# Web aplikacija za nadzor opreme i inventuru

**Šapina, Ružica**

**Undergraduate thesis / Završni rad**

**2020**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:200:201876

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2024-05-06**

*Repository / Repozitorij:*

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

**Stručni studij**

# WEB APLIKACIJA ZA NADZOR OPREME I INVENTURU

**Završni rad**

**Ružica Šapina**

**Osijek, 2020.**

# SADRŽAJ


1. UVOD ..... 1
2. KORIŠTENE TEHNOLOGIJE ..... 2
2.1. Programski jezik PHP ..... 2
2.2. HTML (Hypertext Markup Language) ..... 2
2.3. CSS (Cascading Style Sheets) ..... 4
2.4. Razvojno okruženje Bootstrap ..... 5
2.5. MySQL ..... 6
2.6. Laravel ..... 6
2.6.1. Model ..... 6
2.6.2. Pogled (engl. View) ..... 7
2.1.3. Upravitelj (engl. Controller) ..... 8
3. OPIS PRAKTIČNOG RADA ..... 9
3. 1. Planiranje Web aplikacije ..... 9
3.2. Izrada Web aplikacije ..... 11
3.2.1. Implementacija baze podataka ..... 11
3.2.2. Prijava/registracija korisnika ..... 12
3.2.3. Implementacija modela ..... 15
3.2.4. Implementacija upravitelja ..... 16
3.2.5. Implementacija pogleda ..... 17
3.2.6. Dizajn aplikacije ..... 25
3.2.7. Implementacija Middleware-a ..... 27
4. ZAKLJUČAK ..... 29
LITERATURA ..... 30
SAŽETAK ..... 31
ŽIVOTOPIS ..... 33
PRILOZI ..... 34

# 1. UVOD

Cilj završnog rada je izrada web aplikacije koja služi kao pomoć pri organizaciji rada tvrtke. Aplikacija se temelji na jednostavnom unosu i praćenju opreme, lokacije (ureda), zaposlenika i ostalih stavki vezanih za organizaciju rada tvrtke. Također, prisutne su pripadajuće uloge za zaposlenike, ovisno o pravu pristupa podacima (administrator/korisnik). Aplikacija sadrži mogućnost QR skeniranja te korisnik aplikacije ima mogućnost skenirati QR kod pomoću Android uređaja te na taj način dobiti detaljan uvid u specifikacije određene opreme. Uz QR skeniranje, korisniku je omogućena opcija za izvoz podataka u različitim formatima (pdf, csv, xls), kopiranje podataka u međuspremnik te ispis istih.

# 2. KORIŠTENE TEHNOLOGIJE

## 2.1. Programski jezik PHP

PHP je skriptni jezik na strani poslužitelja koji je posebno prilagođen za razvoj dinamičkih web stranica, odnosno aplikacija. Predstavlja programski jezik koji korisniku je nevidljiv, odnosno vidljiv je samo rezultat koji izgleda kao normalna HTML stranica. Izvorno ga je kreirao programer Rasmus Lerdorf 1994. godine. Rasmus Lerdorf napisao je nekoliko *Common Gateway Interface* (CGI) skipti koje su omogućile jednostavno upravljanje osobnim stranicama. Nakon nekog vremena, dodane su značajke poput podrške mySQL baze podataka te je tada dobio naziv *Personal Home Page/Forms Interpreter*, odnosno PHP/FI. Danas je PHP jedan od najzastupljenijih web programskih jezika. Vrlo je sličan Perl-u, Python-u i Ruby-u, a po sintaksi je najsličniji C-u. Za razliku od spomenutog programskog jezika C, PHP (posebno PHP5) omogućuje objektno orijentirano programiranje. PHP karakterizira jednostavna sintaksa koja se generira unutar <?php i ?> oznaka te implementacija brojnih funkcija i struktura programa (for, if, do, do while, itd.). Također, često se koristi u kombinaciji s relacijskom bazom podataka. Relacijske baze podataka najčešće korištene za PHP su: MySQL, PostgreSQL, Oracle, ODBC, itd. Kao prednosti PHP-a mogu se navesti: softver otvorenog koda, jednostavan za učenje, široka podrška na internetu te neovisnost o operacijskom sustavu. [1]

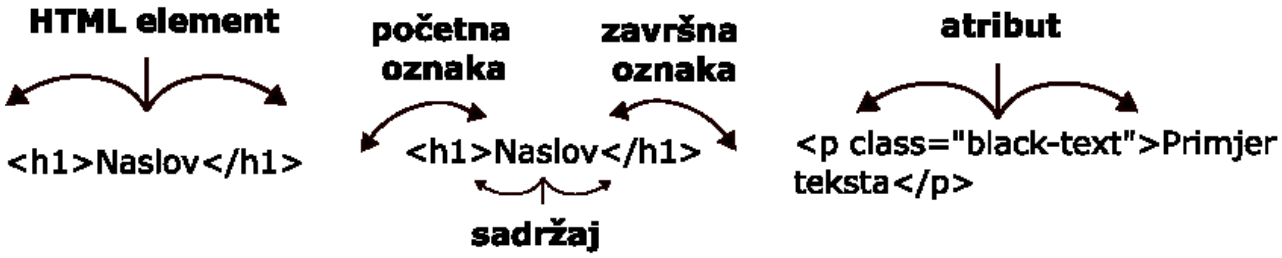
## 2.2. HTML (Hypertext Markup Language)

HTML je opisni jezik za oblikovanje sadržaja web stranice te stvaranja poveznica među hipertekstualnim dokumentima. Možemo reći da HTML pruža sredstva za stvaranje strukturiranih dokumenata označavanjem strukturne semantike za tekst kao što su naslovi, odlomci, popisi, veze, citati i druge stavke. Elementi su osnovna struktura HTML-a, a imaju dva osnovna svojstva: atribute i sadržaj. Svaki atribut i sadržaj imaju određena ograničenja kako bi se HTML dokument smatrao važećim. Većina HTML elemenata obilježena je parom oznaka, odnosno početnom i završnom oznakom. Oznaka početka uvijek počinje znakom "< ". Nakon toga slijedi naziv elementa (npr. "p" za odlomak ili "h1" za naslov) te opcionalno popis njegovih atributa (npr. klasa ili ID). Početna oznaka zatvorena je s "> ". Završna oznaka sastoji se od znakova "</ ", naziva elementa i znaka ">". Neki elementi, kao što je "<br>", nemaju sadržaj ili ne sadrže završnu oznaku. [2]

Svaki HTML element sadrži početnu  i završnu oznaku, a unutar oznaka nalazi se sadržaj opisan "h1", kao što je prikazano na slici 2.1. Iduća važna stavka HTML označavanja je atribut, na dolje prikazanoj slici radi se o "class" atributu, odnosno klasi "black-text" koja je opisana određenim css svojstvima te pridodana željenom elementu. Komentare je moguće pisati unutar oznaka <!— i  —> bilo gdje unutar HTML dokumenta te taj tekst neće biti prikazan na stranici.

**Programski kod 2.1**. Isječak HTML koda

```
<html>
<title>Ime stranice </title>
</head>
<body>
<h1>Naslov stranice</h1>
<h2>Podnaslov stranice <h2>

    <p>Primjer teksta koji pokazuje da internet preglednik sam oblikuje tekst bez obzira na format teksta napisan u HTML datoteci. </p>

</body>
</html>
```




**Sl.2.1.** Označavanje HTML dokumenta [Izvor: autor]

## 2.3. CSS (Cascading Style Sheets)

CSS predstavlja jezik za definiranje i stvaranje prezentacije strukturiranog dokumenta napisanog HTML jezikom. Glavni cilj razvoja CSS-a bio je odvajanje opisa logičke strukture web stranice (koja se izrađuje pomoću HTML-a ili drugih jezika) od izgleda web stranice. Takvo razdvajanje može povećati dostupnost dokumenta, pružiti veću fleksibilnost i mogućnost kontrole njegova izlaganja. Također, kao jedna od prednosti može se navesti mogućnost formatiranja više dokumenata na identičan način. [3]

CSS se može pisati unutar same HTML stranice, na tri načina:

1. stilovi unutar HTML datoteke (tj. između <style> i </style> oznaka),
2. unutar HTML oznaka (npr. <h3 style="color: #ffffff;">),
3. stvaranje vanjske .css datoteke (npr. <link rel="stylesheet" href="style.css" type="text/css">).

Prema slici 2.2., sintaksa CSS-a sastoji se od selektora i deklaracijskog bloka. Selektor označava HTML element (može biti i id ili klasa) na kojemu će biti primijenjen deklaracijski blok, a deklaracijski blok se sastoji od svojstava te njihovih vrijednosti.

