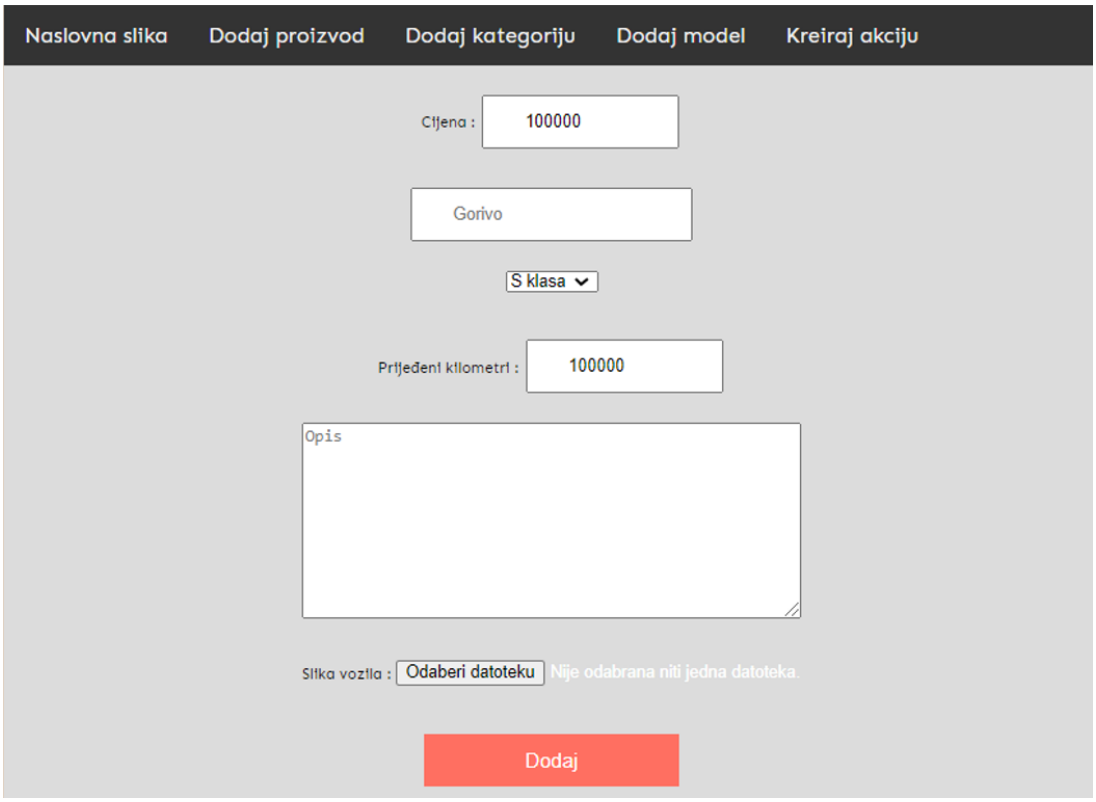


Slika 10: Moji računi

Prikazuju se računi trenutno prijavljenog korisnika koje može označiti kao plaćeno po id-u računa. Nakon što korisnik označi račun kao plaćen šalje se zahtjev administratoru da se uistinu potvrdi plaćanje. Korisniku je omogućeno više načina plaćanja koji se nalaze u tablici *"placanje"*.

10.5. Administrator

Dolaskom na stranicu administratora, nudi se izbornik pomoću kojeg se može upravljati podacima u bazi podataka. Toj stranici ima pristup samo administrator. Primjer u nastavku prikazuje dodavanje novog proizvoda.



