

UNIVERZITET SINGIDUNUM
FAKULTET ZA INFORMATIKU I RAČUNARSTVO

**APLIKACIJA ZA EVIDENCIJU ROBE U
MAGACINU**

- diplomski rad -

Mentor:

prof. dr *Aleksandar Jevremović*

Kandidat:

Marko Petrović

Beograd, 2016.

Sadržaj

1	Uvod	- 3 -
2	HTML	- 4 -
2.1	Semantička obeležja	- 5 -
2.2	Formulari	- 6 -
2.2.1	Tipovi podataka	- 7 -
2.2.2	Novi atributi	- 8 -
2.2.3	Validacija podataka na klijentskoj strani	- 8 -
2.3	Tabele	- 9 -
3	CSS	- 11 -
3.1	Prilagodljiv dizajn	- 11 -
4	JavaScript	- 13 -
4.1	Implementacija	- 13 -
5	AdminLTE	- 15 -
5.1	Implementacija	- 15 -
5.2	Raspored elemenata	- 16 -
5.3	Opcije rasporeda elemenata	- 16 -
6	PHP	- 19 -
6.1	Pripremljeni upiti	- 19 -
7	MariaDB	- 21 -
8	Aplikacija	- 22 -
8.1	Pregled funkcija administratora	- 22 -
8.2	Uspostavljanje konekcije ka bazi podataka	- 22 -
8.3	Prijava	- 23 -
8.4	Pregled stanja proizvoda u magacinu	- 24 -
8.5	Dodavanje proizvoda	- 25 -
8.6	Izmena proizvoda	- 26 -
8.7	Spisak ulaza proizvoda u magacin	- 27 -

8.8 Lokacija proizvoda u magacinu.....	- 28 -
8.9 Model baze podataka	- 29 -
9 Zaključna razmatranja	- 31 -
Literatura	- 32 -

1 Uvod

Tema ovog rada je razvoj veb aplikacije koja omogućava evidenciju robe u magacinu primenom *MVC* arhitekture softvera.

Glavni cilj razvoja veb aplikacije predstavlja omogućavanja administrativnom osoblju firme da vodi preciznu evidenciju o lokaciji robe u magacinskim prostorima firme. Pored metoda adresiranja i vođenja evidencije, aplikacija vodi evidenciju o vrsti robe, datumu njenog evidentiranja u sistem, kao i evidenciju koji korisnik aplikacije je evidentirao robu. Svakom komadu robe se dodeljuje poseban identifikacioni broj.

Relizacijom aplikacije obezbeđena je mogućnost evidentiranja robe u više magacina, od kojih svaki magacin ima definisane prostorije, police unutar prostorije, police sa svojim redovima, a redovi su podeljeni u sekcije. Svaka sekcija reda police prostorije magacina u firmi je jedinstveno adresirana radi kvalitetnije organizacije prostora za odlaganje robe i kasnijeg pronalaska robe.

Jedan od dodatnih ciljeva aplikacije je nadomeštanje nedostataka koje imaju druge aplikacije za evidenciju robe, prednostima kao što su:

- Mogućnost aplikacije da se koristi preko bilo kojeg uređaja
- Smanjenim resursima potrebnim za izvršavanje aplikacije
- Dostupnost aplikacije u bilo kom momentu

Okruženja koja su korišćena tokom razvoja aplikacije su *Netbeans* za programiranje i *Navicat* za modelovanje i izradu baze podataka.

Za realizaciju veb aplikacije za evidenciju robe koristio sam aktuelne veb tehnologije. U ovom radu predstaviću *HTML* jezik za struktuiranje veb stranica, i novine koje su došle sa aktuelnom verzijom *HTML5*. *CSS3* jezik za stilizovanje i pozicioniranje elemenata i stvaranje multifunkcionalne aplikacije. *AdminLTE* alat koji omogućava brz razvoj korisničkog interfejsa (engl. *Interface*). *JavaScript* jezik koji omogućava dinamičnost stranice i rasterećivanja servera na kojem se nalazi aplikacija uz pomoć mreže za isporuku sadržaja. *PHP* serverskog jezika koji je zadužen za logiku unutar aplikacije i *MariaDB* relacione baze podataka koja predstavlja zamenu *MySQL* relacione baze podataka.

2 HTML

HTML (Hyper Text Markup Language) predstavlja standardizovani jezik za pravljenje struktura veb (engl. *Web*) stranica. *HTML* opisuje strukturu veb stranica koristeći svoja obeležja (eng. *tag*). Pretraživači ne prikazuju nazive obeležja, oni ih obrađuju i stvaraju strukturu veb stranica.

Aktuelna verzija je *HTML5* koja predstavlja skraćenicu za niz povezanih i komplementarnih tehnologija, uključujući u sebe *CSS3*, *SCG*, *JavaScript APIs* itd.

Cilj je standardizacija uobičajenih problema i dizajn obrazaca koje su programeri koristili kroz godine i proširivanje osobina elemenata kako bi se zadovoljili zahtevi modernog veba, koji su u današnje vreme mnogo više usmerene ka razvoju aplikacija, nego kao u početku sa svojim nastankom za povezivanje dokumenata.

HTML5 sa svojim novim obeležjima čini pravljenje, održavanje i proširivanje veb stranica znatno lakšim.

Sledeći blok koda prikazuje početni šablon(eng. *Template*) veb stranice:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
  </body>
</html>
```

Od *HTML5* više nije potrebno da se pretraživaču striktno govori koji tip dokumenta je u pitanju, da li je to čist *HTML*, tranzicioni *HTML* ili *XHTML1.1*.

U *HTML5*, za deklaraciju dokumenta koristi se `<!DOCTYPE html>`.

Još jedna promena u odnosu na standardni *HTML* šablon je *meta* obeležje koji definiše opseg skupa karaktera koji će se koristiti za obradu teksta na stranici. U najvećem broju slučajeva na vebu se koristi *UTF-8* skup karaktera.

```
<head>
...
<meta charset="utf-8">
...
</head>
```

2.1 Semantička obeležja

HTML5 donosi nova obeležja za organizovanje strukture *HTML* dokumenta, koji olakšavaju pravljenje i izmenu dokumenta. Nova obeležja imaju više intuitivne nazive u odnosu na elemente iz prethodne *HTML* verzije. Obeležja imaju prikladnija imena za delove stranica nad kojim se primenjuju, kao što su `<header>`, `<section>` i `<footer>`. Davanje sadržaju ovakvo dodatno značenje donosi instant korist, pre svega pretraživačima je lakše da razumeju podatke.

U *HTML4.01* i prethodnim specifikacijama, programer je stvarajući strukturu *HTML* dokumenta koristio `<div>` obeležja tokom razvoja. Obeležje `<div>` često unutar sebe uključuje atribut klasu (eng. *class*) ili atribut *id*, preko kojeg bi u *CSS* na potreban način stilizovao ili pozicionirao element na željenu poziciju.

Atributi *id* i klasa su globalni atributi, što znači da mogu da se koriste sa bilo kojim *HTML* elementom. Za nazive navedenih atributa moguće je koristiti proizvoljan naziv, kao što su *header*, *nav*, *article*, *sidebar* itd.

Primer strukture iz *HTML 4.01*:



Novi *HTML5* elementi za pravljenje strukture i organizaciju sadržaja u *HTML* dokumentu, uključuju `<header>`, `<footer>`, `<section>`, `<nav>`, `<article>` i `<aside>`.

<article>	Definiše artikal, kao što se nalazi u magazinu ili novinski članak, objava na blogu, ili sličan sadržaj
<aside>	Definiše sadržaj koji je odvojen, ali vezan za sadržaj stranice. Slično marginama u odeljcima knjiga ili u novinskim člancima
<footer>	Definiše footer za dokument ili sekciju, može da uključi unutar sebe informacije o autoru, kontakt podatke, autorska prava, ili veze ka povezanim dokumentima
<header>	Definiše header za dokument ili sekciju, može da sadrži uvodni sadržaj ili navigacione linkove
<nav>	Definiše blok sa navigacionim linkovima
<section>	Definiše sekciju u dokumentu, kao što je poglavlje ili deo veb stranice čiji je sadržaj odvojen u odnosu na drugi

Primer strukture *HTML5* dokumenta sa semantičkim obeležjima:



2.2 Formulari

Formulari omogućavaju prikupljanje informacija od korisnika koji će biti unete na veb stranici ili u internoj aplikaciji i koje će kasnije biti korišćene za različite izveštaje.

HTML5 je uveo nekoliko novih elemenata za formu i attribute za polja za unos podataka, kao što je tip podatka *url* za unos adrese veb stranica, polja za unos elektronske adrese (engl. *email*) i polja za pretragu (engl. *search*).

Za kreiranje forme koristi se `<form>` obeležje. Sav sadržaj i polja forme idu između dva `<form>` obeležja. Većina formi uključuju unutar svog obeležja i atribut `id` kao u sledećem primeru:

```
<form id="magacin">
    <Sadržaj i polja za unos>
</form>
```

Element `<fieldset>` se koristi sa formama kako bi se grupisali srodni elementi, `<fieldset>` kreira okvir oko pojedinačnog elemente i/ili oko čitave forme.

2.2.1 Tipovi podataka

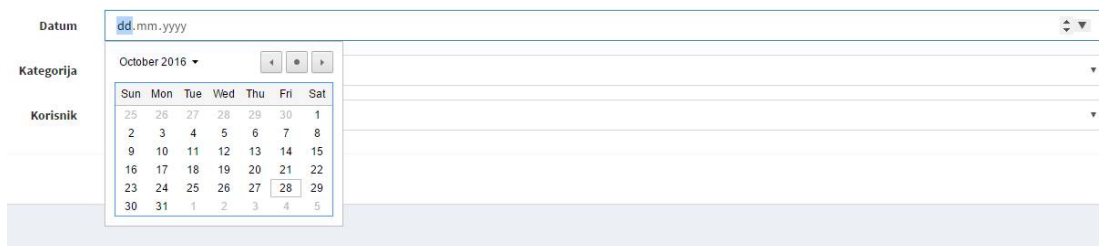
Dosta formi zahteva od korisnika unos brojeva, na primer, unos podataka kreditne kartice, poštanski broj ili količinu proizvoda. Tip podataka broj (engl. *Number*) je polje za unos numeričkih vrednosti. Često je prikazano kao tekstualno polje ali neki pretraživači dodaju i kontrolu, par strelica, jedna na gore, druga na dole za uvećavanje ili smanjivanje vrednosti.

```
<input type="number">
```

Ovaj tip podatka ima nove atribute za kontrolu unosa koji dolaze uz njega a to su `min` i `max` atributi, koji će definisati koje su maksimalne i minimalne brožane vrednosti za unos, i atribut `step` koji predstavlja broj po kojem će se raditi uvećavanje ili smanjivanje.

Još jedna vrsta podatka za unos unutar formulara je datumsko ili vremensko polje, koje se koristi u situacijama kada tražimo od korisnika da unese svoj datum rođenja ili da izabere vreme i datum kada želi da mu proizvod bude dostavljen. Najčešće je implementirano preko dodatka (eng. *Widget*).

```
<input type="date">
```



Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Slika 1: Prikaz dodatka za odabir datuma u pretraživaču

Ukoliko nam je potrebno da korisnik unese datum i vreme, taj zahtev ispunjava tip podatka `datetime`. Polje zahteva unos u formatu `YYYY-MM-DDTHH:MMZ` gde Z predstavlja UTC vremensku zonu.

2.2.2 Novi atributi

Novi atributi donose korisne i važne nove osobine i ponašanja za *HTML* forme. Bazirane su na popularnim *JavaScript* bibliotekama koje su postale sastavni deo *HTML5*.

Autofocus atribut uobičajan je za sve tipove unosa, stavlja fokus na određeni element prilikom učitavanja stranice. U polje za unos teksta kursor miša će biti postavljen i spreman za unos korisnika prilikom učitavanja stranice. *Autofocus* je *boolean* atribut, tako da vrednost za aktiviranje može biti sam naziv atributa, ili može biti izostavljen u potpunosti kao u primeru:

```
<input type="text" autofocus>
```

Ako se koristi više polja za unos sa atributom *autofocus* na stranici, prvi u *DOM* redosledu će se poštovati.

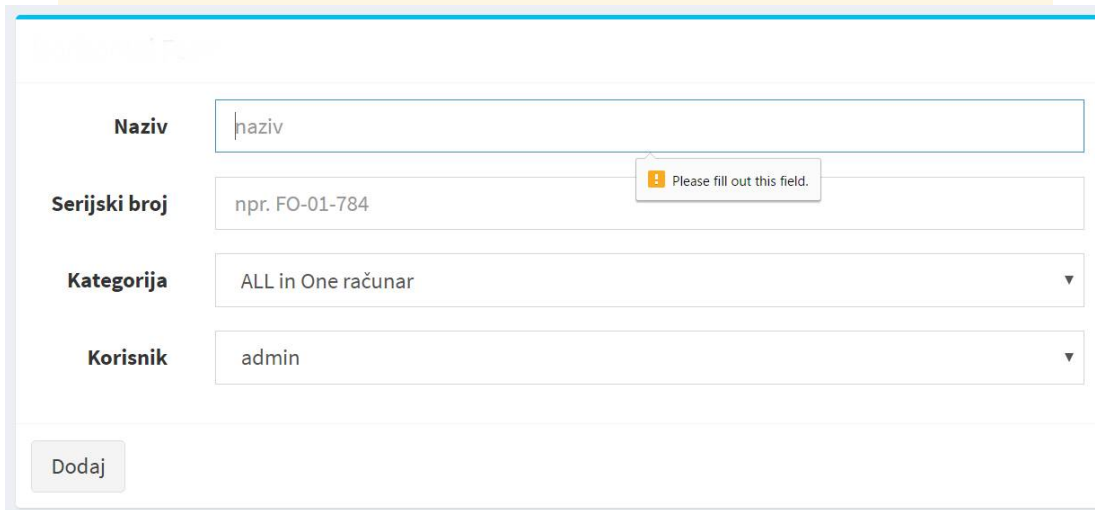
Placeholder atribut može da se koristi na bilo kojem polju za unos teksta, uzima *string* karaktera kao svoju vrednost. Ovaj *string* biće prikazan unutar polja za unos pre nego što se krene sa unosom podataka. Dobro je koristiti ovaj atribut, da bi se korisniku prikazalo kakav unos se očekuje od njega. Na primer u polju za unos elektronske adrese možemo da uključimo opis koji bi izgledao na sledeći način:

```
<input type="email" placeholder="npr email@gmail.com">
```

2.2.3 Validacija podataka na klijentskoj strani

Ukoliko želimo da učinimo da se neko polje u formi ne može poslati ne popunjeno, nego da zahteva uvek unos, može se koristiti atribut *required*. Ovaj atribut će naterati pretraživač da proveri polje unosa da li je prazno. Ukoliko je to tačno, da uputi korisnika da popuni polje pre potvrde forme.

```
<input type="text" required>
```



The screenshot shows a web form with the following fields and elements:

- Naziv**: A text input field containing the value "naziv".
- Serijski broj**: A text input field containing the value "npr. FO-01-784".
- Kategorija**: A dropdown menu with the selected value "ALL in One računar".
- Korisnik**: A dropdown menu with the selected value "admin".
- Dodaj**: A button located at the bottom left of the form.
- Validation Message**: A small yellow box with a red exclamation mark icon and the text "Please fill out this field." is positioned above the "Serijski broj" field, indicating that this field is required.

Slika 2: Prikaz poruke koja će upozoriti korisnika da je polje obavezno za unos

Novi tipovi podataka za unos su korisni, ali najverovatnije najveći poklon za programera predstavlja mogućnost provere podataka na klijentskoj strani. Provera sadržaja forme pre slanja na server i upisa u bazu je veoma važno zbog bezbednosti i kvaliteta podataka koji se unosi. Do sada, nije postojao jednostavan način da se proverí, iako je mnogo *JavaScript* biblioteka napravljeno kako bi vršilo takve provere.

Sada, većina pretraživača će automatski da upozori da podatak koji se unese ne odgovara tipu podatka za taj unos. U *Firefox* pretraživaču npr, ako se unesu samo brojevi u *email* polje za unos ili ukoliko se izostavi *http://* na početku *URL* adrese u *url* polju za unos, pojaviće se polje sa porukom koja će upozoriti da podatak ne odgovara formatu koji je određen za to polje.

Tip polja određuje šablon vrednosti koji se očekuje za unos. Tip podatka *email* zahteva *email* adresu, tip podatka *url* zahteva *URL* adresu sa protokolom, tip podatka *date* zahteva godinu, mesec i dan, itd. Moguće je prepisati ovaj zahtev sa atributom *pattern*. Vrednost za atribut *pattern* je regularni izraz ili *regex*, standardizovani način za zadavanje šablona po kojem skupu karaktera treba da bude struktuiran podatak prilikom unosa.

Definisanje unosa samo ispravno formatiranog serijskog broja proizvoda koji će počinjati sa dva slova biti praćen sa dva broja završavati sa tri cifre i karaktera “-” koji će razdajati delove.