**Sl.2.2.** CSS sintaksa [4]

## 2.4. Razvojno okruženje Bootstrap

Bootstrap predstavlja skup alata za izradu web stranica/aplikacija. Sadrži predloške temeljene na HTML-u i CSS-u prema kojima su definirane forme (engl. *forms*), gumbovi (engl. *buttons*), navigacija (engl. *navigation*), tipografija i ostale komponente sučelja. Bootstrap, razvijen od strane Mark Otto-a i Jacob Thornton-a temeljio se na planu za daljnji razvoj alata za internu analizu i administraciju Twittera. Prije Bootstrap-a korištene su razne biblioteke za razvoj sučelja što je zapravo dovelo do velikog opterećenja održavanja. Jedan od ključnih razloga popularnosti Bootstrapa je besplatan izvor (engl. *free-source*), kompatibilnost s većinom internet preglednika te brza i jednostavna implementacija. Nakon preuzimanja Bootstrap-a potrebno je raspakirati WinRar arhivu, te je tada moguće vidjeti osnovnu njegovu strukturu. [5] Struktura Bootstrap-a sastoji se od css, fonts i js datoteka, kao što je prikazano na slici 2.3.

```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   └── bootstrap-theme.min.css.map
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    ├── glyphicons-halflings-regular.woff
    └── glyphicons-halflings-regular.woff2
```

**Sl.2.3.** Struktura Bootstrap-a [6]

U navedene 3 datoteke se nalazi sve što je potrebno za brzo i jednostavno stvaranje web stranice.

1. CSS – sadrži formirane klase koje treba primijeniti na elemente stranice pisane u HTML-u.
2. fonts – sadrži oko 200 Glyphicons, odnosno ikona u font formatu koje se mogu koristiti na stranici u tekstu, gumbovima, navigaciji, formama itd.
3. js – sadrži JavaScript funkcije koje omogućavaju razne mogućnosti.

## 2.5. MySQL

MySQL jedan je od najčešće korištenih sustava za upravljanje relacijskim bazama podataka. Dostupan je kao softver otvorenog koda (engl. *open-source*) te čini osnovu za mnoge dinamične web stranice, odnosno aplikacije. MySQL je izgrađen na različitim bazama podataka na poslužitelju, gdje svaki korisnik obično ima pristup bazi podataka pomoću korisničkog imena i lozinke. MySQL se široko koristi u web alatima, kao što su Joomla i Wordpress te je njegova popularnost usko povezana sa spomenutim PHP-om i brojnim drugim programskim jezicima, također kompatibilan je s različitim operacijskim sustavima. [7]

## 2.6. Laravel

Laravel predstavlja besplatan, PHP programski okvir (engl. *framework*) čiji je izvorni kod dostupan svim korisnicima te ga mogu mijenjati i prepravljati (engl. *open-source*). Kreiran je od strane Taylora Otwella. Spomenuti programski okvir temelji se na MVC (engl. *Model-View-Controller*) arhitekturi. Prva inačica Laravela, nastala 2011.godine, bazirala se na autentifikaciji, lokalizaciji, modelima, sesijama i određivanju putanja (engl. *routingu*). No, svaka nova inačica dopunjavala je spektar Laravelovih karakteristika, odnosno mogućnosti o kojima ću detaljnije u nastavku. Model-Pogled-Upravitelj možemo definirati kao obrazac softverske arhitekture. Model se sastoji od podataka, poslovnih pravila, logike i funkcija ugrađenih u programsku logiku. Upravitelj (engl. *controller*) prihvaća ulazne parametre i pretvara ih u naloge modelu ili pogledu. Ovakva arhitektura olakšava nezavisan razvoj, testiranje i održavanje aplikacije. [8]

### 2.6.1. Model

ORM (engl. *Object-relation mapping*) jedna je od karakteristika Laravela koja osigurava jednostavnu ActiveRecord implementaciju vezanu za rad s bazom podataka. Svaka tablica podataka ima odgovarajući model kojemu je osnovna namjena interakcija, odnosno komunikacija s tablicom.

Modeli pružaju mogućnost query-ja te unošenja podataka u tablicu. Prije samoga rada s modelom, budući da se radi o bazi podataka, potrebno je konfigurirati spomenutu bazu u datoteci "config/database.php". Sljedeći korak je kreiranje modela, najjednostavniji način je upis naredbe "php artisan make:model Employee" u terminal te ukoliko se želi dodati tablica za spomenuti model, uz naredbu dodajemo nastavak "-m".

**Programski kod 2.2.** Primjer implementacije modela

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;

class Test  extends primjerModel
{
//
}
```

### 2.6.2. Pogled (engl. View)

Pogled (engl. *view*) sadrži HTML, odnosno prezentacijski jezik za izradu web stranica koji je opisan u prethodnom poglavlju. Cilj pogleda je zapravo odvajanje logike upravitelja od logike same prezentacije, točnije izgleda aplikacije. Pogledi se pohranjuju u mapu s putanjom "resources".

**Programski kod 2.3.** Primjer implementacije pogleda (engl. *view*)

```
<html>
<body>
<p>Ovo je primjer pogleda. </p>
</body>
</html>
```

Kako bi se pogled mogao otvoriti u pregledniku, važno je implementirati njegovu putanju u "routes.php" datoteci.

**Programski kod 2.4.** Primjer implementacije putanje u "routes.php" datoteci

```
Route::get('/', function()
{
    return view('primjerPogled');
});
```

Prema primjeru 2.4., proslijeđen je parametar koji se referencira na "primjerPogled.blade.php" datoteku.

### 2.1.3. Upravitelj (engl. Controller)

Kako bi se izbjeglo definiranje logike cijele aplikacije u spomenutoj "routes.php" datoteci, potrebna je organizacija unutar upravitelja, odnosno njihovih klasa. Relativna putanja upravitelja je "app/Http/Controllers".

**Programski kod 2.5.** Primjer implementacije upravitelja (engl. *controller*)

```
<?php
namespace App\Http\Controllers;
use App\User;
use App\Http\Controllers\Controller;

class primjerController extends Controller
{
    public function index
    {
        return view('primjerPogled');
    }
}
```

Implementirani isječak koda predstavlja primjer klase upravitelja koja nasljeđuje osnovnu klasu koja je zapravo integrirana od strane Laravela te pruža nekoliko ključnih funkcionalnosti, točnije metoda kao što je npr. metoda *middleware*-a koja će biti detaljnije opisana u jednom od sljedećih poglavlja.

**Programski kod 2.6.** Implementacija putanje u "routes.php" datoteci

