

VISOKA TEHNIČKA ŠKOLA  
STRUKOVNIH STUDIJA  
SUBOTICA

IZNAJMLJIVANJE STANOVA-KUĆA  
projekat  
iz predmeta Web programiranje

Emanuela Bošnjak  
26122094  
(NDA)

MENTOR  
prof. dr Zlatko Čović

SUBOTICA, 2024.

## Sadržaj

1. Opis zadatka	3
Pregled projekta	3
Nivoi pristupa i dozvole	3
2. Realizacija zadatka	5
3. Struktura baze podataka	34
4. Opis funkcionalnosti	35
5. Korišćena literatura	40

## 1. Opis zadatka

### Pregled projekta

Ova dokumentacija opisuje razvoj veb aplikacije namenjene izdavanju stanova i kuća u određenom gradu. Aplikacija razlikuje tri nivoa pristupa: gost, registrovani / prijavljeni korisnik i administrator.

### Nivoi pristupa i dozvole

#### Pristup za goste:

Gosti imaju ograničen pristup funkcijama aplikacije. Oni mogu da:

- Pogledaju opšte informacije dostupne na sajtu
- Pregledaju liste nekretnina koje su dostupne za iznajmljivanje
- Koriste detaljne funkcije pretrage i razne filtere da bi pronašli odgovarajuća svojstva
- Registruje se za nalog na sajtu

#### Pristup registrovanih/prijavljenih korisnika:

Registrovani ili prijavljeni korisnici imaju pristup poboljšanim funkcijama, uključujući mogućnost:

- Da pogledaju detaljne liste nekretnina za stanove i kuće dostupne za iznajmljivanje
- Sprovedu napredne pretrage koristeći filtere kao što su tip nekretnine, cena, lokacija i period iznajmljivanja da bi pronašli odgovarajuće nekretnine
- Ažuriraju informacije o njihovom profilu, uključujući promenu njihove lozinke, imena, prezimena i broja telefona
- Postave oglase za iznajmljivanje nekretnina, uključujući otpremanje fotografija, navođenje lokacije u gradu, pružanje opisa i postavljanje perioda zakupa i cena
- Zatraže resetovanje lozinke ako su zaboravili lozinku
- Kontaktiraju druge korisnike koji su postavili oglase putem e-pošte

#### Administratorski pristup:

Administratori imaju najviši nivo pristupa i kontrole nad aplikacijom. Njihove mogućnosti uključuju:

- Pregledanje, uređivanje i brisanje svih oglasa postavljenih na sajtu
- Odobravanje ili odbijanje oglasa koje su poslali korisnici
- Omogućavanje ili onemogućavanje prikaza određenih reklama
- Davanje ili uskraćivanje mogućnosti korisnicima da se prijave na sistem

### **Registracija korisnika i bezbednost:**

Registracija korisnika je dizajnirana da bude bezbedna. Proces uključuje slanje veze sa ključem za aktiviranje putem e-pošte kako bi se potvrdio identitet korisnika. Ova veza za aktivaciju se takođe koristi kada korisnici zahtevaju promenu lozinke. Sistem osigurava da je svaka adresa e-pošte jedinstvena, sprečavajući da se više naloga registruje sa istom adresom e-pošte.

### **Objavljivanje oglasa i interakcija korisnika:**

Registровани korisnici mogu postavljati oglase za iznajmljivanje nekretnina i kontaktirati druge korisnike koji su postavili oglase. Kada korisnik objavi oglas, on se prvo drži u stanju čekanja dok ga administrator ne odobri. Nakon odobrenja, postaje javno vidljiv na veb stranici. Kada je nekretnina uspešno izdata, korisnik koji je postavio oglas treba da ažurira status nekretnine u „iznajmljen“. Ova promena pokreće automatsko obaveštenje putem e-pošte za korisnika koji je zainteresovan za iznajmljivanje nekretnine. Pored toga, korisnici koji postavljaju oglase imaju mogućnost da vode evidenciju o tome kada i kome je imovina data u zakup, uključujući i dan kada je zakupac primljen.

### **Administratorske mogućnosti i bezbednost:**

Administratori imaju mogućnost upravljanja korisničkim nalozima, uključujući pregled i deblokiranje blokiranih naloga ako je potrebno. Administrativna oblast je obezbeđena korišćenjem PHP sesija kako bi se osiguralo da samo ovlašćeno osoblje može da joj pristupi. Štaviše, sve korisničke lozinke se bezbedno heširaju korišćenjem BCRYPT algoritma radi poboljšanja bezbednosti.

### **Zahtevi za bazu podataka:**

Aplikacija zahteva bazu podataka pod nazivom “real\_estate”. Ova baza podataka treba da sadrži tabele koje podržavaju sve neophodne funkcionalnosti projekta, kao što su upravljanje korisnicima, lista nekretnina, objavljivanje oglasa i administrativne radnje.

### **Web adresa projektnog zadatka:**

<https://nda.stud.vts.su.ac.rs/>

## 2. Realizacija zadatka

Implementacija počinje sa početnom stranicom, koja se sastoji od poruke dobrodošlice i lanca slika na kojoj se prikazuju fotografije nekih nekretnina koje su navedene za iznajmljivanje na veb stranici. Lanac je implementiran pomoću PHP-a i Javascript-a. PHP se koristi za dodavanje slika svojstava, a JS dodaje potrebnu interakciju korisnika kao što je prebacivanje između navedenih fotografija.

Sistem prijave i registracije funkcioniše na sledeći način:

Prvo, korisnik klikne na dugme Login/Register na navigacionoj traci. Ovo otvara iskačući prozor dizajniran pomoću CSS-a i ima izgled kartica za obrasce za prijavu i registraciju. Ispod je PHP kod uključen u rad na podacima za prijavu ili registraciju koje je dao korisnik:

```
<?php
$loginData = $_POST['loginData'] ?? null;
$registerData = $_POST['registerData'] ?? null;
if ($loginData != null || $registerData != null) {
    $response = ["status" => 500, "message" => 'server error'];
    try {
        require_once 'db_config.php';
        if ($loginData != null) {
            $email = $loginData['email'];
            $password = $loginData['password'];
            $sql = 'SELECT * FROM user_accounts WHERE email=:email';
            $query = $dbh->prepare($sql);
            $query->bindParam(':email', $email, PDO::PARAM_STR);
            $query->execute();
            $results = $query->fetchAll(PDO::FETCH_OBJ);
            if (!empty($results)) {
                if (password_verify($password,
$results[0]->password_hash)) {
                    $state = $results[0]->account_state;
                    $response['message'] = $state;
                    if ($state == 'ACTIVE' || $state == 'ADMIN') {
                        session_start();
                        $_SESSION['user_id'] = $results[0]->user_id;
                        $_SESSION['is_admin'] = ($state == 'ADMIN') ?
true : false;
                        $_SESSION['login_time'] = time();
                        $response['status'] = 200;
                        $response['data'] = $results;
                    } else if ($state == 'NOT_VERIFIED' || $state ==
'BLOCKED') {
                        $response['status'] = 403;
                    }
                }
            }
        }
    }
}
```

```

        }
    } else {
        $response['status'] = 403;
        $response['message'] = 'wrong password';
    }
} else {
    $response['status'] = 404;
    $response['message'] = 'email not found';
}
}

if ($registerData != null) {
    $firstName = $registerData['firstName'];
    $lastName = $registerData['lastName'];
    $phone = $registerData['phone'];
    $email = $registerData['email'];
    $password = $registerData['password'];
    $sql = 'INSERT INTO user_accounts
            (email, password_hash, first_name, last_name, phone_number)
VALUES
            (:email, :password_hash, :first_name, :last_name,
            :phone_number)';
    $query = $dbh->prepare($sql);
    $query->bindParam(':email', $email, PDO::PARAM_STR);
    $pwHash = password_hash($password, PASSWORD_BCRYPT);
    $query->bindParam(':password_hash', $pwHash,
    PDO::PARAM_STR_CHAR);
    $query->bindParam(':first_name', $firstName, PDO::PARAM_STR);
    $query->bindParam(':last_name', $lastName, PDO::PARAM_STR);
    $query->bindParam(':phone_number', $phone, PDO::PARAM_INT);
    if ($query->execute()) {
        $response['status'] = 201;
        $response['message'] = 'registered';
        require_once 'send_mail.php';
        send_verify_email('nda@nda.stud.vts.su.ac.rs', "http://" .
        $_SERVER['HTTP_HOST'] . "/verify_account.php?key=$pwHash");
    } else {
        $response['status'] = 400;
        $response['message'] = 'corrupt data';
    }
}
} finally {
    echo json_encode($response);
    die();
}
}
?>
```

Kada korisnik podnese obrazac za prijavu ili registraciju, podaci obrasca se prikupljaju u JS objekat pod nazivom loginData ili registerData koji sadrži odgovarajuće podatke. Za prijavu, od baze podataka se traži heširana lozinka za datu e-poštu. Ako je pronađena, to znači da je korisnik registrovan, a zatim se lozinka proverava pomoću funkcije PHP password\_verification. Ako heš lozinke potvrđuje datu lozinku, proverava se stanje naloga. Ako je stanje ACTIVE ili ADMIN, prijava je uspešna i PHP sesija je pokrenuta, sa korisničkim ID-om i statusom administratora koji se čuvaju kao promenljive sesije. Ako nalog još uvek nije verifikovan, što ga prikazuje stanje NOT\_VERIFIED naloga, prijavljivanje ne uspeva i šalje se odgovarajuća poruka. Ako je stanje naloga BLOCKED, to znači da je nalog blokirao administrator zbog sumnjiće aktivnosti. Sve ovo se radi u okviru bloka pokušaja tako da se sve greške hvataju i vraćaju kao odgovori.