```
<input type="text" pattern="[A-Z]{2}\-[0-9]{2}\-[0-9]{3}">
```

The screenshot shows a web form with the following fields and values:

- Naziv:** Samsung Galaxy S7
- Serijski broj:** ss-ss
- Kategorija:** ALL in One računar
- Korisnik:** admin

A tooltip message is displayed over the 'Serijski broj' field: "Please match the requested format." The form also includes a 'Dodaj' button at the bottom left.

Slika 3: Prikaz poruke koja će upozoriti korisnika da unos nije ispravan

Ukoliko probamo da upišemo unos koji neće ispoštovati neki deo, dobićemo poruku o grešci koja će zahtevati da koristimo ispravan šablon. U nekim pretraživačima moguće je prilagoditi tekst ove greške dodavanjem dodatnog teksta koji će se ispisivati ispod originalnog teksta preko atributa *title*.

```
<input type="text" pattern="[A-Z]{2}\-[0-9]{2}\-[0-9]{3}" title="Šablon:F0-01-789">
```

2.3 Tabele

HTML tabela sastoji se od redova i kolona, koristi se kako bi se organizovao prikaz informacija u struktuiranom obliku. Neki programeri koriste tabele za pozicioniranje ili poravnavanje sadržaja sa slikama, ali to nije najbolji način za korišćenje tabela.

Svaka *HTML* tabela počinje sa obeležjem `<table>`. Redovi su obeleženi sa `<tr>` elementom, zaglavlja kolona sa `<th>` elementom, a ćelije su definisane sa `<td>` elementom. Obeležje `<td>` predstavlja element koji će sadržati podatke unutar sebe, a podaci mogu da budu tekstovi, slike, liste, druge tabele itd.

Kada pravimo veliku tabelu koja zahteva listanje u pretraživaču, treba koristiti `<thead>`, `<tfoot>` i `<tbody>` elemente. Sadržaj unutar zaglavlja tabele i podnožja će ostati na svojim pozicijama dok će sadržaj koji se nalazi unutar `<tbody>` biti izlistan između njih.

Element `<thead>` kreira kolonu zaglavlja, a `<tfoot>` element koristi se da prikaže poslednji red kao što je recimo suma svih proizvoda. Element `<tbody>` definiše sav sadržaj između zaglavlja i podnožja.

Primer strukture *HTML* tabele:

```
<table border="1">
  <thead>
    <tr>
      <th>Id</th>
      <th>Naziv</th>
      <th>Product number</th>
      <th>Kategorija</th>
      <th>Korisnik</th>
      <th>Opcije</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>TP-Link TL-ER6120</td>
      <td>RT-01-159</td>
      <td>Ruter</td>
      <td>Admin</td>
      <td>Izmena</td>
    </tr>
  </tbody>
</table>
```

Id	Naziv	Product number	Kategorija	Korisnik	Opcije
1	TP-Link TL-ER6120	RT-01-159	Ruter	Admin	Izmena

Slika 4: Prikaz tabele u pretraživaču nakon obrade *HTML* obeležja

3 CSS

CSS (*Cascading Style Sheets*) predstavlja jezik koji definiše stilove za *HTML* elemente. Da bi se napravio korisnički interfejs ili sajt, i da bi se održao izgled i funkcionalnost sajta i aplikacija potrebno je dobro poznavanje CSS i jasno razumevanje kako funkcioniše u kombinaciji sa drugim alatima kao što su *HTML* i *JavaScript*.

3.1 Prilagodljiv dizajn

Prava revolucija u CSS došla je kroz media upite. Sintaksa koja omogućava da obezbedimo stilove na pretraživačima na osnovu njihovih dimenzija i sposobnosti, Ovaj pristup predstavlja prvi korak ka omogućavanju pravih multifunkcionalnih veb stranica, koji će moći da se učitavaju na bilo kojoj dimenziji ekrana, i stranica koje će biti prilagođena za uređaj na kojem se prikazuje.

Media pravilo se koristi da se definišu različita stilska pravila za različite vrste uređaja. U CSS2 pravilo se zvalo media tipovi (engl. *Media types*), dok se u CSS3 zove media upiti (engl. *Media queries*).

Media upiti obraćaju pažnju na karakteristike uređaja, i može se upotrebiti da se provere karakteristike kao što su:

- Širina i visina prikaza
- Orijentacija (da li je tablet/mobilni telefon u *landscape* ili *portret* modu)
- Rezolucija uređaja

Korišćenje media upita je popularna tehnika za dostavljanje prilagođenih stilova za tablete, *iPhone* i *Android* mobilne telefone.

Media upiti se sastoje od media tipa (engl. *type*) i mogu da sadrže jedan ili više izraza, koji će rezultirati tačno ili netačno.

```
@media not | only mediatype and (expressions) {  
    CSS-Code;  
}
```

Rezultat upita je tačan ako se navedeni media tip podudara sa tipom uređaja na kojem se prikazuje i ako su svi izrazi i uslovi unutar media tipa tačni.

Ukoliko se ne primeni *not* ili *only* operater, media tip je u tom slučaju opcionalan i biće primenjen operater *all*.

Moguće je imati posebne stilske fajlove za svaki uređaj:

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">
```

Sledeći primer prilagođiće prikazivanje tabele na uređajima koji su manji od 767px:

```
@media screen and (max-width: 767px) {
  .table-responsive {
    width: 100%;
    margin-bottom: 15px;
    overflow-y: hidden;
    -ms-overflow-style: -ms-autohiding-scrollbar;
    border: 1px solid #ddd
  }
}
```

MP

Level > Here

Spisak svih proizvoda

Id	Naziv	Serijski broj	Kategorija	Ukupno ulaz	Ukupno izlaz	Korisnik
2	Nikon D7200	FO-01-789	Digitalni fotoaparati	20		admin
3	Canon Legria HF-G25	VO-01-661	Kamera	15	1	admin
4	Dell Inspiron 5559	LA-01-337	Laptop	5	10	admin
5	MacBook Air 11"	MC-01-951	MacBook	22		admin
6	HP Desktop AiO ProOne 600 G2	AL-01-246	ALL in One računar	51		admin
7	DELL PowerEdge R330 Xeon E3-1230 v5	SV-01-753	Server	4	3	admin
9	MFP InkJet A4 Canon PIXMA G2400	ST-01-852	Štampač	17	5	admin
10	TP-Link TL-ER6120	RT-01-159	Ruter	67		admin
11	Sony PlayStation 4	KO-01-482	Konzola	7		admin

Copyright © Marko Petrović 2012200210

Slika 5 : Prikaz izgleda tabele na mobilnom uređaju

4 JavaScript

JavaScript čini aplikacije dinamičnim i interaktivnim, prilagođava i odgovara na određene zahteve od strane krajnjih korisnika. Interaktivnost omogućava krajnjem korisniku da načini akciju u aplikaciji, obično tako što će kliknuti na dugme ili pritiskom na taster. Dinamične aplikacije prilagođavaju i odgovaraju na takve zahteve od strane korisnika. *JavaScript* takođe pruža mogućnost da postoje delovi prikaza koji su animirani ili imaju mogućnost da tokom korišćenja aplikacije menjaju svoja svojstva.

HTML i *CSS* su prvenstveno fokusirani na to kako je neki element struktuiran i pozicioniran, dok *JavaScript* više pažnje usmerava ka tome kako se nešto ponaša i kakvu interakciju uspostavlja sa krajnjim korisnikom.