```
Route::get('/primjerRute', 'primjerController@index');
```

Kako bi "primjerPogled" bio vidljiv u pregledniku, potrebno je definirati putanju u "routes.php" datoteci na način da se poziva upravitelj "primjerController" te njegovu metodu "index" koja vraća "primjerPogled" pogled.

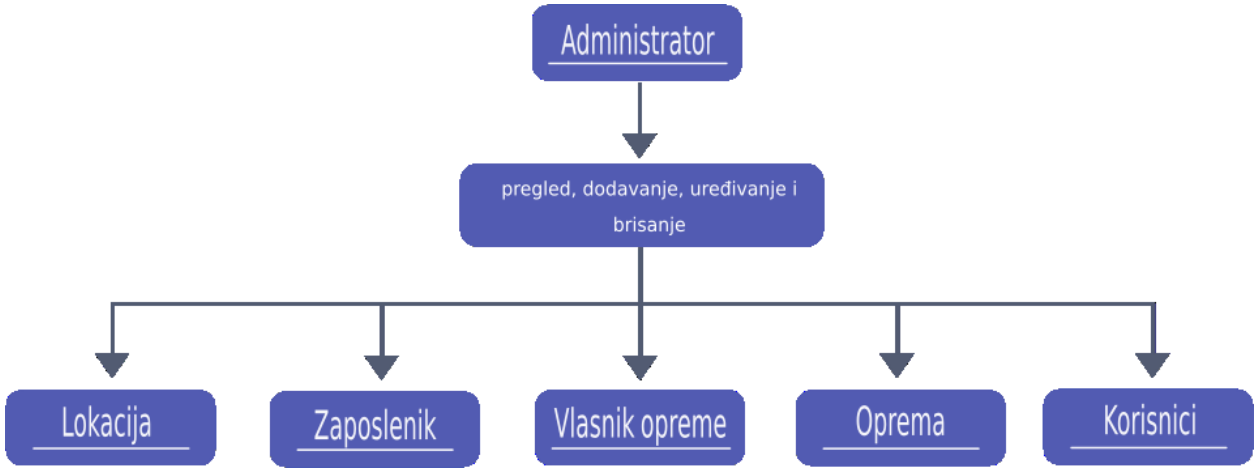
# 3. OPIS PRAKTIČNOG RADA

## 3. 1. Planiranje Web aplikacije

Pri osmišljavanju plana i samoj realizaciji projekta, točnije web aplikacije "Inventura" treba imati u vidu sljedeće stavke:

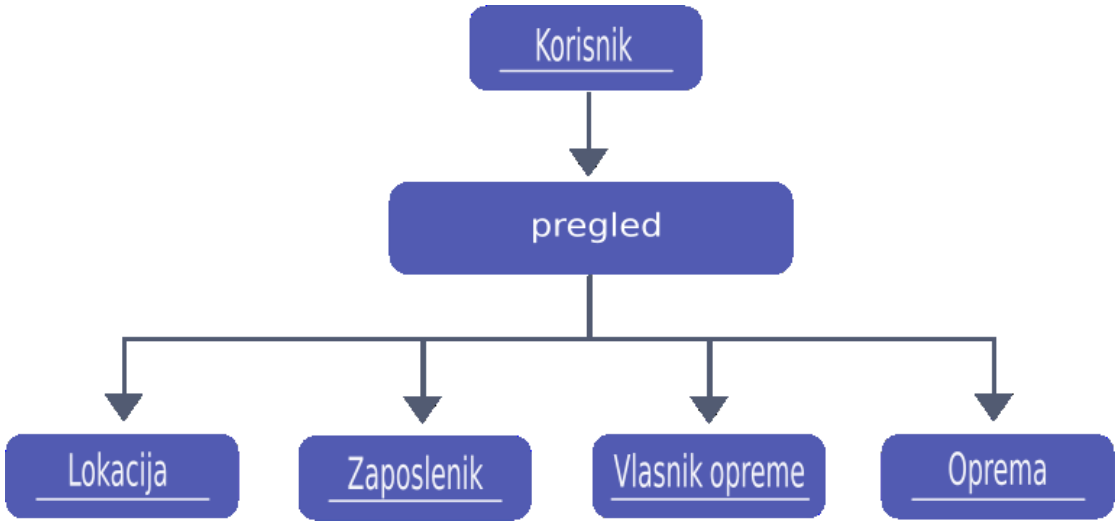
- izbor tehnologija koje će se koristiti pri izradi web aplikacije,
- funkcionalnost web aplikacije (brzo učitavanje, mogućnost korištenja tražilice, mogućnost pregleda, dodavanja, uređivanja, brisanja, izvoza u druge formate, itd.),
- sigurnost korisnika,
- isticanje bitnog sadržaja korištenjem prikladne boje, veličine i vrste fonta,
- praćenje trendova pri izradi web dizajna aplikacije,
- redovito održavanje web sustava i njegovo prilagođavanje napretku tehnologija.

Cilj zadatka, odnosno praktičnog dijela završnog rada bio je napraviti web aplikaciju "Inventura" čija je glavna svrha organizacija rada tvrtke, gdje bi se trebalo bazirati na nekakve osnovne parametre poslovne jedinice (zaposlenici, uredi, oprema i stavke vezane uz istu). Web aplikacija će biti podijeljena na administrativni i korisnički dio. U nastavku, prikazan je dijagram koji opisuje radnje dostupne administratoru.

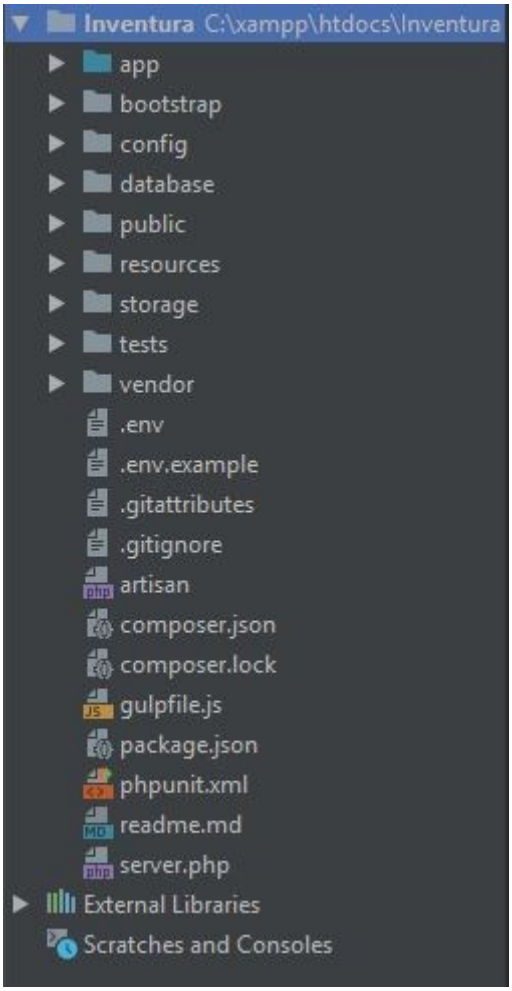



**Sl.3.1.** Dijagram s prikazom podataka kojima upravlja administrator [Izvor: autor]

Sljedeći dijagram prikazuje pregled podataka dostupnih korisniku. Svaki korisnik može kreirati svoj račun (registrirati se) ili se prijaviti ukoliko ima postojeći račun.




**Sl.3.2.** Dijagram s prikazom podataka koji su dostupni korisniku [Izvor: autor]




**Sl.3.3.** Struktura datoteka aplikacije [Izvor: autor]

## 3.2. Izrada Web aplikacije

Kroz ovo poglavlje bit će opisana implementacija web aplikacije "Inventura". Za izradu aplikacije koristile su se sljedeće programske tehnologije: Laravel (PHP framework), HTML, CSS te baza podataka MySQL. Korišteni programski okviri su Bootstrap te DataTable plugin. Od alata je korišten poslužiteljski paket XAMPP, za rad s bazom podataka aplikacija phpMyAdmin te za pisanje samog koda PHP Storm uređivač teksta.

### 3.2.1. Implementacija baze podataka

Implementacija aplikacije može se podijeliti u nekoliko dijelova. Prvi dio se odnosi na rad s bazom podataka, bilo je potrebno kreirati bazu "inventura" putem phpMyAdmin sučelja te implementirati ju u ".env" datoteku s ostalim varijablama.

**Programski kod 3.1.** Povezivanje s bazom podataka "inventura" (.env datoteka)

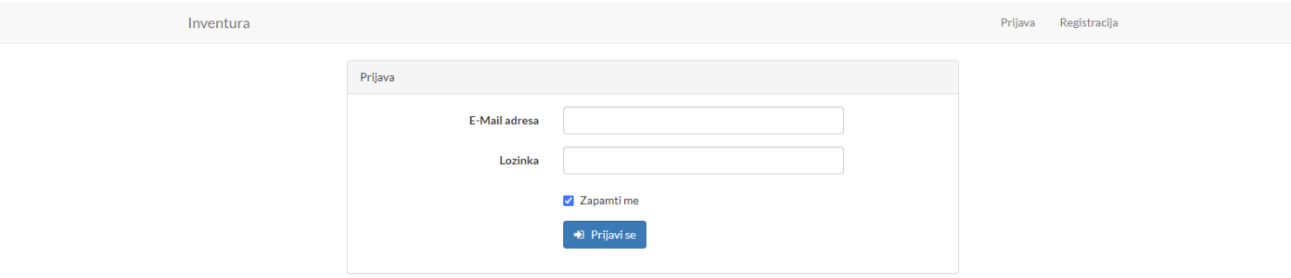
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=inventura
DB_USERNAME=root
DB_PASSWORD=
```

### 3.2.2. Prijava/registracija korisnika

Forma za prijavu nalazi se na početnom zaslonu aplikacije. Forma se sastoji od polja za unos e-mail adrese, lozinke te gumba za prijavu. Polja u formi su definirana html tagom "<input>" dok je gumb definiran html tagom "<button>". Unutar "<form>" taga nalazi se "atribute action" vezan za preusmjeravanje na putanju "/login". Klikom na gumb, generira se provjera o unesenim vrijednostima te ukoliko su podaci ispravno uneseni, korisnik je prijavljen te ima pristup ostalim dijelovima aplikacije ovisno o roli.