Za registraciju, data e-pošta se proverava radi dupliranja, a zatim se podaci ubacuju sa stanjem naloga koji je podrazumevano NOT\_VERIFIED. Nakon što se podaci unesu, na novoregistrovanu adresu e-pošte se šalje email sa vezom za verifikaciju naloga. E-pošta je napravljena kao HTML stranica sa čuvarima mesta za postavljanje URL-a za verifikaciju. PHP funkcija string\_replace se koristi za zamenu teksta čuvara mesta iz šablonu e-pošte. PHP kod za slanje verifikacionog koda je prikazan ispod:

```
<?php

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

require_once 'PHPMailer/Exception.php';
require_once 'PHPMailer/PHPMailer.php';
require_once 'PHPMailer/SMTP.php';

function send_verify_email(string $recp_email, string $verify_url): void
{
    $mail = new PHPMailer;
    $mail->isSMTP();
    $mail->SMTPDebug = 0;
    $mail->Host = 'ikki.vts.su.ac.rs';
    $mail->Port = 587;
    $mail->SMTPSecure = 'tls';
    $mail->Username = 'nda';
    $mail->Password = 'BesZKhwcizViY7s';
    $mail->setFrom('nda@nda.stud.vts.su.ac.rs', 'Subotica Rentals');
    $mail->addAddress($recp_email, '');

    $mail->Subject = "Verify your Subotica Rentals Account";
    $mail_template =
        file_get_contents('mail_templates/verify_account.html');
    $mail_html = str_replace(':VERIFY_URL', $verify_url, $mail_template);
    $mail->msgHTML($mail_html);
    $mail->AltBody = strip_tags($mail_html);

    if (!$mail->send()) {
        echo "Mailer Error: " . $mail->ErrorInfo;
    }
}
```

```
    }
}

...

```

Isti fajl sadrži kod za slanje e-pošte za druge prilike, na primer kada korisnik izrazi interesovanje za iznajmljivanje nekretnine, i kada neko bude prihvaćen kao zakupac nekretnine od strane vlasnika.

Verifikacija obrasca za registraciju se vrši kao u sledećem isečku koda:

```
$('#register-form').submit((e) => {
    e.preventDefault();
    registerData = {
        firstName: $('#first-name').val(),
        lastName: $('#last-name').val(),
        phone: $('#phone').val(),
        email: $('#email').val(),
        password: $('#password').val(),
        cnfPassword: $('#cnf-password').val(),
    };
    console.log(registerData);
    let isValid = true;
    if (registerData.firstName.length == 0) {
        isValid = false;
        $('#first-name-error').text('Please enter your first name.');
    } else $('#first-name-error').text('');
    if (registerData.lastName.length == 0) {
        isValid = false;
        $('#last-name-error').text('Please enter your last name.');
    } else $('#last-name-error').text('');
    if (registerData.phone.toString().length != 9) {
        isValid = false;
        $('#phone-error').text('Please enter a valid phone number.');
    } else $('#phone-error').text('');
    if (registerData.email.length == 0) {
        isValid = false;
        $('#email-error').text('Please enter an email address.');
    } else if (!validateEmail(registerData.email)) {
        isValid = false;
        $('#email-error').text('Please enter a valid email address.');
    } else $('#email-error').text('');
    if (registerData.password.length < 8) {
        isValid = false;
        $('#password-error').text('Please enter a valid password, must be atleast 8 chars long.');
    } else $('#password-error').text('');
    if (registerData.cnfPassword != registerData.password) {
        isValid = false;
        $('#cnf-password-error').text('Passwords do not match.');
    } else $('#cnf-password-error').text('');
    if (isValid) {
```

```

        console.log('register form is valid');
        $.ajax({
            type: "POST",
            url: "login_popup.php",
            data: { registerData: registerData },
            dataType: 'json',
        }).done(function (response) {
            console.log(response);
            if (response.status == 201) {
                $('#register-result').text('Registration successful!
Please check your email for a link to verify your account.');
                $('#register-result').css('display', 'unset');
                $('#register-result').css('color', 'green');
            } else {
                $('#register-result').text('Invalid data entered, please
revise and try again!');
                $('#register-result').css('display', 'unset');
                $('#register-result').css('color', 'red');
            }
        });
    });
});

```

Svaki unos polja se prikuplja u registerData JS objekat. Nakon toga sledi provera da li svako polje ima validne informacije. Ako se pronađe nevažeći unos, odgovarajuća poruka se prikazuje ispod svakog polja. Ako su sva polja ispravna, AJAX se koristi za slanje objekta putem POST zahteva na istu stranicu. Nakon uspešne registracije, rezultat se prikazuje.

Administratori imaju pristup listi svih korisnika i mogu blokirati/deblokirati sve korisnike koji nisu administratori. Lista prikazuje sve korisnike osim trenutno prijavljenog korisnika. PHP kod za blokiranje i deblokiranje korisnika je dat u nastavku:

```

<?php
$formData = $_POST['formData'] ?? null;
if ($formData != null) {
    $response = ["status" => 500, "message" => 'server error'];
    ob_start();
    try {
        $task = $formData['task'];
        require_once 'db_config.php';
        if ($task == 'unblock' || $task == 'block') {
            $query = $dbh->prepare(
                "SELECT * FROM user_accounts
                 WHERE user_id = :user_id AND account_state = 'ADMIN'"
            );
            $query->bindParam(':user_id', $formData['userId'],
PDO::PARAM_INT);
            $query->execute();
            if ($query->rowCount() > 0) {

```

```

        $status = ($task == 'unblock' ? 'ACTIVE' : 'BLOCKED');
        $query = $dbh->prepare(
            "UPDATE user_accounts SET
                account_state = '$status' WHERE user_id = :user_id
                AND account_state NOT IN ('ADMIN', 'NOT_VERIFIED')"
        );
        $query->bindParam(':user_id', $formData['modUserId'],
    PDO::PARAM_INT);
        $query->execute();
        if ($query->rowCount() > 0) {
            $response['status'] = 200;
            $response['message'] = 'updated';
        } else {
            $response['status'] = 400;
            $response['message'] = 'bad data';
        }
    } else {
        $response['status'] = 403;
        $response['message'] = 'unauthorized';
    }
} else {
    $response['status'] = 400;
    $response['message'] = 'invalid data';
}
} catch (Exception $e) {
    echo $e;
} finally {
    $response['echo'] = ob_get_contents();
    ob_end_clean();
    echo json_encode($response);
    die();
}
}

session_start();

$loggedIn = isset($_SESSION['user_id']);
if (!$loggedIn || !$_SESSION['is_admin']) {
    header('Location:index.php');
    die();
}

require_once 'db_config.php';
$userId = $_SESSION['user_id'];
$query = $dbh->prepare(
    "SELECT
        *
    FROM
        user_accounts
    WHERE
        user_id <> $userId"

```

```

);
$query->execute();
$users = $query->fetchAll(PDO::FETCH_OBJ);
?>

<!doctype html>
<html lang="en">
...

```

Dati formData ima ciljnu akciju za uključen ID korisnika. Radnja se sprovodi samo ako dotični nalog nije administratorski nalog i ne čeka verifikaciju putem e-pošte. Ako je nalog koji čeka verifikaciju blokiran, onda njegovo deblokiranje dovodi do toga da nalog bude označen kao ACTIVE, pa nije dozvoljen. HTML kod za prikaz liste korisnika je dat u nastavku:

```

<div class="users-list">
    <?php
        if (empty($users)) {
            echo '<span id="no-users-text">No other users
registered!</span>';
        } else {
            foreach ($users as $u) {
                $state = $u->account_state;
                $actionButton = (($state == 'NOT_VERIFIED' || $state
== 'ADMIN') ? ''
                    : ($state != 'ACTIVE'
                        ? "<a href='#!' id='$u->user_id-active'
data-mod-user-id='$u->user_id' class='button-primary button-sm'><i
class='bi bi-check2-all'></i> Unblock</a>"
                        : "<a href='#!' id='$u->user_id-block'
data-mod-user-id='$u->user_id' class='button-primary button-sm
button-delete'><i class='bi bi-ban'></i> Block</a>"));
                echo
                    "<div class='item'>
                        <span class='userId'>User ID: $u->user_id</span>
                        <span class='name'>$u->first_name
$u->last_name</span>
                        <a class='email'
                            href='mailto:$u->email'>$u->email</a>
                        <a class='phone'
                            href='tel:+381$u->phone_number'>+381 $u->phone_number</a>
                            <span class='btn-bar'>
                                <span class='account-state
$u->account_state'></span>
                                $actionButton
                            </span>
                        </div>";
            }
        }
    ?>
</div>

