**VISOKA TEHNIČKA ŠKOLA**

**STRUKOVNIH STUDIJA**

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

**PROJEKAT**

**PERSONALNI TRENER**

**iz predmeta Web programiranje**

|  |  |
| --- | --- |
| **KANDIDAT(I)**  **Kristijan Dulić 26122012** | **MENTOR**  **prof. dr Zlatko Čović** |
| **SUBOTICA, 2024.** | |

Sadržaj

[1. Opis zadatka 3](#_Toc156903284)

[2. Realizacija zadatka 4](#_Toc156903285)

[3. Struktura baze podataka (MySQL Workbench) 5](#_Toc156903286)

[4. Opis funkcionalnosti 6](#_Toc156903287)

[5. Korišćena literatura 7](#_Toc156903288)

# Opis zadatka

Zadatak projekta je bio da se kreira web aplikacija za dobijanje predloženih treninga na osnovu odabranih parametara. Aplikacija treba razlikuje četiri nivoa pristupa:

* **Gost** koji može da pogleda ponuđene informacije o usluzi na sajtu kao i ponuđene informacije o aktivnim trenerima, a može i da izvrši registraciju.
* **Registrovan korisnik** koji može da pogleda ponuđene informacije o usluzi na sajtu kao i ponuđene informacije o aktivnim trenerima, takođe može da pregleda predložene treninge po kategorijama, da pretražuje treninge preko opcije Pretrage, da kreira sebi program za svoj lični trening i da menja svoje profilne podatke
* **Registrovan korisnik - trener** koji uz mogućnost promene svojih profilnih podataka da kreira, izmeni i obriše trening programe koji su objavljeni javno i dostupni su svima
* **Administrator**  koji može da unosi, menja i briše kategorije i dostupne vežbe, može i da odobri ili odbije korisnika koji je registrovan kao trener i može da zabrani korisniku pravo na prijavljivanje na sistem.

Web adresa projekta je <https://nm.stud.vts.su.ac.rs/>

Ključne funkcionalnosti:

* **Bezbedna registracija -** Obavezna provera svih unetih podataka kao i obavezna aktivacija putem e-maila, jedinstvena e-mail adresa kao korisničko ime.
* **Kreiranje ličnih trening programa -** korisnici koji nisu registrovani kao treneri mogu da kreiraju trening programe koji su samo za njihvou ličnu upotrebu i ne prikazuju se javno
* **Kreiranje javnih trening programa -** treneri mogu da kreiraju, menjaju i brišu trening programe koji su prikazani javno za sve obične korisnike sajta
* **Tehnički zahtevi** - Upotreba HTML, CSS, JavaScript, AJAX/Fetch API, JSON, Bootstrap, PHP, MySQL, PHPMailer, korišćene PHP sesije za zaštitu administratorskog dela, korišćen bcrypt algoritam za heširanje lozinki, korišćena PDO ekstenzija za rad sa bazom, objektno orijentisani PHP kod, responzivan dizajn, validacija na klijentskoj i serverskoj strani, Trello za upravljanje projektom, Git za verzioniranje koda.
* **Dodatne tehnike -** korišćen MVC model podele rada web aplikacije

# Realizacija zadatka

U ovom delu treba prikazati i delove programskog koda. Programski kod napisati uz pomoć fonta **Courier, veličina slova 10**.

**Kod za pokretanje same aplikacije koja poziva odgovarajući kontroler**

<?php  
session\_start();  
require "../app/core/init.php";  
$app = new App();  
$app->loadController();