**Programski kod 3.2.** Isječak "login.blade.php" datoteke

```
<div class="panel-heading">Prijava</div>
<div class="panel-body">
    <form class="form-horizontal" role="form" method="POST" action="{{ url('/login') }}">
        {{ csrf_field() }}
        <div class="form-group{{ $errors->has('email') ? ' has-error' : '' }}">
            <label for="email" class="col-md-4 control-label">E-Mail adresa</label>
            <div class="col-md-6">
                <input id="email" type="email" class="form-control" name="email" value="{{ old('email') }}">

                @if ($errors->has('email'))
                    <span class="help-block">
                        <strong>{{ $errors->first('email') }}</strong>
                    </span>
                @endif
            </div>
        </div>
        <div class="form-group{{ $errors->has('password') ? ' has-error' : '' }}">
            <label for="password" class="col-md-4 control-label">Lozinka</label>

            <div class="col-md-6">
                <input id="password" type="password" class="form-control" name="password">

                @if ($errors->has('password'))
                    <span class="help-block">
                        <strong>{{ $errors->first('password') }}</strong>
                    </span>
                @endif
            </div>
        </div>
```



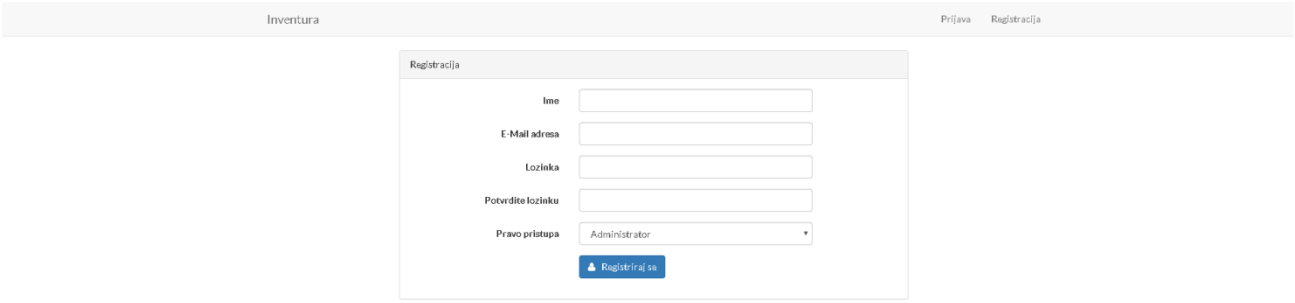

**Sl.3.4.** Početni zaslon aplikacije/prijava korisnika [Izvor: autor]

Klikom na poveznicu "Registracija", korisnik je u mogućnosti ispuniti formu te registrirati se. Forma se sastoji od polja za unos imena, e-mail adrese, lozinke, potvrde lozinke, odabira prava pristupa te gumba za registraciju. Klikom na gumb, generira se provjera o unesenim vrijednostima te ukoliko su podaci ispravno uneseni, korisnik je registriran te ima pristup ostalim dijelovima aplikacije ovisno o roli.

**Programski kod 3.3.** Isječak "register.blade.php" datoteke

```
<div class="panel-heading">Registracija</div>
<div class="panel-body">
    <form class="form-horizontal" role="form" method="POST" action="{{ url('/register') }}">
        {{ csrf_field() }}
        <div class="form-group{{ $errors->has('name') ? ' has-error' : '' }}">
            <label for="name" class="col-md-4 control-label">Ime</label>
            <div class="col-md-6">
                <input id="name" type="text" class="form-control" name="name" value="{{ old('name') }}">
                @if ($errors->has('name'))
                    <span class="help-block">
                        <strong>{{ $errors->first('name') }}</strong>
                    </span>
                @endif
            </div>
        </div>
        <div class="form-group{{ $errors->has('email') ? ' has-error' : '' }}">
            <label for="email" class="col-md-4 control-label">E-Mail adresa</label>
            <div class="col-md-6">
                <input id="email" type="email" class="form-control" name="email" value="{{ old('email') }}">
                @if ($errors->has('email'))
                    <span class="help-block">
                        <strong>{{ $errors->first('email') }}</strong>
```

```
            </span>
        @endif
    </div>
</div>
<div class="form-group{{ $errors->has('password') ? ' has-error' : '' }}">
    <label for="password" class="col-md-4 control-label">Lozinka</label>

    <div class="col-md-6">
        <input id="password" type="password" class="form-control" name="password">

        @if ($errors->has('password'))
            <span class="help-block">
                <strong>{{ $errors->first('password') }}</strong>
            </span>
        @endif
    </div>
</div>
<div class="form-group{{ $errors->has('password_confirmation') ? ' has-error' : '' }}">
    <label for="password-confirm" class="col-md-4 control-label">Potvrdite lozinku</label>
    <div class="col-md-6">
        <input id="password-confirm" type="password" class="form-control" name="password_confirmation">
        @if ($errors->has('password_confirmation'))
            <span class="help-block">
                <strong>{{ $errors->first('password_confirmation') }}</strong>
            </span>
        @endif
    </div>
</div>
<div class="form-group">
    <label for="role" class="col-md-4 control-label">Pravo pristupa</label>
    <div class="col-md-6">
        <select name="role" class="form-control">
            <option value="Administrator">Administrator</option>
            <option value="Korisnik">Korisnik</option>
        </select>
    </div>
</div>
```




**Sl.3.5.** Prikaz forme za registraciju [Izvor: autor]

### 3.2.3. Implementacija modela

Nakon uspješnog kreiranja baze, potrebno je kreirati model s migracijom za kreiranje određene tablice. Kao primjer je naveden model opreme "Equipment.php". Spomenuto kreiranje izvodi se putem terminala uz naredbu "php artisan make: model Equipment -m".

**Programski kod 3.4.** Implementacija "Equipment.php" datoteke

```
<?php

namespace App;
use App\Location;
use Illuminate\Database\Eloquent\Model;

class Equipment extends Model
{
   protected $fillable = [

      'name',
      'serial_number',
      'pn_number',
      'model',
      'comment',
      'location_id',
      'equipment_owner_id'

   ];

   public function location(){
```

```
    return $this->belongsTo('App\Location');
}
```

Popunjava se polje elemenata "fillable" stupcima iz tablice "equipment". Iduća bitna stavka je povezivanje određenih tablica. Navedena funkcija "location" temelji se na 1:N vezi, a implementira se funkcijama "hasMany" ili "belongsTo". Spomenuta relacija koristi se u slučaju kada jedan model ima više drugih modela. Na primjer, određena vrsta opreme se može koristiti na više lokacija.

### 3.2.4. Implementacija upravitelja

Idući dio je kreiranje upravitelja koji se temelji na logici aplikacije. Datoteka "EquipmentsController.php" implementira se naredbom "php artisan make: controller EquipmentsController —resource", dodavanjem parametra "—resource" automatski se implementiraju funkcije potrebne za CRUD (*Create, Read, Update, Delete*) operacije, odnosno operacije za kreiranje, učitavanje, ažuriranje i brisanje podataka iz baze. Funkcija "index" vraća pogled vezan za prikaz opreme, odnosno datoteku "index.blade.php". Budući da će u spomenutom pogledu biti potrebno izlistanje svih podataka vezanih za opremu, podatke je potrebno spremiti u varijablu "equipments" koji će se dohvatiti uz pomoć Laravelove implementirane funkcije "all".

Funkcija "create" dohvaća sve lokacije i vlasnike opreme budući da će podaci iz spomenutih modela biti potrebni u pogledu "create.blade.php", nakon toga potrebno je vratiti, točnije dohvatiti spomenuti pogled vezan za prikaz forme za dodavanje opreme.

Funkcija "store" kao parametar prima instancu "request" pomoću koje se pohranjuju podaci u tablicu te se generira validacija za popunjavanje polja u formi za dodavanje nove opreme. Zatim, uz pomoć Laravelove ugrađene funkcije "create" pohranjuju se podaci u bazu podataka. Idući korak je deklariranje varijable "file" koja sadrži putanju i naziv QR koda, odnosno slike "png" formata. Budući da je implementirana QR biblioteka, idući korak se temelji na dohvaćanju podataka koje će QR kod saržavati (naziv, PN broj, serijski broj i model opreme). Zatim se dodaje vrijednost putanje u bazu te pomoću funkcije "update" ažurira unos podataka. Zadnji korak je vraćanje na početnu stranicu vezanu za opremu.

Funkcija "edit" prima kao parametar "$id" vezan za opremu, pomoću funkcije "findOrFail" dohvaća se traženi "id", odnosno oprema koja se želi urediti te se vraća pogled "edit.blade.php" vezan za prikaz forme uređivanja opreme.

Funkcija "update" prima kao parametar instancu "request" te "id" vezan za opremu, pomoću funkcije "findOrFail" dohvaćamo traženi "id". Nakon što je "id" "dohvaćen", uz funkciju "update", podaci se ažuriraju te se korisnik preusmjerava na početnu stranicu vezanu za opremu.

Posljednja funkcija implementirana u datoteci "EquipmentsController.php" je "destroy" funkcija koja se temelji na brisanju podataka iz tablice. Kao parametar prima "id" te pomoću funkcije "findOrFail" generira se traženi "id". Zatim, poziva se funkcija "delete". Nakon toga, implementiran je "redirect" na početnu stranicu vezan za opremu.

Sljedeći primjer predstavlja isječak koda iz "routes.php" datoteke gdje se zapravo definiraju željene putanje vezane za određene metode, odnosno upravitelje.