```

Lista prikazuje dugme akcije za blokiranje/deblokiranje samo ako je nalog ACTIVE ili BLOCKED. Svaka stavka liste sadrži ID korisnika, ime, adresu e-pošte i telefonski broj korisnika, zajedno sa stanjem naloga. Stanje naloga je obojeno i sadržaj je podešen pomoću CSS-a:

```
.account-state {  
    font-size: x-large;  
    font-weight: bold;  
}  
  
.NOT_VERIFIED {  
    color: rgb(161, 161, 0);  
}  
  
.NOT_VERIFIED::before {  
    content: "AWAITING EMAIL VERIFICATION";  
}  
  
.ACTIVE {  
    color: rgb(142, 211, 40);  
}  
  
.ACTIVE::before {  
    content: "ACCOUNT ACTIVE";  
}  
  
.ADMIN {  
    color: yellowgreen;  
}  
  
.ADMIN::before {  
    content: "ADMIN";  
}  
  
.BLOCKED {  
    color: rgb(187, 0, 0);  
}  
  
.BLOCKED::before {  
    content: "BLOCKED BY ADMIN";  
}
```

JS kod za rukovanje radnjama dugmeta je sledeći:

```
$(".users-list>.item>.btn-bar>a[id$='active']").click(function (e) {  
    e.preventDefault();  
    let id = e.target.id;  
    if (id != '') {  
        let modUserId = $(`#${id}`).data('mod-user-id');  
        if (confirm('Are you sure you want to unblock this account?')) {  
            console.log('gonna unblock ' + modUserId);  
        }  
    }  
}
```

```

        let formData = {
            userId: userId,
            modUserId: modUserId,
            task: 'unblock',
        };
        $.ajax({
            type: "POST",
            url: "admin_users.php",
            data: {
                formData: formData
            },
            dataType: 'json',
        }).done(function (response) {
            console.log(response);
            if (response.status == 200) {
                alert('Account un-blocked successfully!');
                window.location.reload();
            } else {
                alert(`Unknown error occurred, please try again after
some time. ${response.message}`);
            }
        });
    }
});

$(".users-list>.item>.btn-bar>a[id$='block']").click(function (e) {
    e.preventDefault();
    let id = e.target.id;
    if (id != '') {
        let modUserId = `#${id}`).data('mod-user-id');
        if (confirm('Are you sure you want to block this account?')) {
            console.log('gonna block ' + modUserId);
            let formData = {
                userId: userId,
                modUserId: modUserId,
                task: 'block',
            };
            $.ajax({
                type: "POST",
                url: "admin_users.php",
                data: {
                    formData: formData
                },
                dataType: 'json',
            }).done(function (response) {
                console.log(response);
                if (response.status == 200) {
                    alert('Account blocked successfully!');
                    window.location.reload();
                } else {

```

```

        alert(`Unknown error occurred, please try again after
some time. ${response.message}`);
    }
}
}
});

```

jQuery se koristi za dodavanje slušatelja za svako dugme za blokiranje/deblokiranje korišćenjem regularnih izraza u izrazu selektora. Ovaj metod se takođe koristi na drugim stranicama koje imaju dugmad na stavkama liste. HTML5 atributi podataka se takođe koriste za čuvanje primarnog ključa svake korisničke stavke na listi.

Kada se registruju, korisnici mogu nastaviti da dodaju svojstva pod svojim nalogom. Ovo se radi na sličan način, kao što je prikazano u donjem PHP kodu:

```

<?php
$formData = $_POST['formData'] ?? null;
if ($formData != null) {
    $response = ["status" => 500, "message" => 'server error'];
    ob_start();
    try {
        require_once 'db_config.php';
        $sql = "INSERT INTO properties
(user_id, `name`, type_id, location_id, street_address, capacity,
parking, pets_allowed, `description`)
VALUES
(:user_id, :name, :type_id, :location_id, :street_address,
:capacity, :parking, :pets_allowed, :description)";
        $query = $dbh->prepare($sql);
        $query->bindParam(':user_id', $formData['userId'],
PDO::PARAM_INT);
        $query->bindParam(':name', $formData['propName'],
PDO::PARAM_STR);
        $query->bindParam(':type_id', $formData['propType'],
PDO::PARAM_INT);
        $query->bindParam(':location_id', $formData['locationId'],
PDO::PARAM_INT);
        $query->bindParam(':street_address', $formData['streetAddr'],
PDO::PARAM_STR);
        $query->bindParam(':capacity', $formData['capacity'],
PDO::PARAM_INT);
        $query->bindParam(':parking', $formData['parking'],
PDO::PARAM_BOOL);
        $query->bindParam(':pets_allowed', $formData['pets'],
PDO::PARAM_BOOL);
        $query->bindParam(':description', $formData['description'],
PDO::PARAM_STR);
        if ($query->execute()) {
            $lastId = $dbh->lastInsertId();

```

```

        $photosJsonStr = json_encode($formData['photos']);
        file_put_contents("uploads/$lastId.json", $photosJsonStr);
        $response['status'] = 201;
        $response['message'] = 'created';
    } else {
        $response['status'] = 400;
        $response['message'] = 'bad data';
    }
} catch (Exception $e) {
    echo $e;
} finally {
    $response['echo'] = ob_get_contents();
    ob_end_clean();
    echo json_encode($response);
    die();
}
}

session_start();

$loggedIn = isset($_SESSION['user_id']);
if (!$loggedIn) {
    header('Location:index.php');
    die();
}

$returnUrl = $_GET['return-url'] ?? "index.php";

require_once 'db_config.php';
$query = $dbh->prepare("SELECT * FROM property_types");
$query->execute();
$propertyTypes = $query->fetchAll(PDO::FETCH_OBJ);
$query = $dbh->prepare("SELECT * FROM locations ORDER BY location_name
ASC");
$query->execute();
$locations = $query->fetchAll(PDO::FETCH_OBJ);
?>

<!doctype html>
<html lang="en">
...

```

Podaci za svojstvo se pakuju u JS objekat i šalju na stranicu putem POST zahteva. Fotografije imovine se šalju kao base64 stringovi i čuvaju se u JSON listi u datoteci na serveru. Ime datoteke je primarni ključ novoumetnutog svojstva. Ovo osigurava da nema kolizija u imenima ovih datoteka.

Ako je stranica učitana putem GET zahteva, bez podataka obrasca, proverava se promenljiva sesije koja sadrži ID korisnika, a ako nije podešena, korisnik se vraća na dostavljeni povratni URL ili indeksnu stranicu, ako nije navedeno. Ovo osigurava da nijedan neovlašćeni korisnik ne može da vidi stranicu.

Ako je korisnik prijavljen, u bazi podataka se postavlja upit za sve tipove svojstava i lokacije tako da korisnik može da bira između dostupnih opcija.

HTML kod za obrazac je sledeći:

```
...
<form id="prop-form">
    <input id="user-id" value=<?php echo $_SESSION['user_id'];
?>" name="userId" style="display: none;">
    <label for="prop-name">Name of Property</label>
    <input id="prop-name" name="prop-name">
    <span class="input-error" id="prop-name-error"></span>
    <label for="prop-type">Type of Property</label>
    <select name="prop-type" id="prop-type">
        <option value="" hidden>Select a Type</option>
        <?php
        foreach ($propertyTypes as $type) {
            echo "<option value='".$type->type_id'
title='".$type->description'>
                $type->type_name
            </option>";
        }
        ?>
    </select>
    <span class="input-error" id="prop-type-error"></span>
    <label for="location-id">Location in Subotica</label>
    <input value="-1" id="location-id" name="location-id"
style="display: none;">
    <div class="location-selector">
        <?php
        if (empty($locations)) {
            echo '<span id="no-loc-text">No Locations
Available!</span>';
        } else {
            foreach ($locations as $loc) {
                echo "<div class='item'
id='loc_".$loc->location_id'>
                    $loc->iframe
                </div>";
            }
        }
        ?>
    </div>
    <span class="input-error" id="location-id-error"></span>
    <label for="street-addr">Street Address</label>
    <textarea id="street-addr" name="street-addr"
rows="3"></textarea>
    <span class="input-error" id="street-addr-error"></span>
    <label for="capacity">Capacity (Approx. No. of
Residents)</label>
```

```

<input id="capacity" value=1 name="capacity" type="number">
<span class="input-error" id="capacity-error"></span>
<div class="checkbox-group">
    <input type="checkbox" id="parking" name="parking">
    <label for="parking">Parking Available?</label>
</div>
<div class="checkbox-group">
    <input type="checkbox" id="pets" name="pets">
    <label for="pets">Pets Allowed?</label>
</div>
<div style="display: flex; justify-content:space-between; flex-wrap:wrap;">
    <label for="photos">Add Photos (min. 3, max. 10)</label>
    <input multiple type="file" id="add-photo-input" accept="image/*" style="display: none;">
        <a id="add-photo-btn" class="button-primary button-sm" href="#"><i class="bi bi-upload"></i> Add Photo</a>
    </div>
    <input id="photos" name="photos" style="display: none;">
    <div class="photo-uploader">
        <span id="no-photo-text">Please add at least 3 photos</span>
    </div>
    <span class="input-error" id="photos-error"></span>
    <label for="description">Description</label>
    <textarea id="description" name="description" rows="3"></textarea>
    <span class="input-error" id="description-error"></span>
</form>
...