**Struktura App klase koja obavlja rutiranje na osnovu sadržaja url-a**

<?php  
class App  
{  
 private $controller = 'Home';  
 private $method = 'index';  
 private function splitUrl(){  
 $url = $\_GET['url'] ?? "home";  
 return explode("/",trim($url,"/"));  
 }  
 public function loadController()  
 {  
 $url = $this->splitUrl();  
 $filename = "../app/controllers/" . ucfirst($url[0]) . "Controller.php";  
 if(file\_exists($filename)){  
 require $filename;  
 $this->controller = ucfirst($url[0]) . "Controller";  
 unset($url[0]);  
 }  
 else{  
 $filename = "../app/controllers/\_404Controller.php";  
 require $filename;  
 $this->controller = "\_404Controller";  
 }  
 $controller = new $this->controller;  
 if(!empty($url[1])){  
 if(method\_exists($controller,$url[1]))  
 {  
 $this->method = $url[1];  
 unset($url[1]);  
 }  
 }  
 call\_user\_func\_array([$controller,$this->method],$url);  
 }  
}

**Klasa kontrolera koji će da poziva odgovarajući View**

<?php  
class Controller  
{  
 public function view($name, $data = [])  
 {  
 if(!empty($data))  
 extract($data);  
 $filename = "../app/views/" . $name . "View.php";  
 if(file\_exists($filename)){  
 require $filename;  
 }  
 else{  
 $filename = "../app/views/404View.php";  
 require $filename;  
 }  
 }  
}

**Klasa osnovnog modela**

<?php  
class Model extends Database  
{  
 protected $table = 'users';  
 public function findAll()  
 {  
 $query = "SELECT \* FROM $this->table";  
 return $this->query($query);  
 }  
 */\*\*  
 \* @param array $data  
 \* @return bool|array  
 \*/* public function where(array $data): bool|array  
 {  
 $keys = array\_keys($data);  
 $query = "SELECT \* FROM $this->table WHERE ";  
 foreach ($keys as $key) {  
 $query .= $key . " = :" . $key;  
 }  
 return $this->query($query, $data);  
 }  
 */\*\*  
 \* @param $data  
 \* @return false  
 \*/* public function insert(array $data): false  
 {  
 if (!empty($this->allowedColumns)) {  
 foreach ($data as $key => $value) {  
 if (!in\_array($key, $this->allowedColumns)) {  
 unset($data[$key]);  
 }  
 }  
 }  
 $keys = array\_keys($data);  
 $query = "INSERT INTO $this->table (" . implode(",", $keys) . ") VALUES (:" . implode(",:", $keys) . ")";  
  
 $this->query($query, $data);  
 return false;  
 }  
 */\*\*  
 \* @param $id  
 \* @param array $data  
 \* @param string $id\_column  
 \* @return false  
 \*/* public function update($id, array $data, string $id\_column = 'id'): false  
 {  
 if (!empty($this->allowedColumns)) {  
 foreach ($data as $key => $value) {  
 if (!in\_array($key, $this->allowedColumns)) {  
 unset($data[$key]);  
 }  
 }  
 }  
 $keys = array\_keys($data);  
 $query = "UPDATE $this->table SET ";  
 foreach ($keys as $key) {  
 $query .= $key . " = :" . $key . ", ";  
 }  
 $query = trim($query, ", ");  
 $query .= " WHERE $id\_column = :id\_value";  
 $data['id\_value'] = $id;  
 $this->query($query, $data);  
 return false;  
 }  
 */\*\*  
 \* @param $id  
 \* @param string $id\_column  
 \* @return false  
 \*/* public function delete($id, string $id\_column = 'id'): false  
 {  
 $query = "DELETE FROM $this->table WHERE $id\_column = :$id\_column";  
 $this->query($query, [$id\_column => $id]);  
 return false;  
 }  
}