**Programski kod 3.5.** Implementacija putanja u "routes.php" datoteci

```
Route::get('/', 'HomeController@index');
Route::resource('/employees', 'EmployeesController');
Route::resource('/equipment', 'EquipmentsController');
Route::resource('/equipmentOwner', 'EquipmentOwnersController');
Route::resource('/locations', 'LocationsController');
```

### 3.2.5. Implementacija pogleda

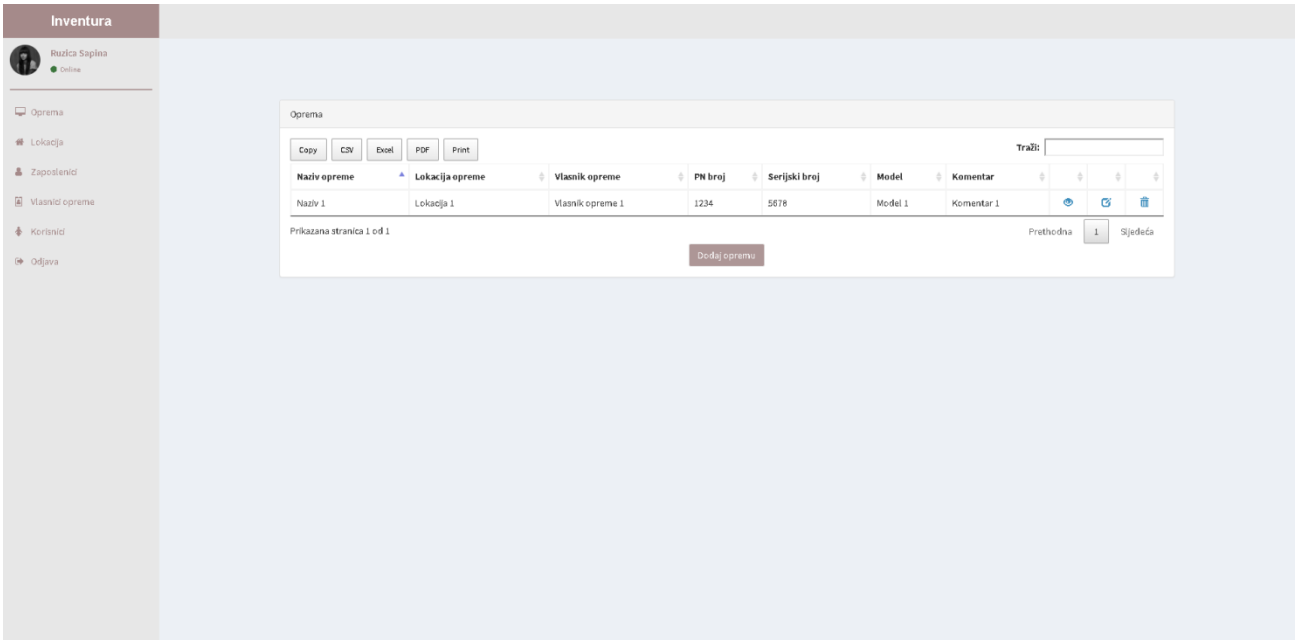
Sljedeći segment aplikacije temelji se na samom prikazu aplikacije u pregledniku. HTML i CSS kod implementira se u *View* dijelu MVC arhitekture. U nastavku, prikazan je kod iz datoteke "index.blade.php" vezan za opremu.

**Programski kod 3.6.** Implementacija "index.blade.php" datoteke

```
@extends ('layouts.admin')

@section('content')
    <div class="col-md-12">
        <div class="box-body">
            <div class="panel panel-default">
                <div class="panel-heading">Oprema</div>
                <div class="panel-body">
                    <table class="table table-bordered table-hover dtable">
                        <thead align="center">
                        <tr>
                            <th>Naziv opreme</th>
                            <th>Lokacija opreme</th>
```

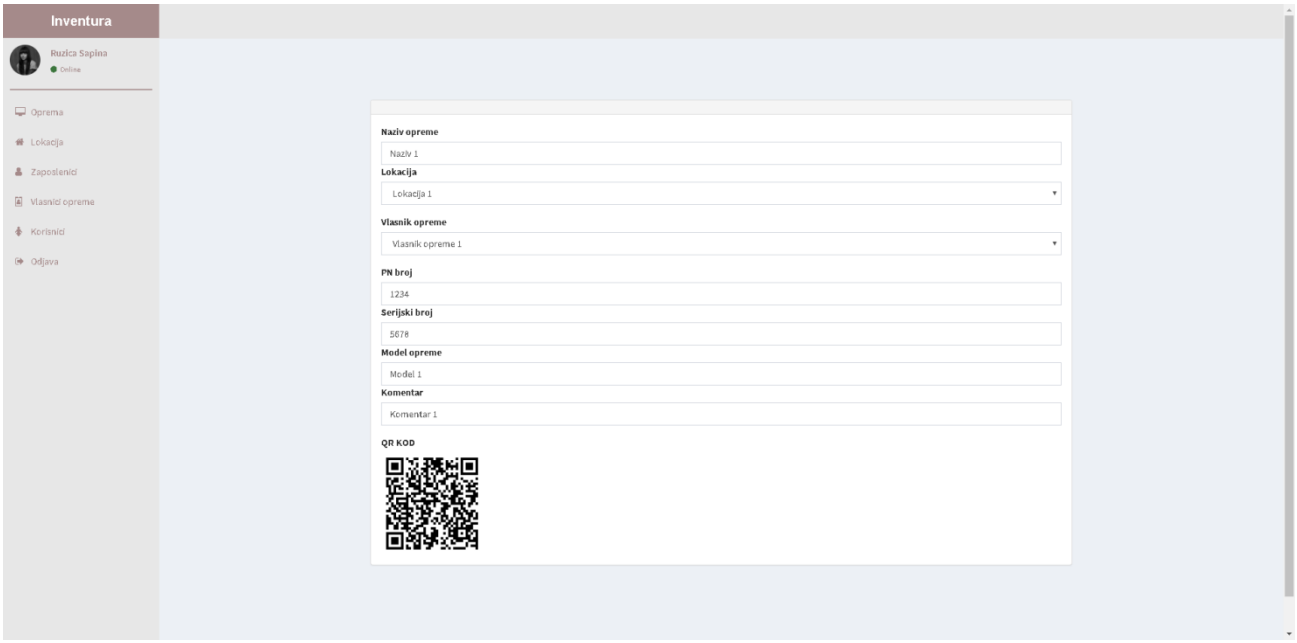
```
                    <th>Vlasnik opreme</th>
                    <th>PN broj</th>
                    <th>Serijski broj</th>
                    <th>Model</th>
                    <th>Komentar</th>
                    @if(Auth::user()->role == "Administrator")
                        <th></th>
                    @endif
                </tr>
                </thead>
                <tbody>
                @foreach($equipments as $equipment)
                    <tr>
                        <td>{{$equipment->name}}</td>
                        <td>{{$equipment->location->name}}</td>
                        <td>{{$equipment->equipmentOwner->name}}</td>
                        <td>{{$equipment->pn_number}}</td>
                        <td>{{$equipment->serial_number}}</td>
                        <td>{{$equipment->model}}</td>
                        <td>{{$equipment->comment}}</td>
                        @if(Auth::user()->role == "Administrator")
                            <td><a href="{{route('equipment.edit', $equipment->id)}}">Uredi</a></td>
                        @endif
                    </tr>
                @endforeach
                </tbody>
            </table>
        </div>
    </div>
</div>
@if(Auth::user()->role == "Administrator")
    <button type="submit" class="btn btn-primary center-block btn-style"><a style="color: white"
                                                                             href="{{route('equipment.create')}}">Dodaj opremu</a></button>
@endif
@stop

@extends('layouts.footer')
```

Iz priloženoga, može se vidjeti da datoteka sadrži "admin.blade.php" datoteku koja će biti detaljnije opisana u jednom od sljedećih poglavlja. Također, korištene su Bootstrap klase implementirane od strane Bootstrap okruženja. Navedeni pogled sadrži tablicu sa stupcima: "Naziv opreme", "Lokacija opreme", "Vlasnik opreme", "PN broj", "Serijski broj", "Model", "Komentar". S "foreach" petljom dohvaćaju se podaci iz tablice te se ispisuju. S "if" funkcijom provjerava se pravo pristupa podacima, ukoliko se radi o roli "administrator", opcija za uređivanje i brisanje opreme te gumb "Dodaj opremu" će biti vidljivi.