4.1 Implementacija

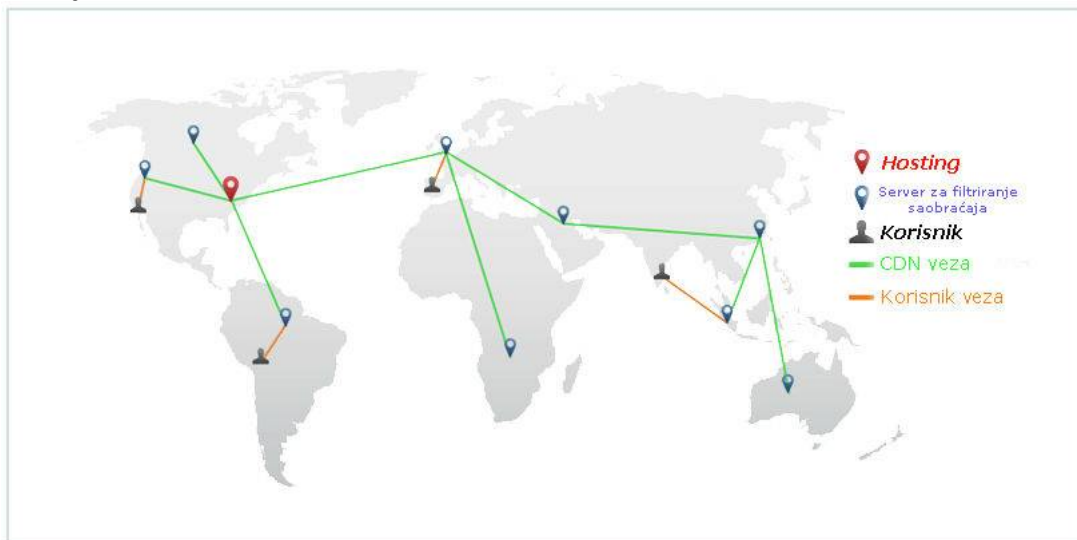
Postoje dva načina kako može da se uključi *Javascript* na veb stranici:

1. Preko eksternog fajla
2. Preko *CDN* (*Content Delivery Network*)

Eksterne skripte su praktične ukoliko se isti kod koristi na više različitih veb stranica. Takođe korisno je kako bi izvorni kod izgledao urednije. Da bi izvorni kod stranice bio čitljiviji dobra je praksa da se uz pomoć eksterne skripte razdvoji *Javascript* kod od *HTML* obeležja. Za eksterno učitavanje *Javascript* fajla u *HTML* koristi se obeležje `<script>` sa atributom `src`, atributu `src` će biti prosleđena putanja do fajla.

```
<script src="nazivFajla.js"></script>
```

Mreža za isporuku sadržaja (engl. *Content Delivery Network*, *CDN*) predstavlja sistem distribuiranih servera koji dostavljaju sadržaj korisniku na osnovu geografske lokacije korisnika.



Slika 6: Mreža za isporuku sadržaja (Content Delivery Network, CDN)

Ova usluga je efikasna u ubrzavanju procesa isporuke sadržaja sajtovima sa velikim prometom i sajtovima koji pružaju svoju uslugu korisnicima širom sveta. Što je geografski bliži *CDN* server korisniku, sadržaj će pre stići do korisnika. *CDN* takođe obezbeđuje zaštitu od preopterećenostima zahtevima za isporuku sadržaja od strane korisnika.

Serveri koji su najbliži posetiocu sajta će odgovoriti na zahtev. *CDN* kopira stranice sajta na mrežu servera koji su postavljeni na geografske različite lokacije i kešira sadržaj stranice. Kada korisnik pošalje zahtev za stranicu/fajl koji je deo mreže za isporuku sadržaja, *CDN* će preusmeriti zahtev servera na kojem se nalazi sajt na *CDN* server koji je najbliži posetiocu sajta i dostaviće keširan sadržaj. *CDN* će komunicirati sa serverom na kojem se nalazi sadržaj da bi dostavio i bilo koji sadržaj koji pre toga nije bio keširan.

Primer implementacije *jQuery Javascript* biblioteke putem mreže za isporuku sadržaja:

```
<head>  
| <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>  
</head>
```

5 AdminLTE

AdminLTE je popularni šablon za kreiranje administratorskog kontrolnog panela, *AdminLTE* je otvorenog koda. Razvijen je na osnovi *Bootstrap* okvira (engl. framework).

Bootstrap predstavlja besplatan skup alata za kreiranje veb sajtova i veb aplikacija. Baziran je na *HTML* i *CSS* šablonima za tipografiju, kreiranje formulara, dugmadi, navigacionih i ostalih komponenti, kao i razne šablone i proširenja za *JavaScript*. Cilj *Bootstrap* okvira je olakšavanje programiranja za veb.

AdminLTE je baziran na modularnom dizajnu, koji omogućava da se jednostavno prilagodi različitim potrebama i implementira, i prema zahtevima da se nadogradi u zavisnosti od situacije.

AdminLTE može da se skine u dve različite verzije:

1. Kompajliran i spreman za korišćenje u produkcionom okruženju. Ova verzija je dobra ukoliko ne postoji potreba za izmenom *AdminLTE LESS* fajlova.

2. Svi fajlovi uključujući i kompajlirani *CSS*. Ova verzija je pogodna ukoliko želimo da izmenimo određene stilizovane elemente panela, ali za tako nešto biće nam potreban i *LESS* kompajler.

LESS predprocesor (engl. *Preprocessors*) je program koji radi na strani servera, nudi proširenja i skraćeno pisanje sintakse u *CSS* opisnom jeziku koji se transformiše u ispravno formatiran stilski fajl nakon izvršavanja. Ova proširenja nude novine u vidu promenljivih, ugnježenih pravila i prilagođenih funkcija koje ubrzavaju razvoj i smanjuju potrebno vreme za izradu.

5.1 Implementacija

Za implementaciju *AdminLTE* alata potrebno je unutar `<head>` obeležja implementirati *Bootstrap* stilski fajl, *AdminLTE* stilski fajl i fajl koji stilizuje *AdminLTE* temu:

```
<link href="<?php echo Configuration::BASE; ?>assets/css/bootstrap.min.css" rel="stylesheet">
<link href="https://cdnjs.cloudflare.com/ajax/libs/ionicons/2.0.1/css/ionicons.min.css" rel="stylesheet">
<!-- Stilovi za AdminLTE temu -->
<link href="<?php echo Configuration::BASE; ?>assets/css/AdminLTE.min.css" rel="stylesheet">
<!-- AdminLTE boje za temu. Moguće je odabrati i druge boje za temu.
     | Potrebno je klasu primeniti i na obeležje body kako bi se stil primenio.
-->
<link href="<?php echo Configuration::BASE; ?>assets/css/skin-blue.min.css" rel="stylesheet">
```

Struktura fajlova *AdminLTE* alata:



5.2 Raspored elemenata

Raspored elemenata se sastoji od četiri glavna dela:

- Omotač(engl. *Wrapper*) *.wrapper*. Predstavlja `<div>` element koji obmotava čitav sajt.
- Glavno zaglavlje (engl. *Main header*) *.main-header*. Sadrži unutar sebe logo i navigacioni bar.
- Navigacioni panel (engl. *Sidebar-wrapper*) *.sidebar-wrapper*. Sadrži unutar sebe korisnički panel i navigacioni meni.
- Sadržaj (engl. *Content*) *.content-wrapper*. Sastoji se od zaglavlja stranice i sadržaja.

5.3 Opcije rasporeda elemenata

AdminLTE obezbeđuje set opcija koji mogu da se primene na glavni raspored. Svaka od navedenih klasa mogu da se pridruže `<body>` obeležju kako bi se postigao željeni rezultat.

- Fiksirano: potrebno je koristiti klasu *.fixed* da bi se dobio fiksirano zaglavlje i navigacioni panel sa leve strane.
- Skupljen navigacioni panel: koristiti klasu *.sidebar-collapse* ukoliko želimo skupljen navigacioni panel prilikom učitvanja stranice
- Navigacija u zaglavlju: koristiti klasu *.layout-top-nav* da bi se napravio navigacioni panel u zaglavlju.