**Dodatne korišćene funkcije**

<?php  
use PHPMailer\PHPMailer\PHPMailer;  
use PHPMailer\PHPMailer\SMTP;  
use PHPMailer\PHPMailer\Exception;  
require 'PHPMailer/src/Exception.php';  
require 'PHPMailer/src/PHPMailer.php';  
require 'PHPMailer/src/SMTP.php';  
function redirect($path)  
{  
 header("Location:" . *ROOT* . "/" . $path);  
 die;  
}  
function now(): string  
{  
 return (new DateTime())->format('Y-m-d H:i:s');  
}  
function createToken(int $length): ?string  
{  
 try {  
 return bin2hex(random\_bytes($length));  
 } catch (\Exception $e) {  
 error\_log("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  
 error\_log($e->getMessage());  
 error\_log("file:" . $e->getFile() . " line:" . $e->getLine());  
 return null;  
 }  
}  
function test\_input($data) {  
 $data = trim($data);  
 $data = stripslashes($data);  
 $data = htmlspecialchars($data);  
 return $data;  
}  
function sendMail(string $email, string $action, $token)  
{  
 $mail = new PHPMailer(true);  
 $link = $action === 'activate' ? *ROOT* . "/activate?token=" . $token : *ROOT* . "/reset?token=" . $token;  
 try {  
 $mail->SMTPDebug = SMTP::*DEBUG\_SERVER*;  
 $mail->isSMTP();  
// $mail->Host = 'mail.nm.stud.vts.su.ac.rs';  
 $mail->Host = 'sandbox.smtp.mailtrap.io';  
 $mail->SMTPAuth = true;  
// $mail->Username = 'nm';  
// $mail->Password = 'u76tVYucyaaf3lI';  
 $mail->SMTPSecure = 'tls';  
// $mail->Port = 587;  
 $mail->Port = 2525;  
 $mail->Username = '57d4b84238770b';  
 $mail->Password = 'aa9bb4c650ae89';  
 $mail->setFrom('nm@nm.stud.vts.su.ac.rs', 'Webmaster');  
 $mail->addAddress($email);  
 $mail->isHTML(true);  
 $mail->Subject = $action === 'activate' ? "Activate your account" : "Reset your password";  
 $mail->Body = ($action === 'activate') ? "<h1>Please activate your account</h1><br><p>To activate your account please click <a href='$link'>here</a>.</p>" : "<h1>Reset password</h1><p>To reset your password please click <a href='$link'>here</a>.</p>";  
 $mail->AltBody = 'Došlo do još jednog neuspešnog prijavljivanja.';  
 $mail->send();  
 } catch (Exception $e) {  
 echo "Message could not be sent. Mailer error {$mail->ErrorInfo}";  
 }  
}

**Api kontroler**

<?php  
class ApiController extends Controller  
{  
 public function index(): void  
 {  
 }  
 public function exercises(): void  
 {  
 $exercise = new Exercise();  
 $exercises = $exercise->findAll();  
 echo json\_encode(['exercises' => $exercises], *JSON\_THROW\_ON\_ERROR* | *JSON\_PRETTY\_PRINT*);  
 }  
 public function categories(): void  
 {  
 $category = new Category();  
 $categories = $category->findAll();  
 echo json\_encode(['category' => $categories], *JSON\_THROW\_ON\_ERROR* | *JSON\_PRETTY\_PRINT*);  
 }  
 public function publicWorkouts(): void  
 {  
 $query = $\_GET['query'] ?? '';  
 $category = $\_GET['category'] ?? '';  
 try {  
 $trainings = new Training();  
 $workouts = $trainings->findAllPublicSearchWithJoins('public', $query, $category);  
 header('Content-Type: application/json');  
 echo json\_encode($workouts, *JSON\_THROW\_ON\_ERROR*);  
 } catch (PDOException $e) {  
 echo json\_encode(['error' => $e->getMessage()]);  
 }  
 }  
 public function addCategory(): void  
 {  
 $category = new Category();  
 $response = ['success' => false, 'message' => ''];  
 if ($\_SERVER["REQUEST\_METHOD"] === "POST") {  
 $title = trim($\_POST['categoryName']);  
 if (!empty($title)) {  
 $category->insert(['name' => $title]);  
 $response['success'] = true;  
 $response['message'] = 'Success';  
 } else {  
 $response['message'] = 'Category name cannot be empty.';  
 }  
 } else {  
 $response['message'] = 'Invalid request method.';  
 }  
 echo json\_encode($response, *JSON\_THROW\_ON\_ERROR*);  
 }  
 public function addExercise(): void  
 {  
 $execise = new Exercise();  
 $response = ['success' => false, 'message' => ''];  
 if ($\_SERVER["REQUEST\_METHOD"] === "POST") {  
 $exerciseTitle = test\_input($\_POST['exerciseTitle']);  
 $exerciseDesc = test\_input($\_POST['exerciseDesc']);  
 $duration = test\_input($\_POST['exerciseDuration']);  
 $photoUrl = test\_input($\_POST['photo\_url']);  
 $videoUrl = test\_input($\_POST['video\_url']);  
 if (!empty($exerciseTitle) && !empty($exerciseDesc) && !empty($photoUrl) && !empty($duration)) {  
 $execise->insert(['title' => $exerciseTitle, 'description' => $exerciseDesc,'duration' => $duration, 'image\_url' => $photoUrl, 'video\_url' => $videoUrl]);  
 $response['success'] = true;  
 $response['message'] = 'Success';  
 } else {  
 $response['message'] = 'Please fill out all fields (video link url is optional).';  
 }  
 } else {  
 $response['message'] = 'Invalid request method.';  
 }  
 echo json\_encode($response, *JSON\_THROW\_ON\_ERROR*);  
 }  
}