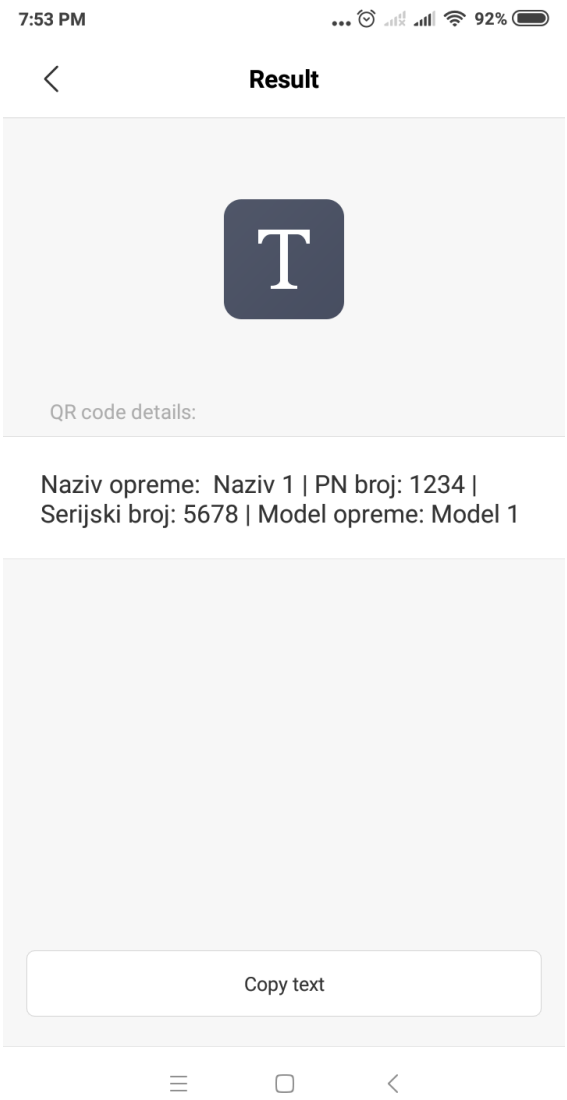



**Sl.3.6.** Prikaz stranice za pregled opreme [Izvor: autor]

Može se primijetiti da je sintaksa spomenutih petlji, formi, dodavanja drugih datoteka i sl. drugačija u odnosu na proceduralni PHP, odnosno HTML. Radi se o Larevelovom Blade template-u. U nastavku, prikazan je kôd iz datoteke "create.blade.php" vezan za opremu. Forma se sastoji od polja za unos naziva opreme, lokacije, vlasnika opreme, PN broja, serijskog broja, modela opreme, komentara te gumba "Dodaj". Pomoću metode "post" i metode "store" iz "EquipmentsController" upravitelja unose se podaci vezane za opremu u formu te spremaju u tablicu "equipment" u bazu podataka.




**Sl.3.7.** Prikaz stranice za pregled pojedine opreme [Izvor: autor]

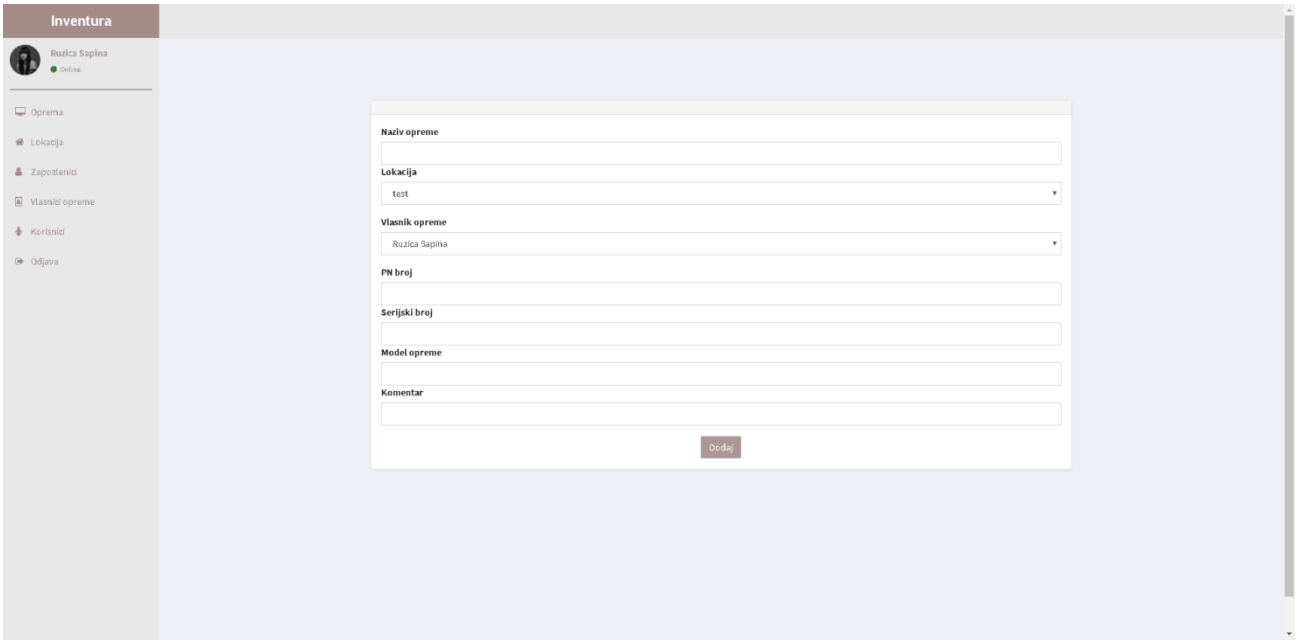



**Sl.3.8.** Prikaz podataka na mobilnom uređaju nakon skeniranja QR koda [Izvor: autor]

**Programski kod 3.7.** Implementacija "create.blade.php" datoteke

```
@extends ('layouts.admin')

@section('content')
    <br>
    <div class="col-md-12">
        <div class="box-body">
            <div class="panel panel-default">
                <div class="panel-heading"></div>
                <div class="panel-body">
                    {!! Form::open(['method'=>'POST', 'action'=>'EquipmentsController@store']) !!}
                    {{csrf_field()}}
                    {!! Form::label('name', 'Naziv opreme') !!}
                    {!! Form::text('name', null, ['class'=>'form-control']) !!}
```

```
            <div class="form-group">
                {!! Form::Label('name', 'Lokacija') !!}
                <select class="form-control" name="location_id">
                    @foreach($locations as $location)
                        <option value="{{$location->id}}">{{$location->name}}</option>
                    @endforeach
                </select>
            </div>
            <div class="form-group">
                {!! Form::Label('name', 'Vlasnik opreme') !!}
                <select class="form-control" name="equipment_owner_id">
                    @foreach($equipmentOwners as $equipmentOwner)
                        <option value="{{$equipmentOwner->id}}">{{$equipmentOwner->name}}</option>
                    @endforeach
                </select>
            </div>
            {!! Form::label('pn_number', 'PN broj') !!}
            {!! Form::text('pn_number', null, ['class'=>'form-control']) !!}
            {!! Form::label('serial_number', 'Serijski broj') !!}
            {!! Form::text('serial_number', null, ['class'=>'form-control']) !!}
            {!! Form::label('model', 'Model opreme') !!}
            {!! Form::text('model', null, ['class'=>'form-control']) !!}
            <div class="form-group">
                {!! Form::label('comment', 'Komentar') !!}
                {!! Form::text('comment', null, ['class'=>'form-control']) !!}
            </div>
            {!! Form::submit('Dodaj', ['class'=>'btn btn-primary center-block btn-style']) !!}
            {!! Form::close() !!}
        </div>
    </div>
</div>
</div>

@endsection('footer')
```