Glavno zaglavlje sadrži u sebi logo i navigaciju. Konstrukcija navigacije razlikuje se u odnosu na *Bootstrap* konstrukciju navigacije pošto obezbeđuje određene elemente koje *Bootstrap* nema. U primeru se nalazi prikaz glavnog zaglavlja:

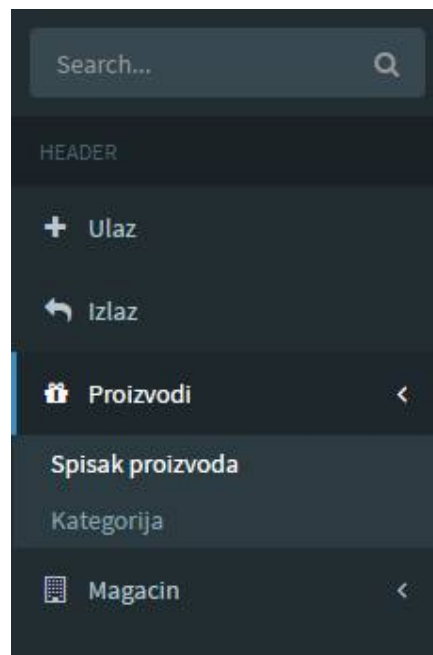
```
<header class="main-header">
  <a href="index.php" class="logo">
    <span class="logo-mini"><b>M</b>P</span>
    <span class="logo-lg"><b>M</b>P</span>
  </a>
  <nav class="navbar navbar-static-top" role="navigation">
    <a href="#" class="sidebar-toggle" data-toggle="offcanvas" role="button">
      <span class="sr-only">Toggle navigation</span>
    </a>
    <div class="navbar-custom-menu">
      <ul class="nav navbar-nav">
        <li class="dropdown messages-menu">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown">
            <i class="fa fa-envelope-o"></i>
            <span class="label label-success">4</span>
          </a>
        </li>
        <li class="dropdown notifications-menu">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown">
            <i class="fa fa-bell-o"></i>
            <span class="label label-warning">10</span>
          </a>
        </li>
        <li class="dropdown tasks-menu">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown">
            <i class="fa fa-flag-o"></i>
            <span class="label label-danger">9</span>
          </a>
        </li>
        <li>
          <a href="#" data-toggle="control-sidebar"><i class="fa fa-gears"></i></a>
        </li>
      </ul>
    </div>
  </nav>
</header>
```



Slika 7 : Prikaz glavnog zaglavlja u pretraživaču

U sledećem primeru prikazana je struktura navigacionog panela, koji će biti prikazan na levoj strani korisničkog panela:

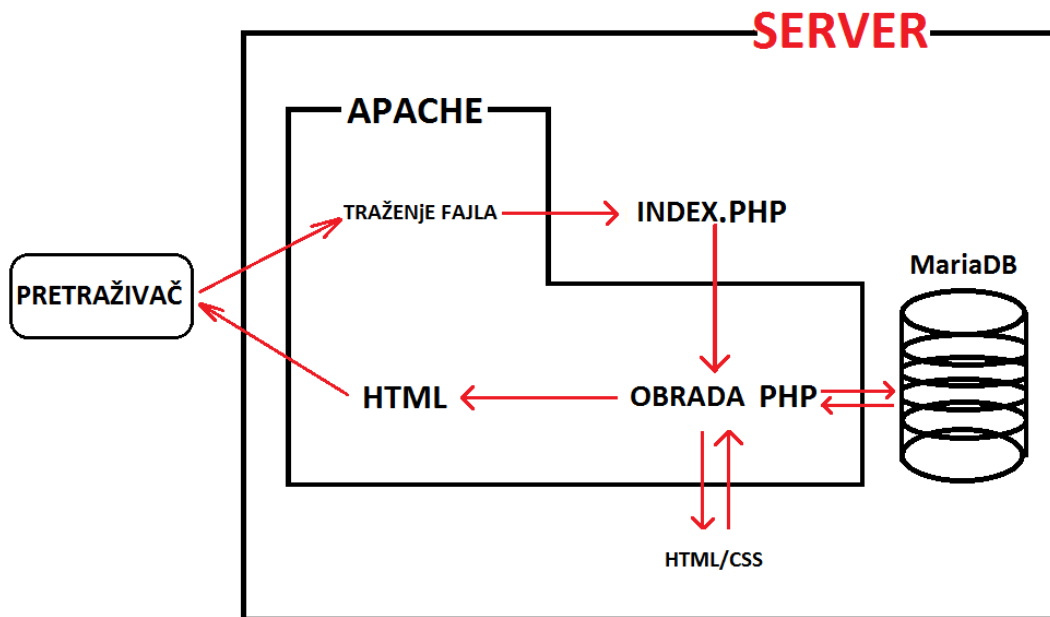
```
<section class="sidebar">
  <ul class="sidebar-menu">
    <li class="header">Zaglavlje</li>
    <li class="active"><a href="#"><i class="fa fa-plus"></i> <span>Ulaz</span></a></li>
    <li><a href="#"><i class="fa fa-reply"></i> <span>Izlaz</span></a></li>
    <li class="treeview">
      <a href="#"><i class="fa fa-gift"></i> <span>Proizvodi</span>
      <span class="pull-right-container">
        <i class="fa fa-angle-left pull-right"></i>
      </span>
      </a>
      <ul class="treeview-menu">
        <li><a href="#">Spisak proizvoda</a></li>
        <li><a href="#">Kategorija</a></li>
      </ul>
    </li>
    <li class="treeview">
      <a href="#"><i class="fa fa-building-o"></i> <span>Magacin</span>
      <span class="pull-right-container">
        <i class="fa fa-angle-left pull-right"></i>
      </span>
      </a>
    </li>
  </ul>
</section>
```



Slika 8: Prikaz navigacionog panela u pretraživaču

6 PHP

PHP je programski jezik koji se izvršava na strani servera. *PHP* obrađuje zahteve uz pomoć *Apache* servera. Kada korisnik pošalje zahtev za određenim resursom, zahtev će preuzeti *Apache* server. *PHP* će započeti komunikaciju sa *MariaDB* bazom podataka i iščitati potrebne podatke. Nakon toga, uspostaviće konekciju sa folderima kako bi preuzeo potrebnu strukturu stranice i *CSS* stilove. Kada preuzme sve što je potrebno, počće sa obradom i vratiće korisniku željeni prikaz, bez *PHP* sintakse u izvornom kodu, pošto se sve izvršava na strani servera.



Slika 9: Prikaz obrade zahteva od strane PHP programskog jezika

6.1 Pripremljeni upiti

Pripremljeni upiti (*PDO*) je mogućnost da se koristi za izvršavanje istih (ili sličnih) *SQL* upita u više navrata sa visokom efikasnošću.

Pripremljeni upiti rade na sledeći način:

- Priprema: *SQL* upit je kreiran i poslat ka bazi podataka. Određene vrednosti su ostale neodređene i nazivaju se parametrima (označene sa “?”)
Primer: `INSERT INTO MyGuests VALUES (?, ?, ?)`
- Baza podataka parsira, kompajlira, radi optimizaciju *SQL* upita i skladišti rezultat bez njegovog izvršavanja.
- Izvršavanje: aplikacija vezuje vrednosti za parametre i u bazi podataka se izvršava upit. Aplikacija može da izvrši neograničen broj puta upit sa različitim vrednostima.

U poređenju sa izvršavanjem SQL upita direktno, pripremljeni upiti imaju dve glavne prednosti:

- a. Smanjuje vreme za parsiranje jer se priprema upita radi samo jednom (iako se izvršavanje upita izvršava više puta).
- b. Vezani parametri smanjuju protok ka serveru pošto su potrebni samo parametri, a nije potreban da se šalje čitav upit ka serveru svaki put.

U SQL upitu stavljamo oznaku “?” tamo gde želimo kasnije tu oznaku da zamenimo sa integer, string ili double vrednostima

7 MariaDB

MariaDB je relacionalna baza podataka, koja predstavlja poboljšanu verziju *MySQL* i dostupna je pod *GPL* licencom. Zbog nepoverenja prema *Oracle* jer imaju u vlasništvu *MySQL*, programeri koji su radili na razvoju *MySQL* relacionalne baze podataka, odvojili su se i napravili su *MariaDB*. Kako je vreme odmicalo, *MariaDB* je počela da zamenjuje *MySQL*.

Prednosti *MariaDB* u odnosu na *MySQL* :

1. Bezbednost

Oracle pravi bezbednosne objave na svaka tri meseca za sve njihove proizvode. Međutim, *MySQL* ima novu verziju na svaka dva meseca. Nekada ovakva situacija dovodi do toga da bezbednosna poboljšanja i bezbednosne informacije nisu usklađene.

MariaDB prati dobre industrijske standarde sa bezbednosnim objavama i nadogradnjama na pravi način.

2. Sistem za skladištenje

MariaDB se posebno ističe sa svojim sistemom skladištenja i dodacima koji dolaze uz nju : povezivanje za *Cassandra* sistem skladištenja za *NoSQL*, *Spider* sistem skladištenja, *TokuDB* sa fraktalnim indeksima, itd. Ova proširenja u *MariaDB* su deo zvanične verzije, koji garantuje da su proširenja dobro integrisana i laka za korišćenje.

3. Bolje performanse

MariaDB ima mnogo bolje optimizovane upite. Određeni pokazatelji (engl. *Benchmark*) ukazuju da je *MariaDB* znatno brža u odnosu na *MySQL*.

