**VISOKA TEHNIČKA ŠKOLA**

**STRUKOVNIH STUDIJA**

**S** **U B O T I C A**

# PROJEKAT

**iz predmeta Web programiranje**

***Online Shopping lista***

**KANDIDAT MENTOR**

**Aleksa Canak dr Zlatko Čović**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SUBOTICA 2023.god.**

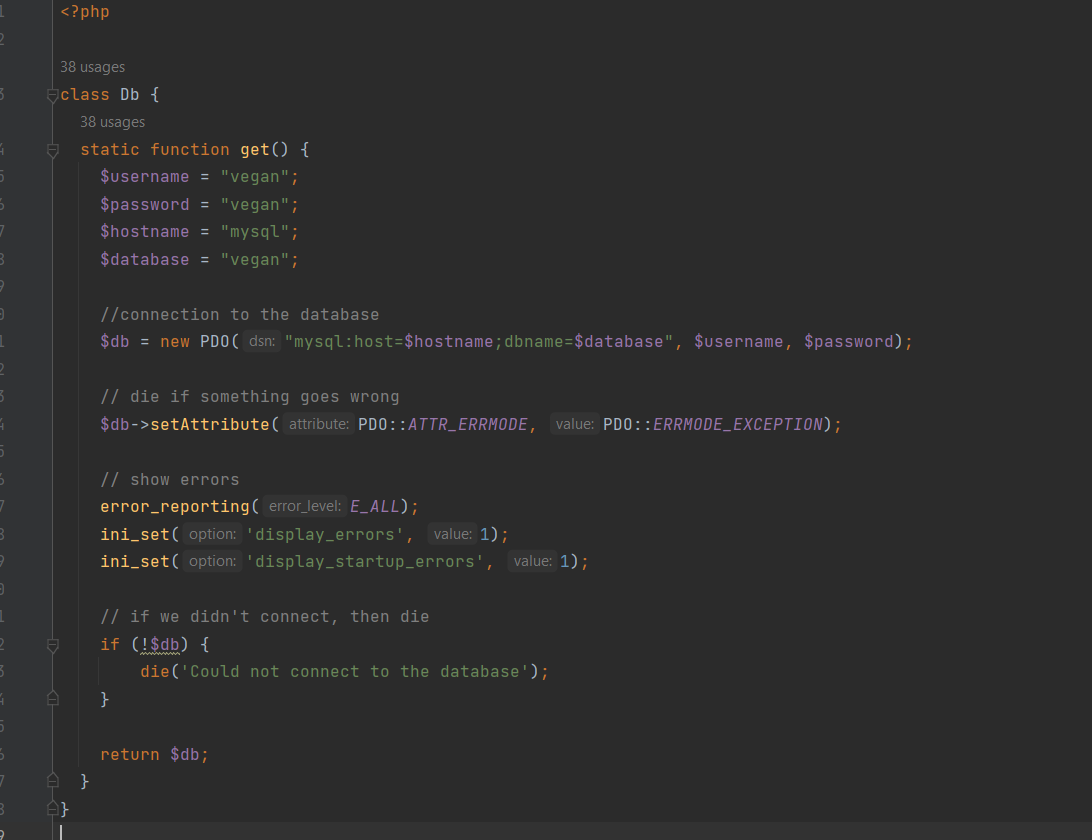
## KATALOG

**[PROJEKAT 1](#_Toc26549)**

[IZRADA 3](#_Toc11539)

[LITERATURA 19](#_Toc29038)

## IZRADA



U ovom kodu sam uspostavio vezu sa Bazom Podataka.