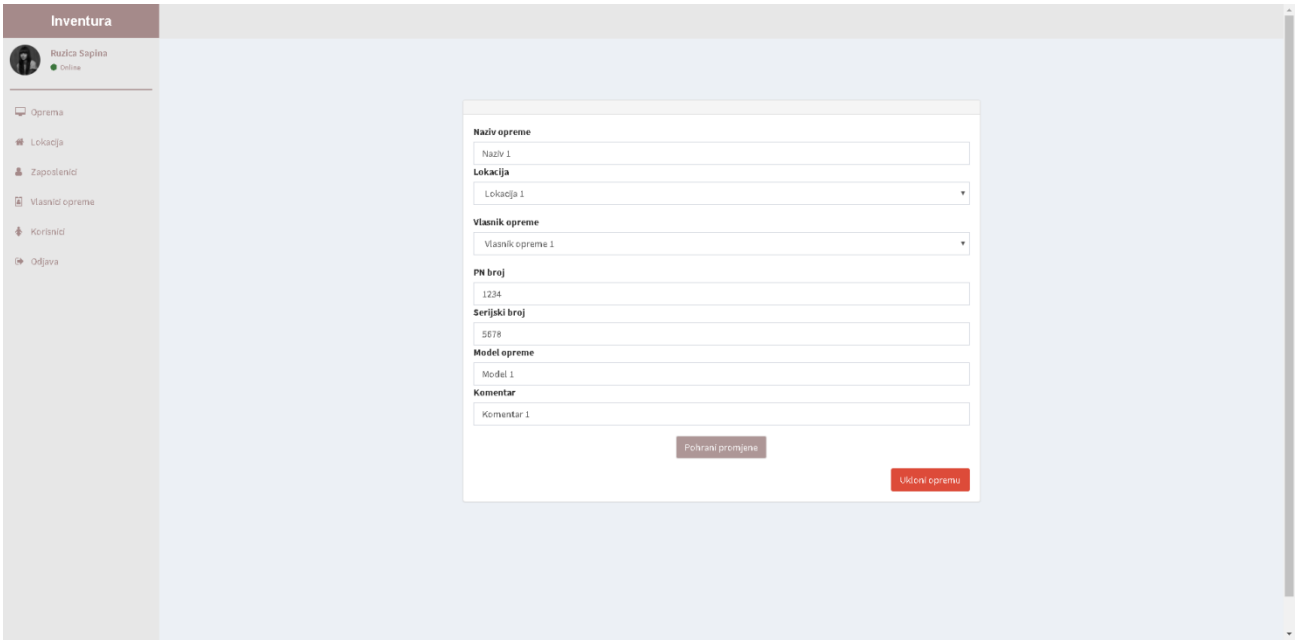

**Sl.3.9.** Prikaz stranice za dodavanje opreme [Izvor: autor]

Nadalje, ukoliko je administrator pritisnuo gumb "Uredi", bit će preusmjeren na datoteku "edit.blade.php" gdje je generirana forma za uređivanje odabrane opreme. U nastavku, prikazan je kod iz datoteke "edit.blade.php" vezan za opremu. Forma funkcionira na način da se pomoću ID-a određene opreme, metode "patch" i ranije spomenute metode "update" iz "EquipmentsController" kontrolera, dohvaćaju svi podaci vezani za opremu te administrator ima mogućnost promijeniti podatke o opremi te klikom na gumb "Pohrani promjene", podaci se ažuriraju u bazi podataka. Nakon ažuriranja podataka, administrator se preusmjerava na početnu stranicu za prikaz opreme. Ukoliko je administrator pritisnuo gumb "Ukloni opremu", pomoću funkcije "destroy" iz spomenutog kontrolera i ID parametra, izbrisat će se određena oprema iz baze podataka te će se administratora preusmjeriti na početnu stranicu.

**Programski kod 3.8.** Implementacija "edit.blade.php" datoteke

```
@extends ('layouts.admin')

@section('content')
    <div class="col-md-12">
        <div class="box-body">
            <div class="panel panel-default">
                <div class="panel-heading"></div>
                <div class="panel-body">
                    {!! Form::model($equipment,['method'=>'PATCH', 'action'=>['EquipmentsController@update', $equipment->id]]) !!}
                    {{csrf_field()}}
```

```
            {!! Form::label('name', 'Naziv opreme') !!}
            {!! Form::text('name', null, ['class'=>'form-control']) !!}
            <div class="form-group">
                {!! Form::Label('name', 'Lokacija') !!}
                <select class="form-control" name="location_id">
                    @foreach($locations as $location)
                        <option value="{{$location->id}}"
                        @if ($equipment->location->name ==$location->name )
                            {{ 'selected' }}
                                @endif
                        >{{$location->name}}</option>
                    @endforeach
                </select>
            </div>
            <div class="form-group">
                {!! Form::Label('name', 'Vlasnik opreme') !!}
                <select class="form-control" name="equipment_owner_id">
                    @foreach($equipmentOwners as $equipmentOwner)
                        <option value="{{$equipmentOwner->id}}"
                        @if ($equipment->equipmentOwner->name ==$equipmentOwner->name )
                            {{ 'selected' }}
                                @endif
                        >{{$equipmentOwner->name}}</option>
                    @endforeach
                </select>
            </div>
            {!! Form::label('pn_number', 'PN broj') !!}
            {!! Form::text('pn_number', null, ['class'=>'form-control']) !!}
            {!! Form::label('serial_number', 'Serijski broj') !!}
            {!! Form::text('serial_number', null, ['class'=>'form-control']) !!}
            {!! Form::label('model', 'Model opreme') !!}
            {!! Form::text('model', null, ['class'=>'form-control']) !!}
            {!! Form::label('comment', 'Komentar') !!}
            {!! Form::text('comment', null, ['class'=>'form-control form-group']) !!}
            {!! Form::submit('Pohrani promjene', ['class'=>'btn btn-primary center-block btn-style']) !!}
            {!! Form::close() !!}
            {!! Form::open(['method'=>'DELETE', 'action'=>['EquipmentsController@destroy', $equipment->id]]) !!}
            {{csrf_field()}}
            {!! Form::submit('Ukloni opremu', ['class'=>'btn btn-danger pull-right']) !!}
            {!! Form::close() !!}
        </div>
    </div>
</div>
</div>

@endsection('footer')
```




**Sl.3.10.** Prikaz stranice za uređivanje opreme [Izvor: autor]

### 3.2.6. Dizajn aplikacije

Važno je spomenuti "admin.blade.php" datoteku koja se nalazi u "layouts" direktoriju. U spomenutoj datoteci su zapravo implementirani svi direktoriji, odnosno datoteke vezane za css, bootstrap, jquery te DataTables dodatak. Spomenuta datoteka predstavlja predložak, točnije dizajn cijele aplikacije tako da je datoteku potrebno implementirati u svaki pogled.

**Programski kod 3.9.** Isječak koda implementacije "DataTables" dodatka

```
<script>
    $(document).ready( function () {
        $.noConflict();
        $('.dtable').DataTable({
            dom:'Bfrtip',
            buttons: [
                'copy', 'csv', 'excel', 'pdf', 'print'
            ],
            "language": {
                "lengthMenu": "Prikaži _MENU_ zapisa po stranici",
                "zeroRecords": "Nema zapisa!",
                "info": "Prikazana stranica _PAGE_ od _PAGES_",
                "infoEmpty": "Nema dostupnih zapisa!",
                "infoFiltered": "(filtered from _MAX_ total records)",
                "search": "Traži:",
                "paginate": {
                    "first": "Prvi",
                    "last": "Zadnji",
                    "next": "Sljedeća",
                    "previous": "Prethodna"
```

```
            }}
        });
    } );

</script>
```

DataTables dodatak korišten je za pregledniji prikaz sadržaja tablice te mogućnosti pretrage, printanja, kopiranja i izvoza sadržaja tablice u željeni format.

**Programski kod 3.10.** Isječak koda "admin.blade.php" vezan za zaglavlje

```
<header class="main-header">
  <a href="index2.html" class="logo">
      <span class="logo-lg"><b>Inventura</b></span>
  </a>
</header>
```

U zaglavlju je implementiran naziv aplikacije, ukoliko korisnik klikne na naziv, bit će preusmjeren na početnu stranicu aplikacije.