```

Lokacije se biraju pomoću fleksibilnog diva koji se pomera horizontalno pod nazivom „location selector“ koji prikazuje ugradene Google mape svake lokacije i korisnik može da izabere bilo koju od njih. Tip svojstva se bira preko selektovanog elementa, čije opcije se popunjavaju preko PHP-a. Otpremljene slike se takođe pregledaju u fleksibilnom divu koji se pomera horizontalno, koji se zove “photo uploader”. U početku se prikazuje poruka da moraju biti najmanje 3 slike. Korisnik mora da klikne dugme iznad flex div-a da bi otvorio dijalog za biranje datoteka. Ovo se rešava preko sledećeg JS koda:

```

var uploadedPhotos = new Map();

$( '#add-photo-btn' ).click( function ( event ) {
    $( '#add-photo-input' ).click();
    event.preventDefault();
});

function addImagePreview(key, base64) {
    let item = $(<div>');
    item.addClass('item');
    let image = $(<img>');

```

```

        image.attr('src', base64);
        item.append(image);
        let removeBtn = $('');
        removeBtn.attr('href', '#');
        removeBtn.attr('tooltip', 'Remove Photo');
        removeBtn.html('<i class="bi bi-x-square-fill"></i>');
        removeBtn.addClass('remove-btn');
        removeBtn.click((event) => {
            item.remove();
            uploadedPhotos.delete(key);
            if (uploadedPhotos.size == 0) {
                $('#no-photo-text').css('display', 'unset');
            }
            if (!isValid && uploadedPhotos.size < 3) {
                $('#photos-error').text("Please select at least 3 images.");
            }
        });
        item.append(removeBtn);
        $('.photo-uploader').append(item);
    }

$('#add-photo-input').on('input', function (event) {
    if (event.target.files.length > 0) {
        let fileCount = uploadedPhotos.size;
        for (let i = 0; i < event.target.files.length; i++) {
            let file = event.target.files[i];
            if (fileCount >= 10) {
                alert('Sorry, you can only upload 10 or less images.
Please remove a previously added image to proceed.');
                return;
            }
            if (uploadedPhotos.size == 0) {
                $('#no-photo-text').css('display', 'none');
            }
            let reader = new FileReader();
            reader.onloadend = () => {
                let base64 = reader.result;
                let key = Date.now();
                uploadedPhotos.set(key, base64);
                addImagePreview(key, base64);
                if (uploadedPhotos.size >= 3) {
                    $('#photos-error').text("");
                }
            };
            reader.readAsDataURL(file);
            fileCount++;
        }
    }
});
```

Kada korisnik klikne na dugme za dodavanje fotografije, pokreće se unos datoteke. Ulaz omogućava odabir više slika odjednom. Brojač osigurava da se ne doda više od 10 slika za svojstvo. Svaka slika se konvertuje u svoju base64 notaciju i čuva na mapi uploadedPhotos, sa vremenskom oznakom kao ključem. Kada se umetne u mapu, pregled fotografije se dodaje u div "photo uploader". Ovo se radi tako što se kreira novi DOM čvor i dodaje se kao dete div. Pregled takođe ima dugme za uklanjanje fotografije sa liste. Ovo briše sliku sa mape otpremljenih fotografija pomoću datog ključa, a zatim uklanja element pregleda iz DOM-a.

Zatim se govori o implementaciji stranice za objavljivanje oglasa. Osnovna struktura je slična stranici Add Property, ali posebna karakteristika je „property selector“. Ovo je horizontalno pomerajući flex div sličan biraču lokacije, koji navodi postojeća svojstva koja je dodao korisnik i takođe ima dugme za dodavanje novog svojstva ako je potrebno, što vodi do stranice Add property. Selektor takođe proverava da li je svojstvo već postavljeno u drugom oglasu i ako jeste, stavka koja odgovara toj osobini je onemogućena. HTML-PHP kod za prikaz birača svojstava je sledeći:

```
<div class="property-selector">
<?php
if (empty($properties)) {
    echo '<span id="no-prop-text">No Properties added yet!</span>';
} else {
    foreach ($properties as $prop) {
        $allJson = file_get_contents("uploads/$prop->id.json");
        $photo = json_decode($allJson)[0];
        $query = $dbh->prepare("SELECT * FROM advertisements WHERE
property_id = :prop_id;");
        $query->bindParam(':prop_id', $prop->id, PDO::PARAM_INT);
        $query->execute();
        $posted = ($query->rowCount() > 0 ? ' posted' : '');
        echo
            "<div id='prop_$prop->id' class='item$posted'>
                <img src='$photo'>
                <span class='name'>$prop->name</span>
                <span class='type'>$prop->type</span>
                <span class='location'>$prop->location</span>
            </div>";
    }
}
?>
</div>
```

JSON lista koja sadrži base64 kodiranja fotografija svojstava se čita sa diska i prva slika se prikazuje na listi. Ako je svojstvo već negde objavljen, dodaje se postavljena CSS klasa, koja onemogućava interakciju korisnika na tom svojstvu i zamagljuje sadržaj, kao što se vidi u CSS kodu ispod:

```
.item.posted::after {
    content: "POSTED";
    font-size: xx-large;
```

```

    position: absolute;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
}

.item.posted {
    pointer-events: none;

    img,
    .name,
    .type,
    .location {
        filter: blur(2px);
    }
}

```

JS kod za rukovanje selektorom svojstava je prikazan ispod:

```

$('.property-selector>.item').click(function (event) {
    let prop_id = event.target.id;
    let property_id = (prop_id.split('_'))[1];
    let propertyField = $('#property-id');
    if (propertyField.val() != -1) {
        `#${prop_id}`).removeClass('selected');
    }
    `#${prop_id}`).addClass('selected');
    propertyField.val(property_id);
    $('#property-id-error').text("");
});

```

Primarni ključ svojstva se izdvaja iz HTML id atributa stavke na koju se klikne, a zatim postavlja kao vrednost polja za unos property\_id. Ako je prethodno izabrano neko drugo svojstvo, izabrana klasa se uklanja i dodaje novoizabranoj stavci.

Korisnik može da izabere dugme Objavi da doda podatke o oglasu u bazu podataka i da ih označi kao PENDING. Alternativno, korisnik može da klikne na Save draft da bi ubacio te podatke u bazu podataka sa statusom DRAFT. Radne verzije ne mogu da uređuju administratori, već su navedene i vidljive samo korisniku i administratorima.

Kada je oglas označen kao PENDING, on je naveden na stranici Pending Advertisements u fazi administratora. Oglas je javno vidljiv kada administrator klikne na dugme Approve na njemu. Alternativno, može se blokirati klikom na dugme BLOCK. Lista prikazuje najvažnije informacije o oglasu zajedno sa fotografijama. HTML-PHP kod za listu oglasa je sledeći:

```

<div class="advert-list">
    <?php
    if (empty($adverts)) {
        echo '<span id="no-ads-text">All caught up!</span>';
    } else {
        foreach ($adverts as $ad) {
            $photos =
        json_decode(file_get_contents("uploads/$ad->property_id.json"));
    }

```

```

$photosStr = "";
$first = true;
foreach ($photos as $photo) {
    if ($first) {
        $photosStr .= "<img class='active' src='$photo'>" .
PHP_EOL;
        $first = false;
    } else {
        $photosStr .= "<img src='$photo'>" . PHP_EOL;
    }
}
echo
"<div class='item'>
    <div class='photo-reel'>
        $photosStr
        <a href='#' class='prev-btn'><i class='bi bi-arrow-left'></i></a>
        <a href='#' class='next-btn'><i class='bi bi-arrow-right'></i></a>
    </div>
    <div class='info'>
        <span class='name'>$ad->name</span>
        <span class='type'>$ad->type</span>
        <span class='location'>$ad->location</span>
        <span class='street-addr'>$ad->addr</span>
        <span class='specs'>
            <span class='capacity' title='Capacity'><i class='bi bi-people-fill'></i> $ad->capacity</span>
            <span class='parking' title='Parking'><i class='bi bi-p-square-fill'></i> " . ($ad->parking == 1 ? 'Yes' : 'No') .
        </span>
            <span class='pets' title='Pets'>🐱 . ($ad->pets ==
        == 1 ? 'Yes' : 'No') . "</span>
        </span>
        <span class='price-period'>RSD $ad->price p/m, " .
($ad->period == 0 ? 'Flexible Period' : "$ad->period Months") . "</span>
        <span class='description'>$ad->description</span>
        <span class='status'>$ad->status'</span>
        <span class='btn-bar'>
            <a href='#' id='$ad->id-view'
data-ad-id='$ad->id' class='button-primary button-sm'><i class='bi bi-eye-fill'></i> View</a>
            <a href='#' id='$ad->id-approve'
data-ad-id='$ad->id' class='button-primary button-sm'><i class='bi bi-check2-all'></i> Approve</a>
            <a href='#' id='$ad->id-block'
data-ad-id='$ad->id' class='button-primary button-sm button-delete'><i class='bi bi-ban'></i> Block</a>
        </span>
    </div>
</div>";

```

```

        }
    }
    ?>
</div>
```

Fotografije imovine navedene u oglasu su prikazane u div-i sa jednom slikom koja je vidljiva istovremeno, a fotografije se mogu kretati korišćenjem sledećeg prethodnog dugmeta. JS kod za rukovanje ovim je dat na sledeći način:

```

$('.photo-reel>.prev-btn').click(function (e) {
    let subjects = e.target.parentNode.childNodes;
    let photos = [], activeIdx = -1;
    for (let i = 0; i < subjects.length; i++) {
        let obj = $(subjects[i]);
        if (obj.is('img')) {
            photos.push(obj);
            if (obj.hasClass('active')) {
                activeIdx = photos.length - 1;
            }
        }
    }
    photos[activeIdx].removeClass('active');
    activeIdx = (activeIdx - 1 >= 0) ? (activeIdx - 1) : (photos.length - 1);
    photos[activeIdx].addClass('active');
});

$('.photo-reel>.next-btn').click(function (e) {
    let subjects = e.target.parentNode.childNodes;
    let photos = [], activeIdx = -1;
    for (let i = 0; i < subjects.length; i++) {
        let obj = $(subjects[i]);
        if (obj.is('img')) {
            photos.push(obj);
            if (obj.hasClass('active')) {
                activeIdx = photos.length - 1;
            }
        }
    }
    photos[activeIdx].removeClass('active');
    activeIdx = (activeIdx + 1) % photos.length;
    photos[activeIdx].addClass('active');
});
```

Kada se klikne na bilo koje od sledećeg ili prethodnog dugmeta, prikupljaju se sibilings dugmeta koji su slike i index trenutno vidljive slike se čuva. Zatim se ovaj index koristi za onemogućavanje trenutno aktivne slike i dodavanje aktivne klase novom index-u. Slikama se može kretati kružno, tako da se prva slika prikazuje nakon što se klikne na sledeću ,na poslednju sliku i obrnuto.