<?php  
$logged\_in = Session::*isLoggedIn*();  
?>  
<!DOCTYPE html>  
<html>  
<head>  
 <title>Vegan Shop</title>  
 <link rel="stylesheet" type="text/css" href="https://bootswatch.com/5/united/bootstrap.min.css">  
 <link rel="stylesheet" type="text/css" href="/css/main.css">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
</head>  
<body>  
 <nav class="navbar navbar-expand-lg navbar-dark bg-primary mb-3">  
 <div class="collapse navbar-collapse" id="mainNavbar">  
 <ul class="navbar-nav">  
 <li class="nav-item">  
 <a class="nav-link" href="/index.php">Home</a>  
 </li>  
 <?php if ($logged\_in) { ?>  
 <li class="nav-item">  
 <a class="nav-link" href="/user/profile.php">Profile</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link" id='cart-link' href="/user/cart.php">Cart</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link" href="/user/shopping-lists.php">Shopping Lists</a>  
 </li>  
 <?php if (Session::*getSessionUser*()['admin']) { ?>  
 <li class="nav-item">  
 <a class="nav-link" href="/admin/index.php">Admin</a>  
 </li>  
 <?php } ?>  
 <li class="nav-item">  
 <a class="nav-link" href="/user/logout.php">Logout</a>  
 </li>  
 <?php } else { ?>  
 <li class="nav-item">  
 <a class="nav-link" href="/user/login.php">Login</a>  
 </li>  
 <li class="nav-item">  
 <a class="nav-link" href="/user/register.php">Register</a>  
 </li>  
 <?php } ?>  
 </ul>  
 </div>  
 </nav>

U ovom PHP kodu sam izkodirao html deo sajta. Izgled se menja u zavisnosti da li smo ulogovani kao admin ili ne, I da li smo uopste ulogovani.

<?php  
// require all libs  
require\_once('lib/lib.php');  
// include header  
include\_once('views/header.php');  
  
$products = Product::*getAll*();  
?>  
<div class="container">  
 <h1>Online shopping lista</h1>  
 <div class="products">  
 <?php  
 foreach ($products as $product) {  
?>  
 <div class="card" style="width: 300px">  
 <img class="card-img-top" src="/uploads/<?= $product['image']; ?>" alt="Card image cap">  
 <div class="card-body">  
 <h5 class="card-title"><?= $product['name']; ?></h5>  
 <p class="card-text"><?= $product['description']; ?></p>  
 <p class="card-text">Price: <?= $product['price']; ?></p>  
 <a href="/product.php?id=<?= $product['id'] ?>" class="btn btn-primary">View</a>  
 </div>  
 </div>  
 <?php  
 }  
 ?>  
 </div>  
</div>  
  
<?php  
// include footer  
include\_once('views/footer.php');  
?>

Ovaj kod kombinuje PHP jezik za obradu podataka i HTML za prikazivanje elemenata na web stranici. Ovaj deo koda obuhvata HOME stranicu na sajtu. Cilj mu je prikazivanje liste proizvoda sa osnovnim informacijama i omogućavanje korisnicima da pregledaju detalje proizvoda klikom na odgovarajuće dugme.

<?php  
// require all libs  
require\_once('lib/lib.php');  
// include header  
include\_once('views/header.php');  
  
if (Request::*hasAllKeys*($\_GET, ['id'])) {  
 $id = $\_GET['id'];  
 $product = Product::*findById*($id);  
  
 if (Request::*isSubmitted*()) {  
 $user = Session::*getSessionUser*();  
 Product::*addToCart*($id, $user['id']);  
  
 Response::*showSuccess*('Product added to cart');  
 }  
}  
else {  
 die(Response::*showError*('Missing required parameters'));  
}  
  