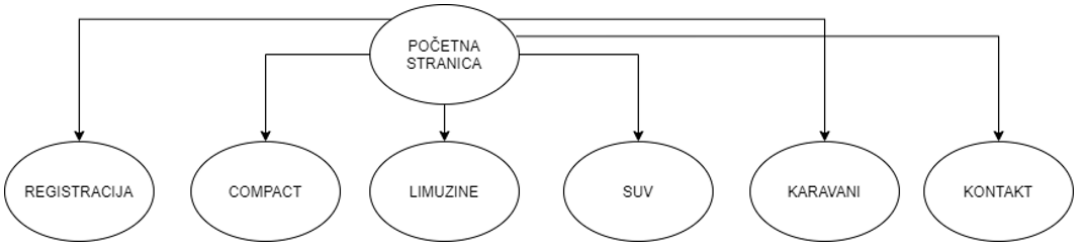
Slika 11: Sučelje za administratora ili moderatora

Nudi se HTML forma koju administrator popunjava s obzirom na funkcionalnosti i karakteristike novog proizvoda koji se dodaje te klikom na gumb "Dodaj", dodaje se novi proizvod.

11. Navigacijski dijagram

Sama svrha navigacijskog dijagrama je da objasni redoslijed i mogućnosti određenog korisnika pri kretanju kroz navigaciju web aplikacije. Na web aplikaciji podrazumijevaju se četiri vrste korisnika: neregistrirani, registrirani, moderator te administrator. Svaki od njih ima pristup određenim stranicama, a jača uloga od prethodne ima pristup svim stranicama prethodne uloge te još neke dodatne stranice. Tako će registrirani korisnik imati pristup svim stranicama neregistriranog korisnika, moderator će imati pristup stranicama registriranog i neregistriranog, a administrator ima pristup svim mogućim stranicama web aplikacije.

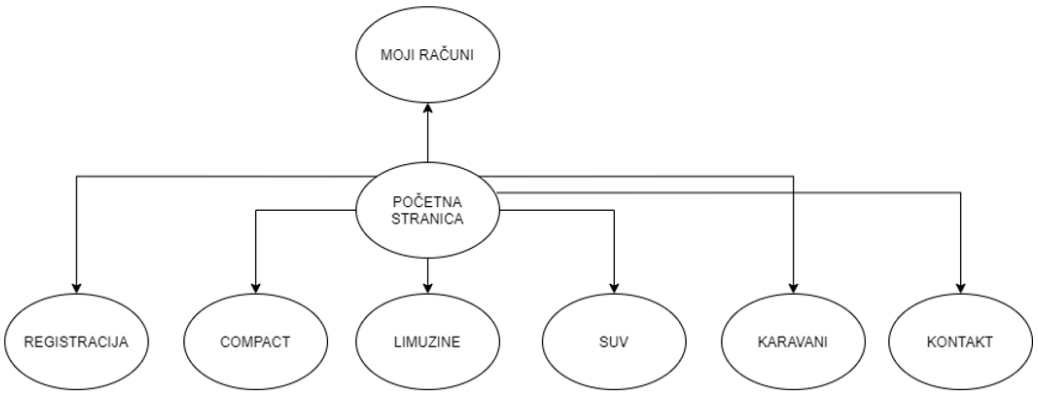
11.1. Neregistrirani korisnik



Slika 12: Navigacijski dijagram, Neregistrirani korisnik

Prva dostupna stranica neregistriranom korisniku je „Početna stranica“ te s nje može pristupiti sljedećim stranicama: „registracija“, „compact“, „limuzine“, „suv“, „karavani“ te „kontakt“. Važno je napomenuti da neregistrirani korisnik ne može izvršiti kupnju dok se ne prijavi.