Svaka stavka na listi takođe ima dugme za pregled svih detalja o oglasu, što uključuje i sve detalje o navedenoj imovini. Ova opcija je dostupna svim stranicama koje navode oglase, kao što je stranica My Advertisements, koja prikazuje oglase koje je dodao korisnik, i stranica Iznajmljivanja, koja prikazuje javno vidljive reklame. Za korisnike gostiju, klikom na dugme Prikaži oglas otvara se iskačući prozor za prijavu jer su detalji vidljivi samo registrovanim korisnicima.

Slično kao na gornjoj stranici, stranica All Advertisements za admin fazu prikazuje sve oglase, uključujući PUBLIC i BLOCKED. Ova stranica omogućava administratorima da deblokiraju reklame ako je potrebno. Ovde, administratori takođe imaju opciju da uređuju sve oglase koji nisu u fazi nacrt. Međutim, administratori mogu da izbrišu bilo koju i sve reklame korišćenjem dugmeta Delete na svakoj stavci liste.

Klikom na dugme za brisanje šalje se AJAX zahtev na stranicu delete\_advertisement.php, koja se nalazi u drugoj datoteci. JS kod za to je prikazan u nastavku:

```
$(".advert-list>.item>.info>.btn-bar>a[id$='delete']").click(function (e)
{
    e.preventDefault();
    let id = e.target.id;
    if (id != '') {
        let adId = `#${id}`).data('ad-id');
        if (confirm('Are you sure you want to delete this
advertisement?')) {
            console.log('gonna delete ' + adId);
            let formData = {
                userId: userId,
                adId: adId,
            };
            $.ajax({
                type: "POST",
                url: "delete_advertisement.php",
                data: {
                    formData: formData
                },
                dataType: 'json',
            }).done(function(response) {
                console.log(response);
                if (response.status == 200) {
                    alert('Advertisement deleted successfully!');
                    window.location.reload();
                } else {
                    alert(`Unknown error occurred, please try again after
some time. ${response.message}`);
                }
            });
        }
    }
});
```

PHP kod unutar delete\_advertisement.php je prikazan ispod:

```
<?php
$formData = $_POST['formData'] ?? null;
if ($formData != null) {
    $response = ["status" => 500, "message" => 'server error'];
    ob_start();
    try {
        require_once 'db_config.php';
        $query = $dbh->prepare("SELECT * FROM user_accounts WHERE user_id
= :u AND account_state = 'ADMIN'");
        $query->bindParam(':u', $formData['userId'], PDO::PARAM_INT);
        $query->execute();
        $isAdmin = ($query->rowCount() > 0);
        if (!$isAdmin) {
            $sql =
                "DELETE FROM advertisements
                WHERE user_id = :user_id AND ad_id = :ad_id;";
            $query = $dbh->prepare($sql);
            $query->bindParam(':ad_id', $formData['adId'],
PDO::PARAM_INT);
            $query->bindParam(':user_id', $formData['userId'],
PDO::PARAM_INT);
        } else {
            $sql =
                "DELETE FROM advertisements
                WHERE ad_id = :ad_id;";
            $query = $dbh->prepare($sql);
            $query->bindParam(':ad_id', $formData['adId'],
PDO::PARAM_INT);
        }
        $query->execute();
        if ($query->rowCount() > 0) {
            $response['status'] = 200;
            $response['message'] = 'deleted';
        } else {
            $response['status'] = 400;
            $response['message'] = 'bad data';
        }
    } catch (Exception $e) {
        echo $e;
    } finally {
        $response['echo'] = ob_get_contents();
        ob_end_clean();
        echo json_encode($response);
        die();
    }
}
```

Skripta proverava da li je korisnik administrator ili ne. Ako nije administrator, oglas se briše samo ako je korisnik vlasnik, u suprotnom daje pogrešan odgovor. Ako je korisnik administrator, korisnik se ne proverava i reklama se direktno briše. Na JS strani se prikazuje upozorenje sa porukom potvrde o brisanju i stranica se ponovo učitava.

Stranica za iznajmljivanje ima nekoliko opcija za filtriranje prikazanih reklama, a HTML-PHP-JS kod za sistem filtriranja je sledeći:

Klasa oglasa:

```
<script>
    class Advert {
        constructor(element, name, type_id, type, location_id,
location, addr, capacity, parking, pets, period, price) {
            this.element = element;
            this.name = name;
            this.type_id = type_id;
            this.type = type;
            this.location_id = location_id;
            this.location = location;
            this.addr = addr;
            this.capacity = capacity;
            this.parking = parking;
            this.pets = pets;
            this.period = period;
            this.price = price;
        }

        matchLocationId(id) {
            return this.location_id == id;
        }

        matchTypeId(id) {
            return this.type_id == id;
        }

        checkPrice(val) {
            return this.price <= val;
        }

        checkPeriod(val) {
            return this.period <= val;
        }

        checkCapacity(val) {
            return this.capacity >= val;
        }

        hasParking() {
            return this.parking;
```

```

        }

        hasPets() {
            return this.pets;
        }

        hasFlexible() {
            return this.period == 0;
        }
    }
    var ads = [];
</script>

```

Ovo je korišćeno kao efikasan način da se prate atributi postojećih reklama. Prvo svojstvo, element čuva jQuery element koji odgovara stavci liste na listi oglasa. Ovo se može koristiti za sakrivanje/prikaz stavke liste.

Generisanje menija filtera i liste:

```

<div class="filters-section">
    <span class="search-box">
        <a href="#" id="expand-filters-btn"><i class="bi bi-filter"></i></a>
        <input type="search" placeholder="Filter by name, location,
address..." name="search-term" id="search-term">
        <a href="#" id="search-btn"><i class="bi bi-search"></i></a>
    </span>
    <div class="filters">
        <span class="inp-group">
            <label for="location-id">Location: </label>
            <select name="location-id" id="location-id">
                <option value="" hidden>Select a Location</option>
                <?php
                    foreach ($locations as $loc) {
                        echo "<option
value='".$loc->location_id'>$loc->location_name</option>";
                    }
                ?>
            </select>
        </span>
        <span class="inp-group">
            <label for="type-id">Type: </label>
            <select name="type-id" id="type-id">
                <option value="" hidden>Select a type</option>
                <?php
                    foreach ($types as $type) {
                        echo "<option
value='".$type->type_id'>$type->type_name</option>";
                    }
                ?>
            </select>
        </span>
    </div>
</div>

```

```

        </span>
        <input placeholder="Maximum Price (RSD)" name="max-price"
id="max-price" type="number" min=0>
        <input placeholder="Maximum Period (Months)" name="max-period"
id="max-period" type="number" min=0>
        <input placeholder="Minimum Capacity" name="min-capacity"
id="min-capacity" type="number" min=1>
        <span style="display: flex; justify-content:space-between;">
            <div class="checkbox-group">
                <input type="checkbox" name="parking" id="parking">
                <label>Parking</label>
            </div>
            <div class="checkbox-group">
                <input type="checkbox" name="pets" id="pets">
                <label>Pets</label>
            </div>
            <div class="checkbox-group">
                <input type="checkbox" name="flexible" id="flexible">
                <label>Flexible Period</label>
            </div>
        </span>
        <span style="display: flex; gap: 10px;">
            <a class="button-primary" style="flex-grow: 1" href="#" id="apply-filters-btn">Apply Filters</a>
            <a class="button-primary" href="#" id="clear-filters-btn"><i
class="bi bi-x-lg"></i></a>
        </span>
    </div>
<div class="advert-list">
    <?php
    if (empty($adverts)) {
        echo '<span id="no-ads-text">No Ads posted yet!</span>';
    } else {
        foreach ($adverts as $ad) {
            $photos =
            json_decode(file_get_contents("uploads/$ad->property_id.json"));
            $photosStr = "";
            $first = true;
            foreach ($photos as $photo) {
                if ($first) {
                    $photosStr .= "<img class='active' src='$photo'>" .
PHP_EOL;
                    $first = false;
                } else {
                    $photosStr .= "<img src='$photo'>" . PHP_EOL;
                }
            }
            $actions = ($ad->is_owner ? ''
                : "<a href='#' id='$ad->id-fav' data-ad-id='$ad->id'
class='button-primary button-sm'><i class='bi bi-star'></i> +Fav</a>