?>  
<div class="container">  
 <h1>Online Shopping lista</h1>  
 <h2>Product: <?= $product['name']; ?></h2>  
 <p>Price: <?= $product['price']; ?></p>  
 <p>Description: <?= $product['description']; ?></p>  
 <img src="/uploads/<?= $product['image']; ?>" height="500" />  
 <?php  
 if (Session::*isLoggedIn*()) {  
 $lists = ShoppingList::*getListsByUser*(Session::*getSessionUser*()['id']);  
 ?>  
 <form action="/product.php?id=<?= $id ?>" method="post">  
 <input type="submit" name="submit" value="Add to Cart" class="btn btn-primary mt-3" />  
 </form>  
 <?php if (!empty($lists)) { ?>  
 <hr />  
 <form action="/user/add-to-list.php" method="get">  
 <input type="hidden" name="product\_id" value="<?= $id ?>" />  
 <div class="form-group">  
 <label for="id">Name</label>  
 <select name="id" class="form-control">  
 <?php foreach ($lists as $list) { ?>  
 <option value="<?= $list['id'] ?>"><?= $list['name'] ?></option>  
 <?php } ?>  
 </select>  
 </div>  
 <input type="submit" name="submit" value="Add to List" class="btn btn-primary mt-3" />  
 </form>  
 <?php } ?>  
 <?php  
 } else {  
 ?>  
 <p class="mt-3">Please <a href="/login.php">login</a> to add to cart</p>  
 <?php  
 }  
 ?>  
</div>  
  
<?php  
// include footer  
include\_once('views/footer.php');  
?>

U ovom kodu u zavisnosti da li je user prijavljen ili ne imamo nekoliko razlicitih opcija kod proizvoda. Ako je user prijavljen onda mozemo da dodajemo proizvode u korpu ili u nasu listu. Ako nije korisnik prijavljen pokazuje se link za prijavu.

<?php  
// Get current time and format it as DD/MM/YYYY HH:MM:SS  
$time = date('d/m/Y H:i:s');  
  
// Print the time  
echo json\_encode(['time' => $time]);

Ovaj kod je PHP skript koji dobavlja trenutno vreme i formatira ga kao "DD/MM/YYYY HH:MM:SS", a zatim ga prikazuje kao JSON odgovor.

async function getTime() {  
 const data = await fetch("/time.php");  
 const { time } = await data.json();  
 ***document***.getElementById("time").innerHTML = time;  
}  
  
setInterval(getTime, 1000);

Ovaj kod je asinhrona JavaScript funkcija koju sam koristio za dobavljanje vremena sa servera i prikazivanje tog vremena na veb stranici.

async function getCartItemCount() {  
 const data = await fetch(`/user/get\_cart.php?uid=${userId}`);  
 const json = await data.json();  
  
 return json.count;  
}  
  
async function clearCart() {  
 const data = await fetch(`/user/clear\_cart.php?uid=${userId}`);  
  
 ***window***.location.reload();  
}  
  
getCartItemCount().then((count) => {  
 ***document***.getElementById("cart-link").textContent = `Cart (${count})`;  
});

Ovaj kod je asinhrona JavaScript funkcionalnost koja se koristi za prikazivanje broja stavki u korpi i brisanje korpe na veb stranici.

<?php  
// require all libs  
require\_once('../lib/lib.php');  
  
Session::*guestGuard*();  
  
// form handler  
if (Request::*isSubmitted*()) {  
 if (Request::*hasAllKeys*($\_POST, ['email', 'password'])) {  
 $info = Request::*getFromPost*(['email', 'password']);  
 $user = User::*login*($info['email'], $info['password']);  
 if ($user) {  
 Session::*setSessionUser*($user);  
 Response::*showSuccess*("User logged in");  
 } else {  
 Response::*showError*("Invalid email or password");  
 }  
 }  
 else {  
 Response::*showError*("Please fill in all fields");  
 }  
}  
  
// include header  
include\_once('../views/header.php');  
?>  
<?php   
if (!Session::*isLoggedIn*()) {  
?>  
<div class="container">  
 <form action="/user/login.php" method="post">  
 <div class="form-group">  
 <label for="email">Email</label>  
 <input class="form-control" type="email" name="email" placeholder="Email" required>  
 </div>  
  
 <div class="form-group">  
 <label for="password">Password</label>  
 <input class="form-control" type="password" name="password" placeholder="Password" required>  
 </div>  
   
 <input class="btn btn-primary mt-3" type="submit" name="submit" value="Login">  
 </form>  
  
 <a href="/user/forgot-password.php">Forgot password?</a>  
</div>  
<?php  
}  
?>  
<?php  
// include footer  
include\_once('../views/footer.php');  
?>

Prvo se uključuju potrebne biblioteke pomoću require\_once('../lib/lib.php')

Nakon toga se koristi Session::guestGuard() zaštita kako bi se osiguralo da samo gosti koji nisu prijavljeni mogu pristupiti ovoj stranici. Ako je korisnik već prijavljen, biće preusmeren na drugu stranicu.