4. Galera cluster sistem

Galera je nova vrsta *cluster* sistema, koja u odnosu na tradicionalnu *MySQL master-slave* repliku, obezbeđuje *master-master* repliku, koja omogućava novu vrstu skalabilne arhitekture za *MySQL/MariaDB*. *Galera* je postala sastavni deo *MariaDB* od verzije 10.1.

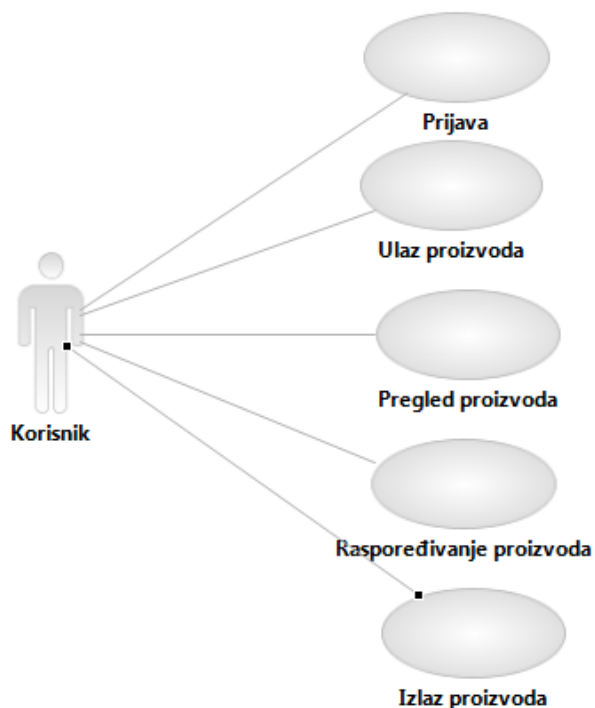
U *XAMPP* serverskom paketu koji omogućava konfigurisanje veb servera na lokalnom računaru, *MySQL* server je zamenila *MariaDB*. Pored *MariaDB*, *XAMPP* unutar sebe uključuje *Apache* server, *PHP* i *Perl* programski jezik.

8 Aplikacija

Nakon opisa tehnologije koje su korišćene za realizaciju aplikacije za evidenciju robe, u nastavku rada nalazi se prikaz funkcionalnosti sistema koji su implementirani.

8.1 Pregled funkcija administratora

Način funkcionisanja aplikacija sa stanovišta administratora može se predstaviti preko *UML* dijagrama slučajeva korišćenja:



8.2 Uspostavljanje konekcije ka bazi podataka

Ova klasa predstavlja centralno mesto gde se nalaze svi bitni konfiguracioni parametri veb aplikacije, koji su definisani kao članovi podaci konstante.

```

final class Configuration {
    const DB_HOST = 'localhost';
    const DB_USER = 'root';
    const DB_PASS = '';
    const DB_BASE = 'evidencija_robe';

    const BASE = 'http://localhost/evidencija_robe/';
    const PATH = '/evidencija_robe/'; # Na normalnom sajtu to ce biti samo /

    const SALT = '09560ughkjfgh08u35608utrgho8u54yjhi0569uyrth';

    const PRODUCT_IMAGE_DATA_PATH = 'data/images/';
}

```

Metod vraća konekciju ka bazi podataka (instancu klase *PDO*) za već postojeću konekciju, ako ovaj objekat nije instanciran, odnosno nije ostvarena veza ka bazi podataka, metod uspostavlja konekciju ka *MariaDB* serveru sa parametrima zadatim u klasi *Configuration* i tek zatim vraća kao rezultat objekat klase *PDO* za uspostavljenu konekciju

```

public static function getInstance() {
    if (self::$db === null) {
        self::$db = new PDO('mysql:host=' . Configuration::DB_HOST .
                           ';dbname=' . Configuration::DB_BASE . ';charset=utf8',
                           Configuration::DB_USER,
                           Configuration::DB_PASS);
        self::$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
    }

    return self::$db;
}

```

8.3 Prijava

Pre početka korišćenja sistema za evidenciju robe, potrebno je da se korisnik autentifikuje sa ispravnim korisničkim nalogom i šifrom, kako bi uspešno mogao da nastavi korišćenja sistema. Nakon uspešne autentifikacije korisnik će biti preusmeren na korisnički panel.

Slika 10 : Prikaz izgleda stranice za prijavu korisnika

Funkcija proverava da li postoje podaci za prijavu poslatih HTTP POST metodom, vrši njihovu validaciju, proverava postojanje korisnika sa prosleđenim pristupnim parametrima i u slučaju da sve provere prođu bez greške metod kreira sesiju za korisnika i preusmerava korisnika na unapred definisanu rutu.

```
function login() {
    if ($_POST) {
        $username = filter_input(INPUT_POST, 'username');
        $password = filter_input(INPUT_POST, 'password');

        if (!preg_match('/^[a-z0-9]{4,}$/', $username) or !preg_match('/^\.{5,}$/', $password)) {
            $this->set('message', 'Invalid username or password format.');
```

```
            return;
        }

        $passwordHash = hash('sha512', $password . Configuration::SALT);
        $user = UserModel::getActiveUserByUsernameAndPasswordHash($username, $passwordHash);

        if (!$user) {
            $this->set('message', 'Username ili password su pogrešni ili korisnik nije aktivan u db.');
```

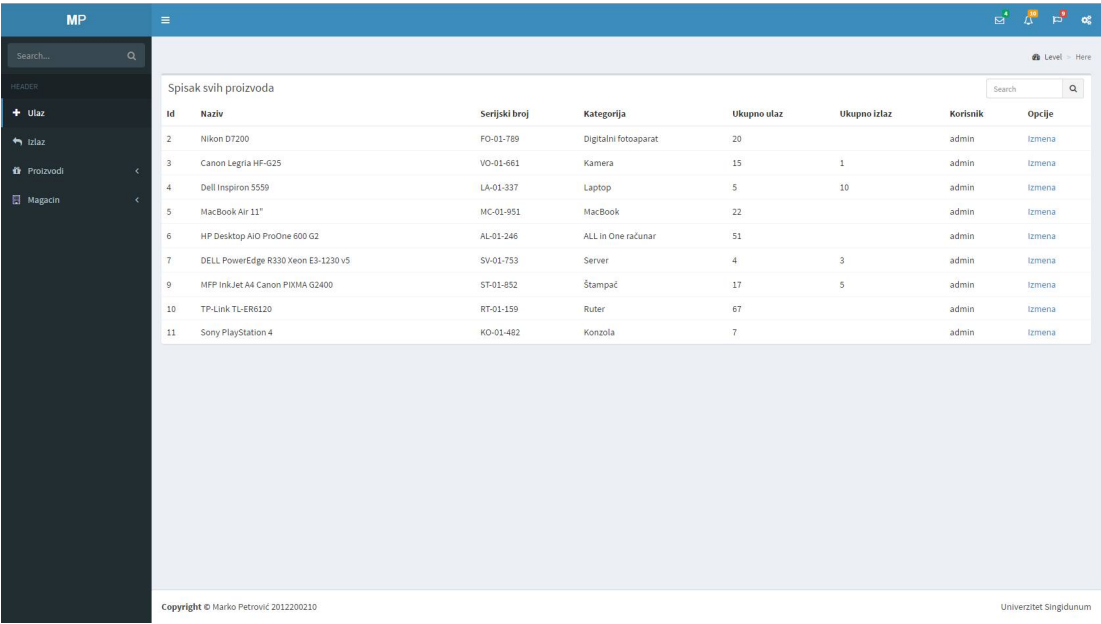
```
            return;
        }

        Session::set('user_id', $user->user_id);
        Session::set('username', $username);
        Session::set('user_ip', filter_input(INPUT_SERVER, 'REMOTE_ADDR', FILTER_FLAG_IPV4));
        Session::set('user_agent', filter_input(INPUT_SERVER, 'HTTP_USER_AGENT'));

        Misc::redirect('');
```

8.4 Pregled stanja proizvoda u magacinu

Funkcija pregled stanja proizvoda u magacinu, pruža prikaz sume pojedinačnih proizvoda koji su ušli u magacin, i sumu izlaza pojedinačnih proizvoda iz magacina.