```

```

        <a href='#' id='$ad->id-contact' data-ad-id='$ad->id'
class='button-primary button-sm'><i class='bi
bi-envelope-paper-fill'></i> Contact</a>");
echo
"<div class='item' id='$ad->id'>
    <div class='photo-reel'>
        $photosStr
        <a href='#' class='prev-btn'><i class='bi
bi-arrow-left'></i></a>
        <a href='#' class='next-btn'><i class='bi
bi-arrow-right'></i></a>
    </div>
    <div class='info'>
        <span class='name'$ad->name</span>
        <span class='type'$ad->type</span>
        <span class='location'$ad->location</span>
        <span class='specs'>
            <span class='capactiy' title='Capacity'><i
class='bi bi-people-fill'></i> $ad->capacity</span>
            <span class='parking' title='Parking'><i
class='bi bi-p-square-fill'></i> " . ($ad->parking == 1 ? 'Yes' : 'No') .
"</span>
            <span class='pets' title='Pets'>😺" . ($ad->pets
== 1 ? 'Yes' : 'No') . "</span>
        </span>
        <span class='price-period'>RSD $ad->price p/m, " .
($ad->period == 0 ? 'Flexible Period' : "$ad->period Months") . "</span>
        <span class='description'>$ad->description</span>
        <span class='btn-bar'>
            <a href='#' id='$ad->id-view'
data-ad-id='$ad->id' class='button-primary button-sm'><i class='bi
bi-eye-fill'></i> View</a>
            $actions
        </span>
    </div>
</div>";
echo "<script>
    ads.push(
        new Advert(
            $('#$ad->id'),
            '$ad->name',
            $ad->type_id,
            '$ad->type',
            $ad->location_id,
            '$ad->location',
            '$ad->addr',
            $ad->capacity,
            ($ad->parking == 1 ? true : false),
            ($ad->pets == 1 ? true : false),
            $ad->period,
            $ad->price

```

```

        )
    );
</script>";
}
}
?>
</div>

```

Meni filteri pruža opcije za filtriranje prema terminu, koji je uparen sa imenima svojstava, lokacijama, tipovima i adresama; ili filtriranje kroz određenu lokaciju, tip imovine, minimalni kapacitet, maksimalnu cenu i period, da li je parking dostupan, dozvoljeni su kućni ljubimci i da li je period fleksibilan ili ne. Interno, fleksibilni period je predstavljen vrednošću 0, a po potrebi se konvertuje u nizove.

Primena filtera:

```

var filtersOpen = false;

$('#expand-filters-btn').on('click', function (e) {
    e.preventDefault();
    if (filtersOpen) {
        filtersOpen = false;
        $('.filters').css('height', '0');
        $('.filters').css('opacity', '0');
    } else {
        filtersOpen = true;
        $('.filters').css('height', '430px');
        $('.filters').css('opacity', '1');
    }
});

var filtersApplied = true;

function clearFilters() {
    $('#location-id').val("");
    $('#type-id').val("");
    $('#max-price').val("");
    $('#max-period').val("");
    $('#min-capacity').val("");
    $('#parking').prop("checked", false);
    $('#pets').prop("checked", false);
    $('#flexible').prop("checked", false);
    $('#search-term').val("");
    for (let i = 0; i < ads.length; i++) {
        ads[i].element.css('display', 'flex');
    }
}

$('#clear-filters-btn').on('click', function (e) {
    e.preventDefault();
    filtersOpen = false;
    filtersApplied = false;

```

```

$('.filters').css('height', '0');
$('.filters').css('opacity', '0');
clearFilters();
});

$('#apply-filters-btn').on('click', function (e) {
    e.preventDefault();
    filtersOpen = false;
    filtersApplied = true;
    $('.filters').css('height', '0');
    $('.filters').css('opacity', '0');

    let filters = {
        locationId: $('#location-id').val(),
        typeId: $('#type-id').val(),
        maxPrice: parseInt($('#max-price').val()),
        maxPeriod: parseInt($('#max-period').val()),
        minCapacity: parseInt($('#min-capacity').val()),
        hasParking: $('#parking').is(":checked"),
        hasPets: $('#pets').is(":checked"),
        hasFlexible: $('#flexible').is(":checked"),
    };
    console.log(filters);
    for (let i = 0; i < ads.length; i++) {
        let ad = ads[i];
        let isHidden = (filters.locationId != '' &&
            !ad.matchLocationId(filters.locationId)) ||
            (filters.typeId != '' && !ad.matchTypeId(filters.typeId)) ||
            (!isNaN(filters.maxPrice) &&
            !ad.checkPrice(filters.maxPrice)) ||
            (!isNaN(filters.maxPeriod) &&
            !ad.checkPeriod(filters.maxPeriod)) ||
            (!isNaN(filters.minCapacity) &&
            !ad.checkCapacity(filters.minCapacity)) ||
            (filters.hasParking && !ad.hasParking()) ||
            (filters.hasPets && !ad.hasPets()) ||
            (filters.hasFlexible && !ad.hasFlexible());
        if (isHidden) ads[i].element.css('display', 'none');
        else ads[i].element.css('display', 'flex');
    }
});

function search() {
    let term = $('#search-term').val().toLowerCase();
    if (term == '') {
        clearFilters();
    }
    for (let i = 0; i < ads.length; i++) {
        let ad = ads[i];
        let isHidden = !(ad.name.toLowerCase().includes(term) ||
            ad.location.toLowerCase().includes(term) ||

```

```

        ad.type.toLowerCase().includes(term) ||
        ad.addr.toLowerCase().includes(term));
    if (isHidden) ads[i].element.css('display', 'none');
    else ads[i].element.css('display', 'flex');
}
}

$('#search-term').on('search', function (e) {
    search();
});

$('#search-btn').on('click', function (e) {
    search();
});

```

Pošto je meni filtera padajući meni, njegovo stanje vidljivosti se čuva u promenljivoj filtersOpen. Kada korisnik klikne na dugme primeni filtere, unos u svim nepraznim poljima filtera se prikuplja u JS objekat i onda se utvrđuje da li ta stavka liste oglasa treba da bude sakrivena ili ne. Promena vidljivosti se lako primenjuje pomoću jQuery-a. Kada kliknete na dugme za brisanje filtera, sva polja se brišu i sve stavke liste postaju vidljive tako što se ekran vrati na flex.

Slično tome, kada korisnik klikne na dugme za pretragu ili pritisne taster enter, funkcija pretrage se koristi za proveru sličnosti sa nekoliko nizova u svakoj stavci liste oglasa i shodno tome sakriva nerelevantne stavke.

Ako je registrovani korisnik zainteresovan za iznajmljivanje nekretnine, može kliknuti na dugme Contact na stavci liste. Ovo dugme je skriveno ako je korisnik vlasnik navedene imovine. Kada ga kliknu, biće odvedeni na stranicu na kojoj mogu dodati poruku vlasniku imovine. Ova poruka se zatim uključuje sa ostalim detaljima o objavljenom oglasu i šalje se vlasniku nekretnine (osobi koja je postavila oglas), gde oni mogu da kontaktiraju osobu koja je zainteresovana tako što će odgovoriti na ovu e-poštu. Ovo se radi podešavanjem pomoću funkcije addReplyTo kao što je prikazano u nastavku:

```
$mail->addReplyTo($sender_email, $sender_name);
```

Kada se vlasnik i potencijalni zakupac dogovore, vlasnik može nastaviti sa uređivanjem postavljenog oglasa i označiti ga kao Rented Out. Ovo ga sakriva sa liste za iznajmljivanje. Ovo se radi pomoću sledećeg JS koda:

```

$('#mark-rented-btn').click(function(e) {
    e.preventDefault();
    root.style.setProperty('--modal-blur', '5px');
    root.style.setProperty('--user-popup-opacity', '1');
    root.style.setProperty('--user-popup-events', 'unset');
});

var tenantFound = false;
var searching = false;

```

```

function searchTenant() {
    if (searching) return;
    let email = $('#search-email').val();
    if (!validateEmail(email)) {
        $('#search-email-error').text("Please enter a valid email.");
    } else {
        $('#search-email-error').text("");
        let formData = {
            email,
            task: 'search_tenant',
        };
        console.log(formData);
        $('#tenant-submit').val('Searching...');
        $('#tenant-submit').prop('disabled', true);
        $('#search-email').prop('disabled', true);
        $.ajax({
            type: "POST",
            url: "edit_advertisement.php",
            data: {
                formData: formData
            },
            dataType: 'json',
        }).done(function(response) {
            console.log(response);
            searching = false;
            $('#tenant-submit').prop('disabled', false);
            $('#search-email').prop('disabled', false);
            if (response.status == 200) {
                tenantFound = true;
                $('#search-email-error').css('color', 'green');
                $('#search-email-error').text("Email found.");
                $('#tenant-submit').val('Mark as Rented');
            } else if (response.status == 404) {
                $('#search-email-error').text("Email not found.");
                $('#tenant-submit').val('Search');
            } else {
                alert(`Unknown error occurred, please try again after
some time. ${response.message}`);
            }
        });
    }
}

$('#search-email').on('input', function(e) {
    if (tenantFound) {
        $('#search-email-error').text("");
        $('#search-email-error').css('color', 'red');
        $('#tenant-submit').val('Search');
        tenantFound = false;
    }
});