Zatim se proverava da li je zahtev poslat (HTTP POST metoda). Ako jeste, proverava se da li su sve ključne vrednosti ('email' i 'password') prisutne u POST parametrima. Ako jesu, dobavljaju se vrednosti 'email' i 'password' iz POST parametara pomoću funkcije Request::getFromPost(). Zatim se koristi funkcija User::login($email, $password) da bi se pokušalo prijavljivanje korisnika sa datim emailom i šifrom.

Ako je prijavljivanje uspešno, korisnički podaci se smeštaju u sesiju pomoću funkcije Session::setSessionUser($user), a zatim se prikazuje poruka o uspešnoj prijavi. Ako prijavljivanje nije uspelo, prikazuje se poruka o nevalidnom emailu ili šifri.

Ako POST zahtev nije poslat, prikazuje se forma za prijavljivanje korisnika. Ovo se vrši samo ako korisnik nije već prijavljen, što se proverava pomoću Session::isLoggedIn(). Forma sadrži polja za unos emaila i šifre, kao i dugme za prijavljivanje.

<?php  
// require all libs  
require\_once('../lib/lib.php');  
  
Session::*userGuard*();  
Session::*clearSessionUser*();  
  
// include header  
include\_once('../views/header.php');  
  
Response::*showSuccess*("User logged out");  
  
// include footer  
include\_once('../views/footer.php');  
?>

Prvo se uključuju potrebne biblioteke pomoću require\_once('../lib/lib.php').

Nakon toga se koristi Session::userGuard() zaštita kako bi se osiguralo da samo prijavljeni korisnici mogu pristupiti ovoj stranici. Ako korisnik nije prijavljen, biće preusmeren na drugu stranicu.

Zatim se koristi Session::clearSessionUser() da bi se izbrisali podaci o prijavljenom korisniku.

Zatim se uključuje fajl za zaglavlje stranice pomoću include\_once('../views/header.php').

Nakon toga se koristi Response::showSuccess("User logged out") za prikazivanje poruke o uspešnoj odjavi korisnika.

<?php  
// require all libs  
require\_once('../lib/lib.php');  
  
Session::*guestGuard*();  
  
// include header  
include\_once('../views/header.php');  
  
// form handler  
if (Request::*isSubmitted*()) {  
 if (Request::*hasAllKeys*($\_POST, ['email', 'password', 'confirm\_password'])) {  
 $info = Request::*getFromPost*(['email', 'password', 'confirm\_password']);  
 if (User::*passwordsMatch*($info['password'], $info['confirm\_password'])) {  
 $user = User::*createUser*($info['email'], $info['password']);  
 if ($user) {  
 Response::*showSuccess*("User created. Check your email to activate your account.");  
 } else {  
 Response::*showError*("Email already taken");  
 }  
 } else {  
 Response::*showError*("Passwords don't match");  
 }  
 }  
 else {  
 Response::*showError*("Please fill in all fields");  
 }  
}

Prvo se uključuju potrebne biblioteke pomoću require\_once('../lib/lib.php').

Nakon toga se koristi Session::guestGuard() zaštita kako bi se osiguralo da samo gosti (nisu prijavljeni) mogu pristupiti ovoj stranici. Ako je korisnik već prijavljen, biće preusmeren na drugu stranicu.

Nakon toga sledi obrada forme. Proverava se da li je zahtev poslat (HTTP POST metoda). Ako jeste, proverava se da li su sve ključne vrednosti (npr. 'email', 'password', 'confirm\_password') prisutne u POST parametrima.

Ako jesu, dobavljaju se vrednosti 'email', 'password' i 'confirm\_password' iz POST parametara pomoću funkcije Request::getFromPost().