Id	Naziv	Serijski broj	Kategorija	Ukupno ulaz	Ukupno izlaz	Korisnik	Opcije
2	Nikon D7200	FO-01-789	Digitalni fotoaparati	20		admin	Izmena
3	Canon Legria HF-G25	VO-01-661	Kamera	15	1	admin	Izmena
4	Dell Inspiron 5559	LA-01-337	Laptop	5	10	admin	Izmena
5	MacBook Air 11"	MC-01-951	MacBook	22		admin	Izmena
6	HP Desktop AIO ProOne 600 G2	AL-01-246	ALL in One računar	51		admin	Izmena
7	DELL PowerEdge R330 Xeon E3-1230 v5	SV-01-753	Server	4	3	admin	Izmena
9	MFP InkJet A4 Canon PIXMA G2400	ST-01-852	Štampač	17	5	admin	Izmena
10	TP-Link TL-ER6120	RT-01-159	Ruter	67		admin	Izmena
11	Sony PlayStation 4	KD-01-482	Konzola	7		admin	Izmena

Slika 11: Prikaz stanja proizvoda u magacinu

U kontroleru *AdminProizvodContoller* koji je kreiran za upravljanje proizvodima u magacinu, kreirana je funkcija koja će vraćati ukupno stanje ulaza i izlaza pojedinačnih proizvoda u magacinu.

```

/**
 * Indeks metod admin kontrolera za rad sa proizvodima prikazuje spisak svih proizvoda u magacinu
 */
public function index() {
    $this->set('proizvod', ProizvodModel::getAll());
}

/**
 * Metod koji vraca spisak proizvoda i ukupna stanja ulaza i izlaza proizvoda
 * @return array
 */
public static function getAll() {
    $SQL = 'SELECT p.*,
                k.naziv kategorija_naziv,
                u.username korisnik,
                SUM(ul.kolicina) sum_ulaz,
                SUM(iz.kolicina) sum_izlaz
            FROM proizvod p
            LEFT JOIN kategorija k ON p.kategorija_id = k.kategorija_id
            LEFT JOIN ulaz ul ON p.proizvod_id = ul.proizvod_id
            LEFT JOIN izlaz iz ON p.proizvod_id = iz.proizvod_id
            LEFT JOIN `user` u ON p.user_id=u.user_id
            GROUP BY proizvod_id;';
    $prep = DataBase::getInstance()->prepare($SQL);
    $prep->execute();
    return $prep->fetchAll(PDO::FETCH_OBJ);
}

```

8.5 Dodavanje proizvoda

Funkcija administratora koja omogućava da se doda novi proizvod sa pripadajućom kategorijom i serijskim brojem proizvoda, pre nego što će biti omogućen ulaz proizvoda u magacin.

Slika 12 : Prikaz dodavanja novog proizvoda

Funkcija u *AdminProizvodController* koja će omogućiti administratoru da napravi novi proizvod sa pripadajućom kategorijom proizvoda i serijskim brojem proizvoda.

```

/**
 * Ovaj metod prikazuje formuilar za dodavanje ili vrši dodavanje ako su podaci poslali HTTP POST metodom
 * @return void
 */
public function add() {
    $this->set('kategorija', KategorijaModel::getAll());
    $this->set('user', UserModel::getAll());

    if (!$_POST) return;

    $naziv = filter_input(INPUT_POST, 'naziv');
    $product_number = filter_input(INPUT_POST, 'product_number');
    $kategorija_id = filter_input(INPUT_POST, 'kategorija_id', FILTER_SANITIZE_NUMBER_INT);

    $proizvod_id = ProizvodModel::add($naziv, $product_number, $kategorija_id, Session::get('user_id'));

    if ($proizvod_id) {Misc::redirect('admin/proizvod/');}
    else {$this->set('message', 'Doslo je do greske prilikom dodavanja proizvoda u bazu podataka.');}
}
/**
 * Metod vrši unos novog proizvoda u bazu podataka
 * @param string $naziv
 * @param string $product_number
 * @param int $kategorija_id
 * @param int $user_id
 * @return boolean
 */
public static function add($naziv, $product_number, $kategorija_id,$user_id) {
    $SQL = 'INSERT INTO proizvod (naziv, product_number, kategorija_id,user_id)
    VALUES (?, ?, ?, ?)';
    $prep = DataBase::getInstance()->prepare($SQL);
    $res = $prep->execute([$naziv, $product_number, $kategorija_id,$user_id]);
    if ($res) return DataBase::getInstance()->lastInsertId();
    else return false;
}

```

8.6 Izmena proizvoda

Funkcija koja omogućava administratoru sistema izmenu podataka vezana za proizvod ukoliko je uneta pogrešno u bazu podataka.

Slika 13 : Prikaz izmene postojećeg proizvoda

```

/**
 * Ovaj metod prikazuje formular za izmenu ili vrsi izmenu ako su podaci poslati HTTP POST metodom
 * @return void
 */
public function edit($id) {
    $this->set('kategorija', KategorijaModel::getAll());
    $this->set('user', UserModel::getAll());

    $proizvod = ProizvodModel::getById($id);
    if (!$proizvod) {
        Misc::redirect('admin/proizvod/');
    }
    $this->set('proizvod', $proizvod);
    if (!$_POST) return;

    $naziv = filter_input(INPUT_POST, 'naziv');
    $product_number = filter_input(INPUT_POST, 'product_number');
    $kategorija_id = filter_input(INPUT_POST, 'kategorija_id', FILTER_SANITIZE_NUMBER_INT);
    $user_id = filter_input(INPUT_POST, 'user_id', FILTER_SANITIZE_NUMBER_INT);

    $res = ProizvodModel::add($naziv, $product_number, $kategorija_id, $user_id);
    if ($res) {Misc::redirect('admin/proizvod/');}
    else {$this->set('message', 'Doslo je do greske prilikom izmene podataka o proizvodu.');}
}

/**
 * Metod menja sadrzaj zapisa u bazi za izabrani proizvod
 * @param int $id
 * @param string $naziv
 * @param string $product_number
 * @param int $kategorija_id
 * @param int $user_id
 */
public static function edit($id, $naziv, $product_number, $kategorija_id, $user_id) {
    $SQL = 'UPDATE proizvod SET naziv = ?, product_number = ?, kategorija_id = ?, user_id = ?
        WHERE proizvod_id = ?;';
    $prep = DataBase::getInstance()->prepare($SQL);
    return $prep->execute([$naziv, $product_number, $kategorija_id, $user_id, $id]);
}

```

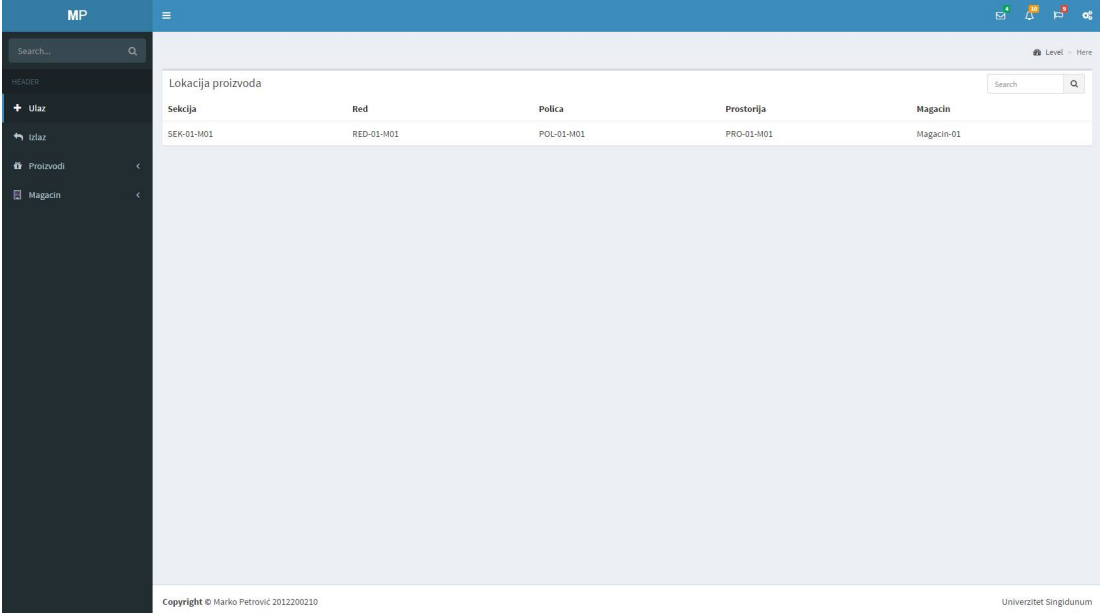
Funkcija prikazuje proizvode koje su ušli u magacin, količinu proizvoda i vreme unosa proizvoda.