```

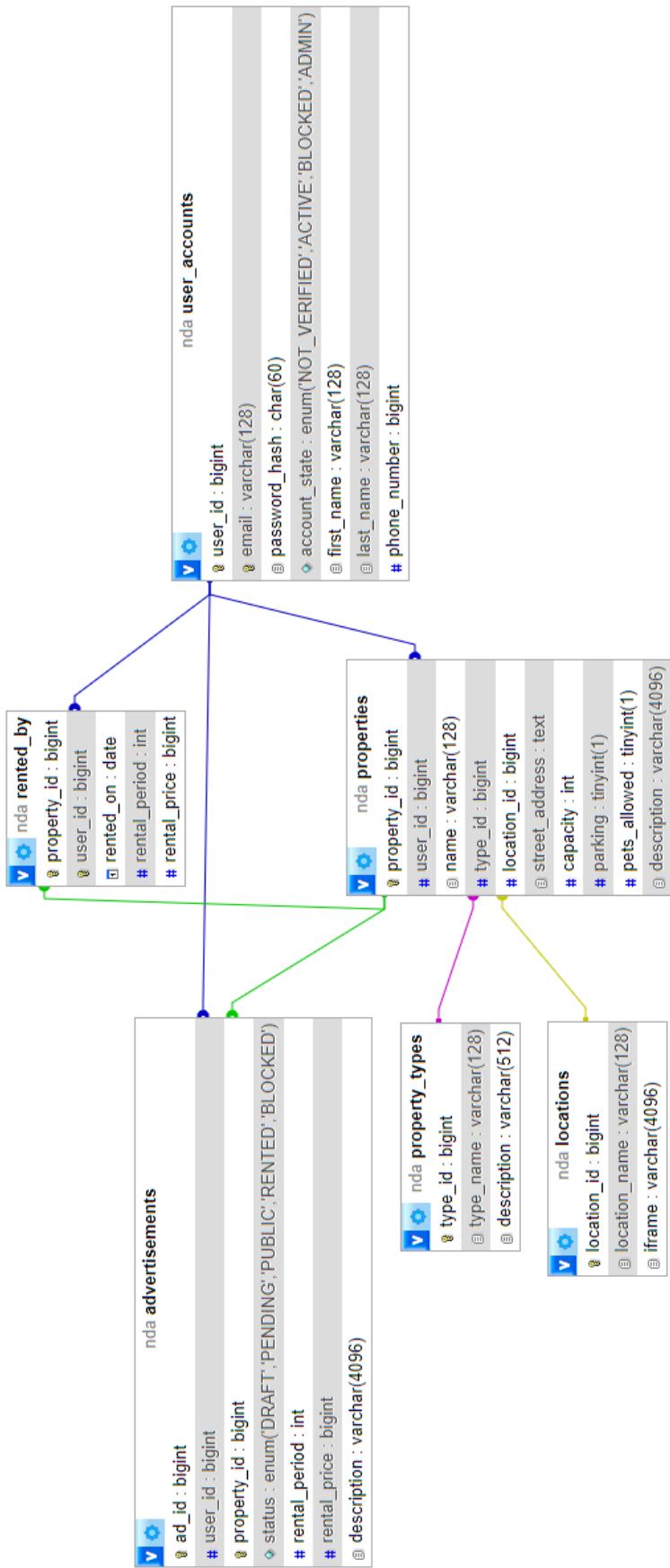
```

$( '#tenant-form' ).submit( function(e) {
    e.preventDefault();
    if (!tenantFound) {
        searchTenant();
        return;
    }
    let formData = {
        adId: $('#ad-id').val(),
        userId: $('#user-id').val(),
        tenantEmail: $('#search-email').val(),
        task: 'mark_rented',
    };
    console.log(formData);
    $.ajax({
        type: "POST",
        url: "edit_advertisement.php",
        data: {
            formData: formData
        },
        dataType: 'json',
    }).done(function(response) {
        console.log(response);
        if (response.status == 200) {
            alert('Advertisement successfully marked as Rented!');
            $returnUrl = $('#cancel-btn').attr('href');
            window.location.replace($returnUrl);
        } else if (response.status == 400) {
            alert('Invalid data entered, please check and try again!');
        } else {
            alert(`Unknown error occurred, please try again after some
time. ${response.message}`);
        }
    });
});

```

Kada korisnik klikne na dugme Označi kao iznajmljeno, iskačući prozor postaje vidljiv. Ovaj iskačući prozor ima polje za unos za unos e-pošte novog zakupca. Prilikom prvog slanja obrasca, e-pošta se ispituje u bazi podataka i ako se pronađe, može se postaviti kao zakupac. Ako nije pronađena, prikazuje se odgovarajuća poruka o grešci. Prilikom ponovnog slanja obrasca, korisnik koji odgovara unetom e-mailu je označen kao trenutni zakupac nekretnine, a oglas je označen kao RENTED.

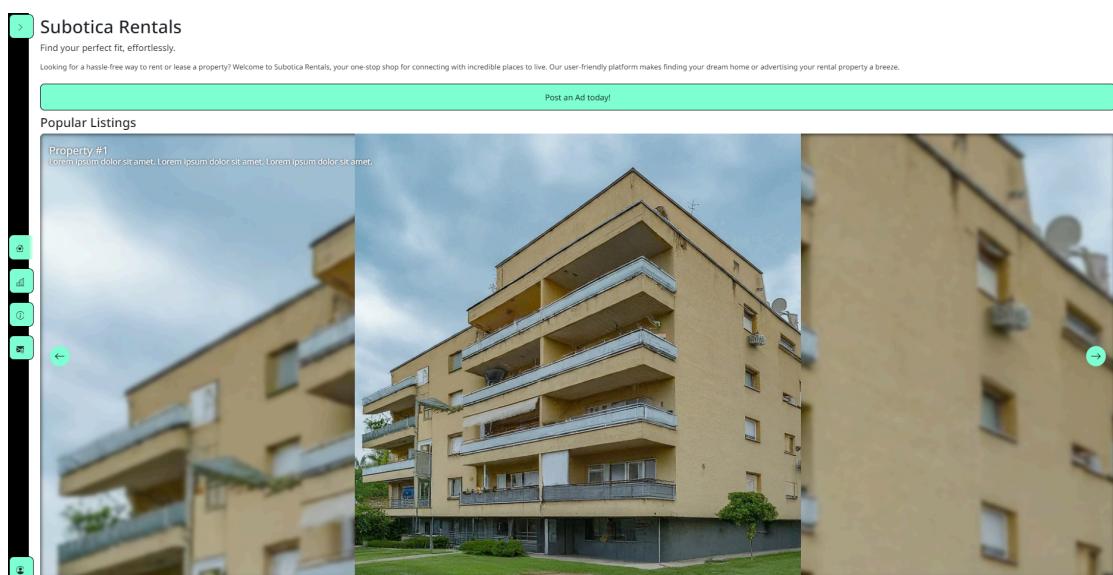
### 3. Struktura baze podataka



1.Slika struktura baze

## 4. Opis funkcionalnosti

Početna stranica, koja uključuje poruku dobrodošlice i galeriju slika, prikazujući fotografije nekih nekretnina navedenih za iznajmljivanje na web stranici. Galerija je implementirana koristeći PHP i JavaScript. PHP se koristi za učitavanje slika nekretnina, dok JavaScript omogućava interakciju korisnika, poput prelaska između prikazanih fotografija.

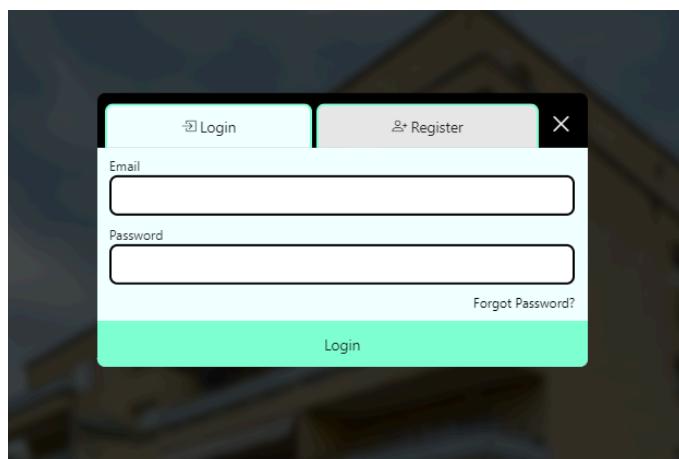


2. Slika Home Page

Kod prijave, podaci iz obrasca se prikupljaju u JavaScript objektu `loginData`. Sistem traži heširanu lozinku za unetu e-poštu. Ako je nađena i verifikovana, proverava se status naloga. Aktivni nalozi se prijavljuju, dok neverifikovani ili blokirani nalozi ne mogu da se prijave.

Kod registracije, podaci iz obrasca se prikupljaju u objektu `registerData`. Sistem proverava jedinstvenost e-pošte. Ako je jedinstvena, podaci se ubacuju sa statusom `NOT_VERIFIED`. Verifikacioni email sa generisanim URL-om se šalje korisniku pomoću PHP funkcije `str_replace`.

Ovaj proces osigurava sigurnu prijavu i registraciju korisnika.