Zatim se koristi funkcija User::passwordsMatch($password, $confirmPassword) da bi se proverilo da li se šifre podudaraju.

Ako se podudaraju, koristi se funkcija User::createUser($email, $password) da bi se kreirao korisnik sa datim emailom i šifrom.

Ako je kreiranje uspešno, prikazuje se poruka o uspešnom kreiranju korisnika i uputstvo da proveri email za aktivaciju naloga.

Ako email već postoji, prikazuje se poruka o zauzetom emailu. Ako se šifre ne podudaraju, prikazuje se poruka o neslaganju šifri.

Ako POST zahtev nije poslat, nema obrade forme i korisniku se prikazuje forma za registraciju.

<?php  
// require all libs  
require\_once('../lib/lib.php');  
// include header  
include\_once('../views/header.php');  
  
if (Request::*hasAllKeys*($\_GET, ['id', 'product\_id'])) {  
 $id = $\_GET['id'];  
 $product\_id = $\_GET['product\_id'];  
  
 ShoppingList::*removeItemFromList*($id, $product\_id);  
  
 Response::*showSuccess*('Item removed from list successfully');  
}  
else {  
 die(Response::*showError*('Missing required parameters'));  
}  
?>  
<div class="container">  
</div>  
  
<?php  
// include footer  
include\_once('../views/footer.php');  
?>

Prvo se uključuju potrebne biblioteke pomoću require\_once('../lib/lib.php').

Zatim se uključuje fajl za zaglavlje stranice pomoću include\_once('../views/header.php').

Nakon toga se proverava da li su sve ključne vrednosti (npr. 'id', 'product\_id') prisutne u GET parametrima. Ako jesu, uzimaju se vrednosti 'id' i 'product\_id' iz GET parametara.

Koristi se funkcija ShoppingList::removeItemFromList($id, $product\_id) da bi se uklonila stavka iz liste proizvoda. Parametri 'id' i 'product\_id' se prosleđuju funkciji.

Nakon uklanjanja stavke iz liste, prikazuje se poruka o uspešnom uklanjanju.

Ako ključne vrednosti nisu prisutne u GET parametrima, program se prekida i prikazuje se greška "Missing required parameters".

Nakon toga sledi HTML kod gde može biti sadržaj kontejnera ili dodatne informacije koje se prikazuju na stranici. U ovom primeru, kontejner je prazan.

Na kraju, uključuje se fajl za podnožje stranice pomoću include\_once('../views/footer.php')

<?php  
// require all libs  
require\_once('../lib/lib.php');  
  
Session::*guestGuard*();  
  
// include header  
include\_once('../views/header.php');  
  
// get the user id from the url  
if (Request::*hasAllKeys*($\_GET, ['token'])) {  
 $info = Request::*getFromGet*(['token']);  
 $token = $info['token'];  
  
 // verify the user  
 if (User::*verifyUserWithToken*($token)) {  
 Response::*showSuccess*('User verified');  
 } else {  
 Response::*showError*('Invalid token');  
 }  
} else {  
 Response::*showError*('Missing user id');  
}  
  
// include footer  
include\_once('../views/footer.php');  
?>

Ovaj kod je PHP skript koji se koristi za verifikaciju korisnika putem tokena koji je dostupan u URL-u

Prvo se uključuju potrebne biblioteke pomoću require\_once('../lib/lib.php').

Nakon toga se koristi Session::guestGuard() zaštita kako bi se osiguralo da samo gosti (nisu prijavljeni) mogu pristupiti ovoj stranici. Ako je korisnik već prijavljen, biće preusmeren na drugu stranicu.

Zatim se uključuje fajl za zaglavlje stranice pomoću include\_once('../views/header.php').

Nakon toga se proverava da li su sve ključne vrednosti (npr. 'token') prisutne u GET parametrima. Ako jesu, dobavlja se vrednost 'token' iz GET parametara pomoću funkcije Request::getFromGet(). Zatim se koristi funkcija User::verifyUserWithToken($token) da bi se verifikovao korisnik sa datim tokenom. Ako je verifikacija uspešna, prikazuje se poruka o uspešnoj verifikaciji. Ako token nije validan, prikazuje se poruka o nevalidnom tokenu.