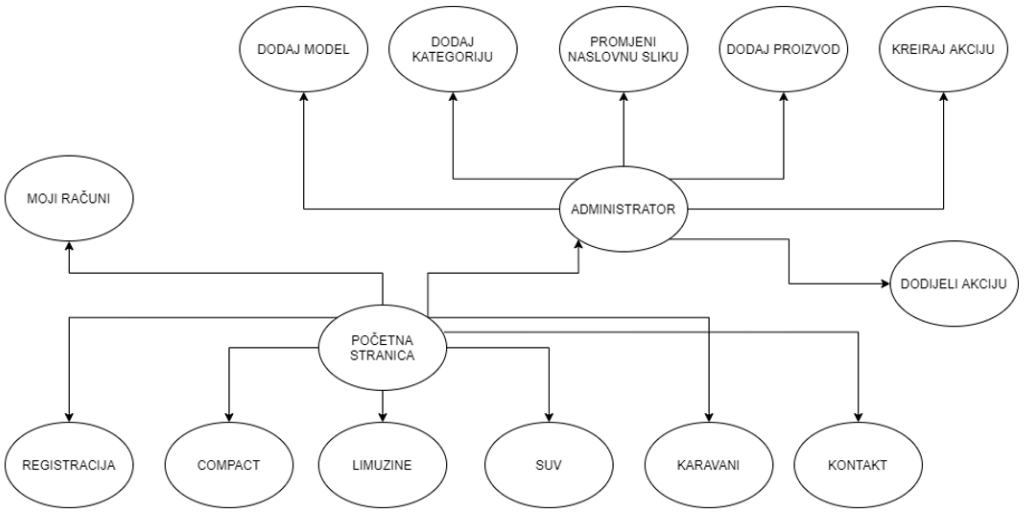
11.2. Registrirani korisnik



Slika 13: Navigacijski dijagram, Registrirani korisnik

Kao što je već objašnjeno, registrirani korisnik može pristupiti svim stranicama kao i neregistrirani korisnik, no uz sve to može pristupiti i stranici „moji računi“ što znači da registrirani korisnik ima mogućnost kupnje vozila te pregleda vlastitih računa koje može označavati kao plaćene.

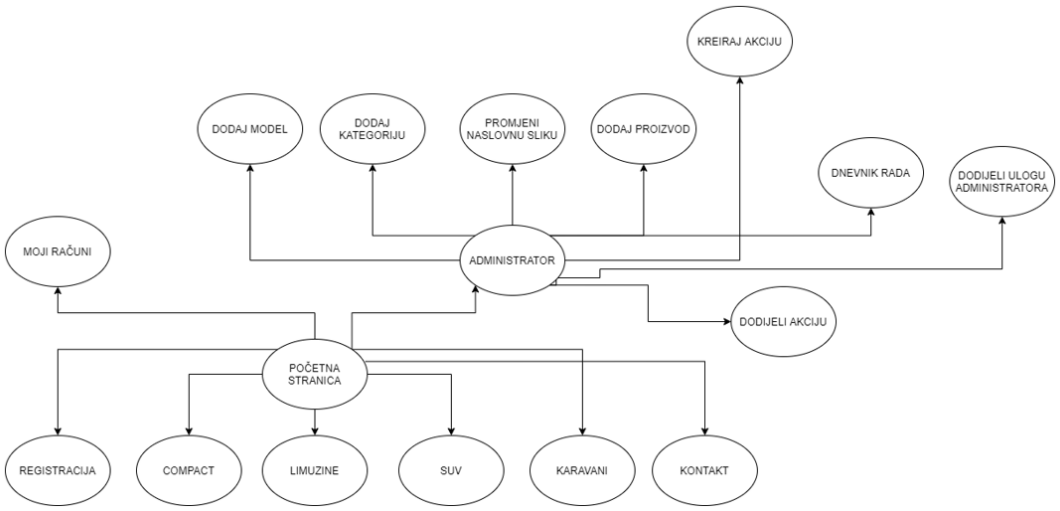
11.3. Moderator



Slika 14: Navigacijski dijagram, Moderator

Moderator ima pristup svim stranicama registriranog i neregistriranog korisnika, te dodatne opcije kojima mu je omogućeno upravljene aplikacijom i bazom podataka, kao naprimjer promjena naslovne slike, dodavanje novog proizvoda, modela, kategorije, dodjela akcije...

11.4. Administrator



Slika 15: Navigacijski dijagram, Administrator

Administrator ima pristup svim stranicama kao i moderator. Jedina je razlika ta što administrator ima pristup dnevniku rada te ima mogućnost dodjele uloge administratora i moderatora te oduzimanje navedenih uloga.

12. Zaključak

Baze podataka su iznimno korisne prilikom poslovanja i upravljanjem podacima nekog poduzeća te danas imaju veoma široku primjenu u mnogobrojnim poslovima. Razlog tome što je uz njihovo prisustvo omogućeno brže i isplativije izvođenje poslova. Također, povećana je sigurnost i točnost podataka.