3. Slika Login

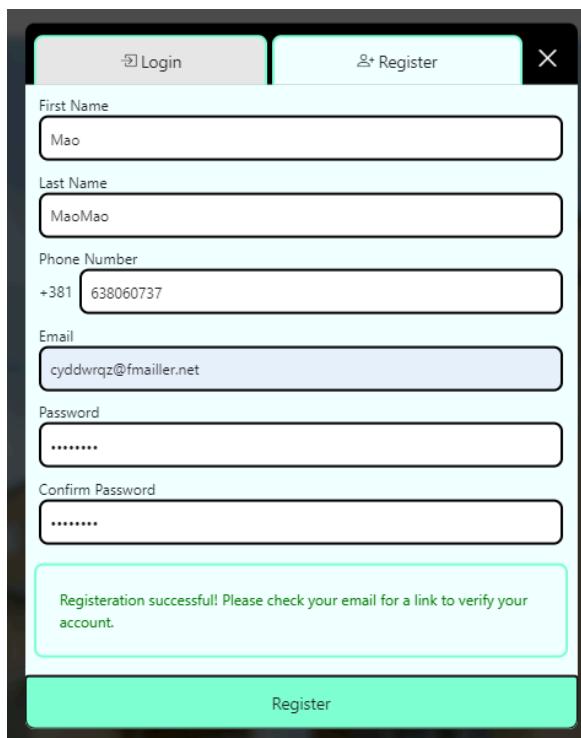


The image shows a registration form window titled 'Register'. It contains fields for First Name, Last Name, Phone Number, Email, Password, and Confirm Password. Each field has a placeholder text above it. A large green 'Register' button is at the bottom.

First Name	<input type="text"/>
Last Name	<input type="text"/>
Phone Number	+381 <input type="text"/>
Email	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
<input type="button" value="Register"/>	

4.Slika Register

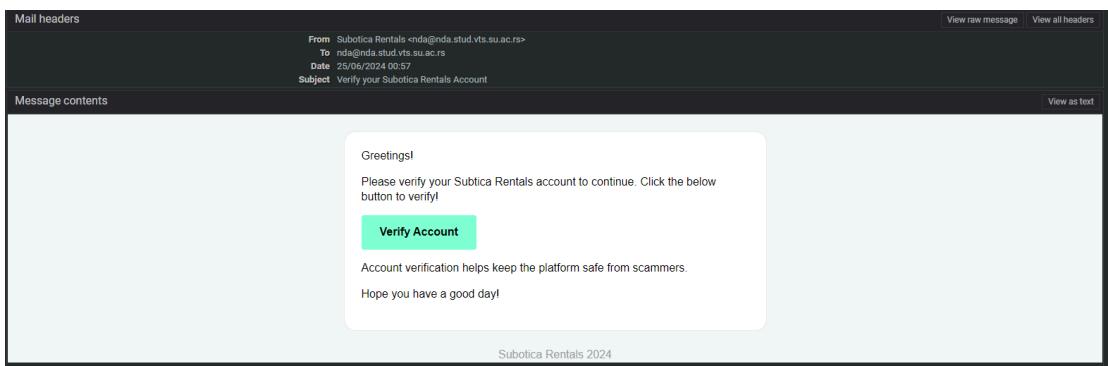
Kada korisnik uneće podatke u registraciona polja, svi unosi se prikupljaju u JavaScript objekt nazvan `registerData`. Nakon prikupljanja podataka, sistem proverava da li su sve informacije validne. Ukoliko postoji nevažeći unos, odgovarajuća poruka se prikazuje ispod svakog problematičnog polja, obaveštavajući korisnika o grešci. Ako su svi unosi ispravni, AJAX se koristi za slanje `registerData` objekta putem POST zahteva na server. Nakon uspešne registracije, korisniku se prikazuje rezultat procesa registracije, potvrđujući da je sve prošlo u redu. Ovaj postupak osigurava da se unose samo tačne informacije i omogućava nesmetanu registraciju korisnika.



The image shows the same registration form as in Slika 4, but with a green message box at the bottom stating 'Registration successful! Please check your email for a link to verify your account.' The rest of the form fields and layout are identical.

First Name	<input type="text" value="Mao"/>
Last Name	<input type="text" value="MaoMao"/>
Phone Number	+381 <input type="text" value="638060737"/>
Email	<input type="text" value="cyddwrqz@fmailler.net"/>
Password	<input type="password" value="*****"/>
Confirm Password	<input type="password" value="*****"/>
Registration successful! Please check your email for a link to verify your account.	
<input type="button" value="Register"/>	

5.Slika Verify Email



## 6.Slika Email Verification

### Users

User ID: 1 Priyanshi Agrahari priyanshi.agr4@gmail.com +381 917991890	<b>ACCOUNT ACTIVE</b>
User ID: 7 Greg Green greggreen9@gmail.com +381 765456209	<b>BLOCKED BY ADMIN</b>
User ID: 8 Rick Sanchez earth.c3@gmail.com +381 737373737	<b>AWAITING EMAIL VERIFICATION</b>
User ID: 9 Mao MaoMao cyberwiz@mailster.net +381 038690737	<b>ACCOUNT ACTIVE</b>

## 7.Slika Admin Users

Lista prikazuje dugme za blokiranje ili deblokiranje samo ako je nalog u stanju ACTIVE ili BLOCKED. Svaka stavka na listi sadrži ID korisnika, ime, adresu e-pošte, telefonski broj i stanje naloga. Stanje naloga je vizualno naglašeno bojom, a stilizacija se vrši pomoću CSS-a.

## 8.Slika My Properties

Podaci o nekretnini se pakiraju u JavaScript objekt i šalju na server putem POST zahtjeva. Fotografije nekretnina šalju se kao base64 stringovi i pohranjuju u JSON listu u datoteci na serveru, gdje je ime datoteke primarni ključ za novu nekretninu. Ovaj pristup osigurava da nema problema s imenovanjem datoteka.

Kada je korisnik prijavljen, iz baze podataka se dohvataju svi dostupni tipovi nekretnina i lokacije, omogućujući korisniku da odabere željene opcije.

Add Property

Name of Property

Type of Property

Location in Subotica  


Street Address

Capacity (Approx. No. of Residents)

Parking Available?  
 Pets Allowed?

Add Photos (min. 3, max. 10)  


Description

### 9.Slika Add Property

Korisnik bira lokacije pomoću "location selector" koji prikazuje Google mape za svaku lokaciju. Tip nekretnine se odabire preko selektora popunjeno preko PHP-a. Fotografije se otpremaju i pregledaju u "photo uploader" divu. Korisnik može dodati do 10 slika po nekretnini, a svaka se čuva sa vremenskom oznakom kao ključem u folderu "uploadedPhotos". Pregledane slike se dodaju dinamički u div "photo uploader", omogućavajući korisniku jednostavno dodavanje i uklanjanje fotografija sa liste.

#### Post Advertisement

Select Property to Advertise  


Rental Period (in Months)

Rental Price (per month)

Description

### 10.Slika Post Property

Kada je oglas označen kao PENDING, prikazuje se na stranici "Pending Advertisements" za administratore. Oglas postaje javno vidljiv nakon što administrator klikne na dugme "Approve". Alternativno, može ga blokirati klikom na dugme "BLOCK". Lista prikazuje ključne informacije o oglasu zajedno sa fotografijama.

## My Advertisements

The screenshot shows a list of four advertisements:

- RENTED OUT**: A yellow house with a green lawn. Details: hjjkl, 3 BHK Apartment, Dudova Sma, RSD 9077 p/m, Flexible Period. Status: PUBLICLY VISIBLE.
- PROPERTYEAFDAJ**: An interior room with a sofa and a window. Details: PROZVKA, 4 BHK Apartment, RSD 5000 p/m, 12 Months. Status: PUBLICLY VISIBLE.
- something**: A train in a station. Details: Zorka, 2 BHK Apartment, RSD 20000 p/m, 4 Months. Status: PENDING APPROVAL.
- something**: A train in a station. Details: Zorka, 2 BHK Apartment, RSD 20000 p/m, 4 Months. Status: PENDING APPROVAL.

## 11.Slika Status of AD

### Approval Pending Advertisements

The screenshot shows one advertisement in pending approval status:

- something**: A train in a station. Details: Zorka, 2 BHK Apartment, RSD 20000 p/m, 4 Months. Status: PENDING APPROVAL.

## 12.Slika Approval Pending Advertisment by Admin

Meni filteri omogućavaju korisnicima da biraju između različitih opcija za filtriranje, uključujući termin, lokaciju, tip nekretnine, kapacitet, cenu i druge karakteristike kao što su dostupnost parkinga i kućnih ljubimaca.

### Rental Advertisements

The screenshot shows a filtering interface for rental advertisements:

- Filter by name, location, address... (text input)
- Location: Select a Location (dropdown)
- Type: Select a type (dropdown)
- Maximum Price (RSD) (text input)
- Maximum Period (Months) (text input)
- Minimum Capacity (text input)
- Parking (checkbox)
- Pets (checkbox)
- Flexible Period (checkbox)
- Apply Filters (button)
- Result preview: An interior room with a sofa and a window. Details: RSD 5000 p/m, 12 Months. Status: PUBLICLY VISIBLE.

## 13.Slika Filteri

## 5. Korišćena literatura

1. Danny Goodman: „JavaScript Biblija”, Mikro knjiga, Beograd, 2000.
2. David Flanagan: „JavaScript:sveobuhvatni vodič” Mikro knjiga, Beograd, 2011.

W1. <http://www.w3schools.com/>

W2. <http://www.jquery.com>

W3. <https://getbootstrap.com/>

W4. <https://fonts.googleapis.com/> (Noto Sans)

W5. <https://icons.getbootstrap.com/>