Ako ključne vrednosti nisu prisutne u GET parametrima, prikazuje se greška "Missing user id".

Nakon toga sledi uključivanje fajla za podnožje stranice pomoću include\_once('../views/footer.php').

<?php  
// require all libs  
require\_once('../lib/lib.php');  
  
Session::*adminGuard*();  
  
// include header  
include\_once('../views/header.php');  
  
// form handler  
if (Request::*isSubmitted*()) {  
 if (Request::*hasAllKeys*($\_POST, ['name', 'price', 'description'])) {  
 $name = $\_POST['name'];  
 $price = $\_POST['price'];  
 $description = $\_POST['description'];  
 $image = Request::*uploadImage*();  
  
 if ($image) {  
 Product::*create*($name, $description, $price, $image);  
 Response::*showSuccess*('Product created');  
 }  
 else {  
 Response::*showError*('Image upload failed');  
 }  
 }  
 else {  
 Response::*showError*('Missing required parameters');  
 }  
}  
?>  
  
<div class="container">  
 <h1>Add Product</h1>  
 <form action="/admin/add\_product.php" method="post" enctype="multipart/form-data">  
 <div class="form-group">  
 <label for="name">Name</label>  
 <input class="form-control" type="text" name="name" placeholder="Name" required>  
 </div>  
  
 <div class="form-group">  
 <label for="price">Price</label>  
 <input class="form-control" type="number" name="price" min="1" placeholder="Price" required>  
 </div>  
  
 <div class="form-group">  
 <label for="description">Description</label>  
 <textarea class="form-control" name="description" placeholder="Description" required></textarea>  
 </div>  
  
 <div class="form-group">  
 <label for="image">Image</label>  
 <input class="form-control" type="file" name="image" required>  
 </div>  
  
 <input class="btn btn-primary mt-3" type="submit" name="submit" value="Add Product">  
 </form>  
</div>  
  
<?php  
// include footer  
include\_once('../views/footer.php');  
?>

Ovaj kod je PHP skript koji se koristi za dodavanje novog proizvoda od strane administratora na veb stranici.

Prvo se uključuju potrebne biblioteke pomoću require\_once('../lib/lib.php').

Nakon toga se koristi Session::adminGuard() zaštita kako bi se osiguralo da samo administratori mogu pristupiti ovoj stranici. Ako korisnik nije administrator, biće preusmeren na drugu stranicu.

Zatim se uključuje fajl za zaglavlje stranice pomoću include\_once('../views/header.php').

Nakon toga sledi obrada forme. Proverava se da li je zahtev poslat (HTTP POST metoda). Ako jeste, proverava se da li su sve ključne vrednosti (npr. 'name', 'price', 'description') prisutne u POST parametrima. Ako jesu, vrednosti 'name', 'price' i 'description' se uzimaju iz POST parametara. Zatim se koristi funkcija Request::uploadImage() za otpremanje slike. Ako otpremanje slike uspe, koristi se funkcija Product::create($name, $description, $price, $image) da bi se kreirao novi proizvod sa datim podacima. Ako kreiranje proizvoda uspe, prikazuje se poruka o uspešnom kreiranju. Ako otpremanje slike ne uspe, prikazuje se poruka o neuspešnom otpremanju slike.

Ako POST zahtev nije poslat, nema obrade forme i prikazuje se forma za dodavanje proizvoda. Forma sadrži polja za unos naziva, cene, opisa i slike proizvoda.