- 27 -

Slika 14 : Prikaz svih ulaza proizvoda u magacin

8.8 Lokacija proizvoda u magacinu

Funkcija pregled lokacije proizvoda u magacinu, omogućava administratoru aplikacije da zna tačnu lokaciju na kojoj je proizvod raspoređen u magacinu.



Sekcija	Red	Polica	Prostorija	Magacin
SEK-01-M01	RED-01-M01	POL-01-M01	PRO-01-M01	Magacin-01

Slika 15: Prikaz lokacije proizvoda u magacinu

U kontroleru *AdminUlazController* koji je kreiran za lakšu kontrolu ulaza proizvoda u magacin, kreirana je funkcija koja će dati administratoru tačnu lokaciju na kojoj je proizvod raspoređen u magacinu.

```

/**
 * Lokacija metod admin kontrolera za rad sa lokacijama na kojima je smešten proizvod u magacinu
 */
public function lokacija($ulaz_id) {
    $lokacija = UlazModel::lokacijaRasporedjivanja($ulaz_id);
    $this->set('lokacija', $lokacija);
}

/**
 * Ovaj metod vraća lokaciju na kojoj je smešten proizvod u magacinu
 * @param type $ulaz_id
 * @return type array
 */
public static function lokacijaRasporedjivanja($ulaz_id) {
    $ulaz_id = intval($ulaz_id);
    $SQL = 'SELECT sek.oznaka sekcija_oznaka,
                re.oznaka red_oznaka ,
                pol.oznaka polica_oznaka,
                pro.oznaka prostorija_oznaka,
                mag.naziv magacin_naziv, rasp.*
    FROM sekcija sek
    LEFT JOIN red re ON sek.red_id = re.red_id
    LEFT JOIN polica pol ON re.polica_id=pol.polica_id
    LEFT JOIN prostorija pro ON pol.prostorija_id=pro.prostorija_id
    LEFT JOIN magacin mag ON pro.magacin_id=mag.magacin_id
    LEFT JOIN rasporedjivanje rasp ON rasp.sekcija_id=sek.sekcija_id
    WHERE ulaz_id=?';
    $prep = DataBase::getInstance()->prepare($SQL);
    $prep->execute([$ulaz_id]);
    return $prep->fetch(PDO::FETCH_OBJ);
}

```

8.9 Model baze podataka

Prikaz modela baze podataka sa svim tabelama i međusobnim vezama između tabela:

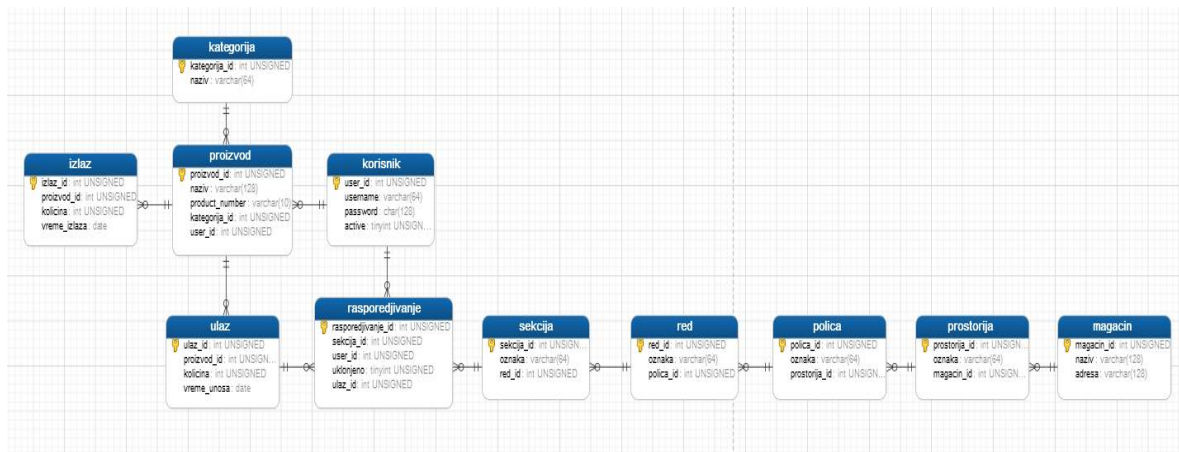


Tabela izlaz sa primarnim ključem izlaz_id i atributima proizvod_id, količina i vreme_izlaza

Fields	Indexes	Foreign Keys	Triggers	Options	Comment	SQL Preview	
Name	Type	Length	Decimals	Not null			
izlaz_id	int	10	0	☒			1
proizvod_id	int	10	0	☒			
kolicina	int	10	0	☒			
▶ vreme_izlaza	date	0	0	☒			

Tabela proizvod sa primarnim ključem proizvod_id i atributima naziv, product_number, kategorija_id i user_id

Fields	Indexes	Foreign Keys	Triggers	Options	Comment	SQL Preview	
Name	Type	Length	Decimals	Not null			
▶ proizvod_id	int	10	0	☒			1
naziv	varchar	128	0	☒			
product_number	varchar	10	0	☒			
kategorija_id	int	10	0	☒			
user_id	int	10	0	☒			

Tabela raspoređivanje sa primarnim ključem raspoređivanje_id i atributima sekcija_id, user_id, uklonjeno i ulaz_id.

Fields	Indexes	Foreign Keys	Triggers	Options	Comment	SQL Preview	
Name	Type	Length	Decimals	Not null			
▶ rasporedjivanje_id	int	10	0	☒			1
sekcija_id	int	10	0	☒			
user_id	int	10	0	☒			
uklonjeno	tinyint	1	0	☒			
ulaz_id	int	10	0	☒			

Tabela ulaz_id sa primarnim ključem ulaz_id i atributima proizvod_id, količina i vreme_unosa

Fields	Indexes	Foreign Keys	Triggers	Options	Comment	SQL Preview	
Name	Type	Length	Decimals	Not null			
▶ ulaz_id	int	10	0	☒			1
proizvod_id	int	10	0	☒			
kolicina	int	10	0	☒			
vreme_unosa	date	0	0	☒			

9 Zaključna razmatranja

Aplikacija za evidenciju robe u magacinu predstavlja savremenu aplikaciju, koja može da se koristi na bilo kom uređaju i kojoj se u bilo kom momentu može pristupiti. Ovakva osobina aplikacije u današnje vreme najverovatnije predstavlja i jednu od najvažnijih karakteristika aplikacije kako bi uspešno mogla da se plasira na tržištu.

Postupkom razdavajanja aplikacije na delove omogućava lakše održavanje aplikacije i proširivanje funkcionalnosti aplikacije u budućnosti, takođe i mogućnost korišćenja postojećeg koda bez potrebe pisanja istog dela.

U budućnosti sa daljem razvojem aplikacije bilo bi poželjno da se razvije i aplikacija za mobilne uređaje. Prednost dodatne aplikacije ogledao bi se u tome, što bi mobilna aplikacija bila konektovana na istu bazu podataka koju koristi i sistem za evidenciju robe i mogla bi uz pomoć kamere da skenira barkod proizvoda i da vidi prikaz stanja kao i lokacije raspoređenosti istog proizvoda u drugim magacinskim prostorima.

Revolucija u Internet i veb tehnologija su konstantne, odnosno traju od trenutka kada je Internet pušten u rad. Niko ne može tačno da pretpostavi dokle će se tehnologije razviti. Možda će se jednog dana sva poslovna logika sajtova, portala i aplikacija izvršavati na strani korisnika i personalni uređaji predstavljati servere za skladištenje izvornih fajlova.

Literatura

1. Veinović, M. i Jevremović, A. (2013) *Internet tehnologije, prvo izdanje*, Univerzitet Singidunum, Beograd
2. Cvetković, D., Marković, D., Savanović, N., (2015) *Multimedija*, Univerzitet Singidunum, Beograd
3. Veinović, M., Šimić, G., Jevremović, A., i Franc, I. (2013) *Baza podataka, prvo izdanje*, Univerzitet Singidunum, beograd
4. Njeguš, A. (2015) *Poslovni informacioni sistemi, sedmo izdanje*, Univerzitet Singidunum, Beograd
5. Mišković, V. (2016) *Praktikum – Internet i Veb tehnologije*, Univerzitet Singidunum, Beograd
<http://predmet.singidunum.ac.rs/course/view.php?id=261>
6. MariaDB Foundation, “*Learn – MariaDB.org*” (2016) Dostupno na veb adresi:
<https://mariadb.org/learn>