**Programski kod 3.11**. Isječak koda "admin.blade.php" vezan za glavni izbornik

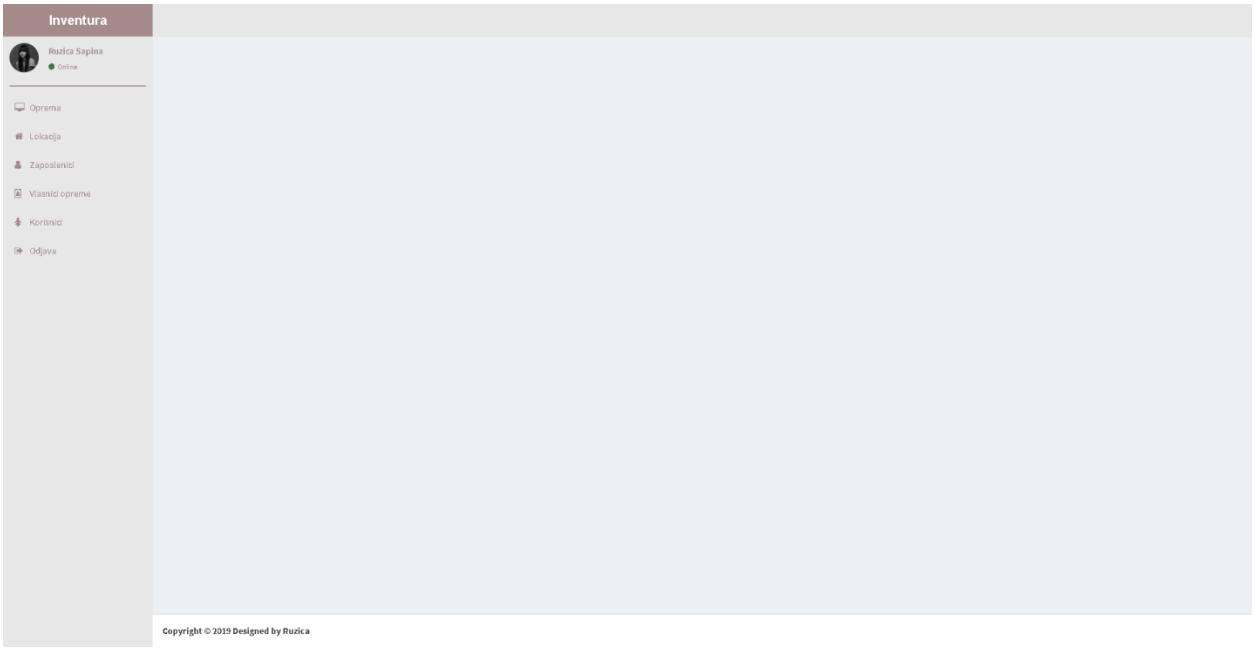
```
<ul class="sidebar-menu" data-widget="tree">
    <li><a href="{{ url('/equipment') }}"><i class="fa fa-desktop"></i> <span>Oprema</span></a></li>

    <li><a  href="{{ url('/locations') }}"><i class="fa fa-home"></i> <span>Lokacija</span></a></li>

    <li><a href="{{ url('/employees') }}"><i class="fa fa-user"></i> <span>Zaposlenici</span></a></li>

    <li><a href="{{ url('/equipmentOwner') }}"><i class="fa fa-id-badge"></i> <span>Vlasnici opreme</span></a></li>

    <li><a href="{{ url('/users') }}"><i class="fa fa-female"></i> <span>Korisnici</span></a></li>

    <li><a href="{{ url('/logout') }}"><i class="fa fa-sign-out"></i> <span>Odjava</span></a></li>
</ul>
```

U "admin.blade.php" datoteci definiran je izbornik koji se referencira na tablice iz baze: Oprema, Lokacija, Zaposlenici, Vlasnici opreme, Korisnici te sadrži mogućnost odjave iz aplikacije.

**Programski kod 3.12.** Isječak koda "admin.blade.php" vezan za prikaz sadržaja tablice

```
<div class="content-wrapper">
    <div class="col-sm-12">
                <section class="content container-fluid">
                            @yield ('content')
                </section>
    </div>
</div>
```

U spomenutoj datoteci nalazi se spomenuta Blade sintaksa, odnosno opcija *@yield* koja usmjerava na *content* dio gdje je prikazan sadržaj određene tablice.




**Sl.3.11.** Prikaz osnovnog dizajna web aplikacije [Izvor: autor]

Osnovna struktura dizajna temelji se na "admin.blade.php" datoteci, a sastoji se od zaglavlja, glavnog izbornika koji se nalazi na lijevoj strani, dijela gdje se prikazuje željena tablica te podnožja.

### 3.2.7. Implementacija Middleware-a

Middleware možemo opisati kao mehanizam koji se bazira na filtriranju http zahtjeva koji se odnose na web aplikaciju. Laravel uključuje middleware koji provjerava autentifikaciju korisnika. Primjerice, ukoliko korisnik nije registriran/prijavljen, bit će preusmjeren na početnu stranicu prijave/registracije korisnika. Međutim, ukoliko je korisnik prijavljen/registriran, middleware će omogućiti korisnikov zahtjev za daljni rad u aplikaciji. Middleware se implementira pomoću naredbe ’’php artisan make:middleware Authenticate’’. Ova naredba će kreirati novu ‘’Authenticate’’ klasu u mapi s putanjom “app/Http/Middleware”.

**Programski kod 3.13.** Datoteka "Authenticate.php"

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\Auth;

class Authenticate
{

    public function handle($request, Closure $next, $guard = null)
    {
        if (Auth::guard($guard)->guest()) {
            if ($request->ajax() || $request->wantsJson()) {
                return response('Unauthorized.', 401);
            } else {
                return redirect()->guest('login');
            }
        }
        return $next($request);
    }
}
```

Nakon kreiranja *middleware*-a, potrebno ga je implementirati u "Kernel.php" datoteku, naziv *middleware*-a koji će se koristiti je "auth", zatim se isti "poziva" u "routes.php" datoteci.

**Programski kod 3.14.** Isječak koda "Kernel.php" datoteke

```php
protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
                                                                                                    …
```

**Programski kod 3.15.** Isječak koda "routes.php" datoteke

```php
Route::get('/locations', 'LocationsController@index')->middleware('auth');

Route::get('/employees', 'EmployeesController@index')->middleware('auth');

Route::get('/equipment', 'EquipmentsController@index')->middleware('auth');

Route::get('/equipmentOwner', 'EquipmentOwnersController@index')->middleware('auth');
```

# 4. ZAKLJUČAK

Primarni cilj, odnosno zadatak završnog rada bio je izrada web aplikacije za inventuru i nadzor opreme koristeći se naprednom tehnologijom kao što je spomenuti PHP programski okvir (engl. *framework*) Laravel. Uz učenje novih tehnologija, cilj je bio proširiti postojeće znanje iz već korištenih alata i tehnologija. Kao sekundarni cilj, omogućeno je korištenje aplikacije u realnom svijetu budući da temelji na jednostavnom unosu i praćenju opreme, lokacije (ureda), zaposlenika i ostalih stavki vezanih za organizaciju rada tvrtke bila bi korisna većini poslovnih jedinica. Upoznata sam s programskim jezikom PHP, te korištenjem objektno-orijentiranog i proceduralnog principa istoga, pa mogu navesti nekoliko prednosti spomenutog programskog okvira: jednostavna implementacija baze podataka te povezivanje sa istom, unaprijed definiran Bootstrap, jednostavna sintaksa, veliki broj implementiranih funkcija te jednostavna dokumentacija.

# LITERATURA

[1] Uvod u PHP, https://php.com.hr/, lipanj 2020.

[2] HTML, http://www.webtech.com.hr/html.php, lipanj 2020.

[3] CSS, http://www.webtech.com.hr/css.php, lipanj 2020.

[4] CSS sintaksa, https://www.mojwebdizajn.net/edukacija/web-dizajn/prezentacije/uvod-u-css.aspx#slide-3 , lipanj 2020.

[5] Bootstrap 4 Get Started, https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp, lipanj 2020.

[6] How files are structured, https://wpamelia.com/what-is-bootstrap/, lipanj 2020.

[7] Uvod u MySQL, https://php.com.hr/66, lipanj 2020.

[8] Where To Start, https://laravel.com/docs/4.2/introduction#where-to-start, lipanj 2020.

# SAŽETAK

U radu je prikazana izrada web aplikacije za inventuru i nadzor opreme. Prikazani su "isječci", odnosno primjeri koda pojedinih datoteka te su popraćeni detaljnijim objašnjenjima. Aplikacija se temelji na prijavi, odnosno registraciji korisnika. Podaci vezani za organizaciju rada tvrtke, spremljeni su u bazu podataka. Aplikacija se može proširiti, odnosno prilagoditi uz implementaciju dodatnih stavki vezanih za pojedinu tvrtku. Također, postoji mogućnost skeniranja QR koda određenih podataka te izvoz, kopiranje i ispis istih.

**Ključne riječi:** inventura, QR skeniranje, web aplikacija

## ABSTRACT

The paper presents a web application development for inventory and equipment organization. The codes shown are used in the production followed by detailed explanations. The application is based on user login and registration. The data related to company organization is linked to database. Application can be expanded with new database parametars related to the company. Also, there is possibility of QR scanning, data export in several formats, data copying and printing.

**Key words:** inventory, QR scanning, web application

# ŽIVOTOPIS

Ružica Šapina rođena je 04.05.1997. godine u Uznach-u (Švicarska). Živi u Piškorevcima na adresi I. Meštrovića 28. Osnovnoškolsko obrazovanje započinje 2004. godine u OŠ Matije Gupca u Piškorevcima. Nakon osnovnoškolskog obrazovanja, 2012. godine upisuje srednju Ekonomsku školu Braće Radić u Đakovu, smjer ekonomist. Nakon srednjoškolskog obrazovanja upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija, stručni studij, smjer Automatika. Nakon završene prve godine stručnog studija, prebacuje se na smjer Informatika. Posjeduje znanje u govoru, čitanju i pisanju engleskog jezika te vozačku dozvolu B kategorije. Osim navedenog posjeduje i osnovno znanje programskih alata, odnosno tehnologija: HTML, CSS, PHP, MySQL, Bootstrap, C, C# i JAVA te upravljanje Microsoft Office alatima.

# PRILOZI

Projektna mapa s izvornim kôdom nalazi se na optičkom disku koji je priložen uz tiskanu verziju završnog rada.