Na kraju, uključuje se fajl za podnožje stranice pomoću include\_once('../views/footer.php')

<?php  
// require all libs  
require\_once('../lib/lib.php');  
// include header  
include\_once('../views/header.php');  
  
Session::*adminGuard*();  
$user = Session::*getSessionUser*();  
  
if (Request::*hasAllKeys*($\_GET, ['id'])) {  
 $id = $\_GET['id'];  
 $user = User::*findById*($id);  
 if ($user) {  
 if ($user['id'] === Session::*getSessionUser*()['id']) {  
 Response::*showError*('Cannot ban yourself');  
 }  
 else if (!$user['verified']) {  
 Response::*showError*('User is already banned');  
 }  
 else {  
 Admin::*banUser*($id);  
 Response::*showSuccess*('User is now banned');  
 }  
 }  
 else {  
 Response::*showError*('User not found');  
 }  
}  
else {  
 Response::*showError*('Missing required parameters');  
}  
  
// include footer  
include\_once('../views/footer.php');  
?>

Ovaj kod je PHP skript koji se koristi za banovanje korisnika od strane administratora na veb stranici.

Prvo se uključuju potrebne biblioteke pomoću require\_once('../lib/lib.php').

Zatim se uključuje fajl za zaglavlje stranice pomoću include\_once('../views/header.php').

Koristi se Session::adminGuard() kako bi se osiguralo da samo administratori mogu pristupiti ovoj stranici. Ako korisnik nije administrator, biće preusmeren na drugu stranicu.

Dobavlja se trenutni korisnik iz sesije koristeći Session::getSessionUser() i smešta se u promenljivu $user.

Nakon toga se proverava da li su sve ključne vrednosti (npr. 'id') prisutne u GET parametrima. Ako jesu, uzima se vrednost 'id' iz GET parametara i smešta se u promenljivu $id.

Koristi se funkcija User::findById($id) da bi se pronašao korisnik sa datim ID-om. Ako korisnik postoji, vrše se dodatne provere.

Proverava se da li je ID korisnika iz GET parametara isti kao ID trenutno prijavljenog korisnika. Ako jeste, prikazuje se poruka o nemogućnosti zabrane samog sebe.

Proverava se da li je korisnik već zabranjen (polje 'verified' postavljeno na false). Ako jeste, prikazuje se poruka da je korisnik već zabranjen.

Ako korisnik nije zabranjen i prošao je sve provere, koristi se funkcija Admin::banUser($id) da bi se zabranio korisnik sa datim ID-om.

Na kraju, prikazuje se poruka o uspešnom zabranjivanju korisnika.

Ako korisnik sa datim ID-om nije pronađen, prikazuje se poruka da korisnik nije pronađen.

Ako ključne vrednosti nisu prisutne u GET parametrima, prikazuje se greška "Missing required parameters".

<?php  
// require all libs  
require\_once('../lib/lib.php');  
// include header  
include\_once('../views/header.php');  
  
Session::*adminGuard*();  
$user = Session::*getSessionUser*();  
  
if (Request::*hasAllKeys*($\_GET, ['id'])) {  
 $id = $\_GET['id'];  
 $user = User::*findById*($id);  
 if ($user) {  
 if ($user['id'] === Session::*getSessionUser*()['id']) {  
 Response::*showError*('Cannot delete yourself');  
 }  
 else {  
 User::*deleteUser*($id);  
 Response::*showSuccess*('User is now deleted');  
 }  
 }  
 else {  
 Response::*showError*('User not found');  
 }  
}  
else {  
 Response::*showError*('Missing required parameters');  
}  
  
// include footer  
include\_once('../views/footer.php');  
?>

Ovaj kod je PHP skript koji se koristi za brisanje korisnika od strane administratora na veb stranici.

U ovom kodu sam sve isto uradio kao I u prethodnom samo za delete a ne za ban.

## LITERATURA

W1. <https://mailtrap.io/home>

W2. <https://www.w3schools.com/bootstrap/default.asp>

W3. <https://www.w3schools.com/js/default.asp>

W4. <https://www.w3schools.com/sql/default.asp>

W5. <https://www.w3schools.com/php/default.asp>

W6. <https://www.w3schools.com/html/default.asp>

W7. <http://www.jquery.com>

W8. <https://www.w3schools.com/js/js_ajax_intro.asp>

W9. <https://people.vts.su.ac.rs/~chole/wp/primeri-peldak/>