U ovome radu prikazan je primjer modela baze podataka za potrebe web-dućana. Baza podataka je modelirana uz alat phpPgAdmin odnosno uz PostgreSQL. Korisničko sučelje je izrađeno pomoću HTML-a, CSS-a, JavaScripta, a sami upiti i povezivanje korisničkog sučelja s bazom podataka ostvareno je uz PHP. Putem SQL-a kreirano je nekoliko funkcija kako bi se demonstrirao njihov rad.

Na ovom primjeru je jasno objašnjeno da baza podataka i popratna aplikacija teško funkcioniraju jedno bez drugoga. Na primjer, puno je teže unositi podatke putem SQL-a, a i bez baze podataka ima li bi statičnu aplikaciju koja nema mogućnost dinamike. Poželjno je, ali nije nužno izraditi korisničko sučelje za vođenje baze podataka. Danas postoje razni sustavi za upravljanjem bazom podataka. Za PostgreSQL to je PgAdmin ili phpPgAdmin te oni omogućuju lakši unos podataka, brisanje ili ažuriranje. No, ako želimo da svaki korisnik može na neki način koristiti bazu podataka i donekle manipulirati podacima koji su u njoj skladišteni potrebno je izraditi korisničko sučelje. Bez baze podataka ne bi mogli ostvariti mogućnost prijave, registracije, kupnje proizvoda, izdavanje računa odnosno, svih stavki koje naša web stranica sadrži. Iz sveg navedenog može se zaključiti kako je poželjno za svako poduzeće, koje nudi usluge ili prodaje proizvode, omogućiti svojim klijentima ili kupcima korištenje web-dućana.

Popis literature

[1] *Baze Podataka*. adresa: https://www.znanje.org/abc/tutorials/accessMMX/01/Baze_podataka.htm (pogledano 9. 9. 2020).

[2] *What Are Advantages and Disadvantages of Using a Database?*, en. adresa: https://www.reference.com/world-view/advantages-disadvantages-using-database-4277419c4c49f63 (pogledano 9. 9. 2020).

[3] *PostgreSQL: The world's most advanced open source database*. adresa: https://www.postgresql.org/ (pogledano 9. 9. 2020).

[4] K. Rabuzin, *Uvod u SQL*. FOI, 2011, ISBN: 9789536071357.

[5] *database | Definition, Types, & Facts*, en. adresa: https://www.britannica.com/technology/database (pogledano 10. 9. 2020).

[6] M. Robert, *Osnove projektiranja baza podataka*, serija Ur. S. Rako. D310-20141203. HR: Sveučilište u Zagrebu, Sveučilišni računski centar: Tečajevi srca D310. Zagreb, 2010, ISBN: 978-953-7138-49-3.

[7] *REFERENCIJALNI INTEGRITET*. adresa: https://www.znanje.org/knjige/computer/access/access_01/referencijalni_integritet%20.htm (pogledano 9. 9. 2020).

[8] *Relational Data Model in DBMS: Concepts, Constraints, Example*. adresa: https://www.guru99.com/relational-data-model-dbms.html (pogledano 17. 9. 2020).

[9] *Relational Database Advantages | 8 Advantages of Relational Database*, en-US, veljača 2020. adresa: https://www.educba.com/relational-database-advantages/ (pogledano 17. 9. 2020).

[10] *Database Models in DBMS | Studytonight*. adresa: https://www.studytonight.com/dbms/database-model.php (pogledano 17. 9. 2020).

[11] *What is the graph database? What is data model?*, en-US, listopad 2016. adresa: https://bitnine.net/blog-graph-database/what-is-the-graph-database/ (pogledano 17. 9. 2020).

[12] *What is an Object Data Model? - Definition from Techopedia*, en. adresa: http://www.techopedia.com/definition/30736/object-data-model (pogledano 17. 9. 2020).

[13] C. Churcher, *Beginning database design: from novice to professional*, Second edition. New York, NY: Apress, 2012, ISBN: 9781430242093.

Popis slika

Popis tablica