**Model za treninge**

<?php  
class Training extends Model  
{  
 protected $table = 'workouts';  
 public $errors = [];  
 protected $allowedColumns = [  
 'title',  
 'description',  
 'duration\_value',  
 'duration\_unit',  
 'id',  
 'category\_id',  
 'privacy',  
 'workout\_image'  
 ];  
 public function findAllWithJoins()  
 {  
 $query = "SELECT w.workout\_id,title,description,u.name,surname,email,duration\_value,duration\_unit,c.name AS category\_name FROM $this->table w INNER JOIN users u ON w.id = u.id INNER JOIN category c ON c.category\_id = w.category\_id";  
 return $this->query($query);  
 }  
 public function findAllPublicWithJoins($privacy): bool|array  
 {  
 $query = "SELECT w.workout\_id,title,description,u.name,surname,email,duration\_value,duration\_unit,c.name AS category\_name FROM $this->table w INNER JOIN users u ON w.id = u.id INNER JOIN category c ON c.category\_id = w.category\_id WHERE w.privacy = :privacy";  
 return $this->query($query,['privacy' => $privacy]);  
 }  
 public function findAllPublicSearchWithJoins($privacy,$searchQuery,$category): bool|array  
 {  
 $query = "SELECT w.workout\_id,w.workout\_image,title,description,u.name,surname,email,duration\_value,duration\_unit,c.name AS category\_name FROM $this->table w INNER JOIN users u ON w.id = u.id INNER JOIN category c ON c.category\_id = w.category\_id WHERE (w.privacy = :privacy) AND (w.title LIKE :query OR w.description LIKE :query) AND (:category = '' OR c.name = :category)";  
 return $this->query($query,['privacy' => $privacy, 'query' => "%". $searchQuery ."%", 'category' => $category]);  
 }  
 public function findAllPrivateWithJoins($id)  
 {  
 $query = "SELECT w.workout\_id,title,description,u.name,surname,email,duration\_value,duration\_unit,c.name AS category\_name FROM $this->table w INNER JOIN users u ON w.id = u.id INNER JOIN category c ON c.category\_id = w.category\_id WHERE w.id = :id";  
 return $this->query($query,['id' => $id]);  
 }  
 public function findWhereIdWithJoinsDetailed(int $id)  
 {  
 $query = "SELECT w.title,w.description,e.title AS e\_title,e.description AS exercise\_desc,w.duration\_value,w.duration\_unit,e.duration AS exercise\_duration,e.video\_url AS video\_link,e.image\_url,we.day\_of\_week FROM `workoutexercises` we INNER JOIN workouts w ON we.workout\_id = w.workout\_id INNER JOIN exerecise e ON e.exercise\_id = we.exercise\_id WHERE w.workout\_id = :id; ";  
 return $this->query($query,['id' => $id]);  
 }  
}

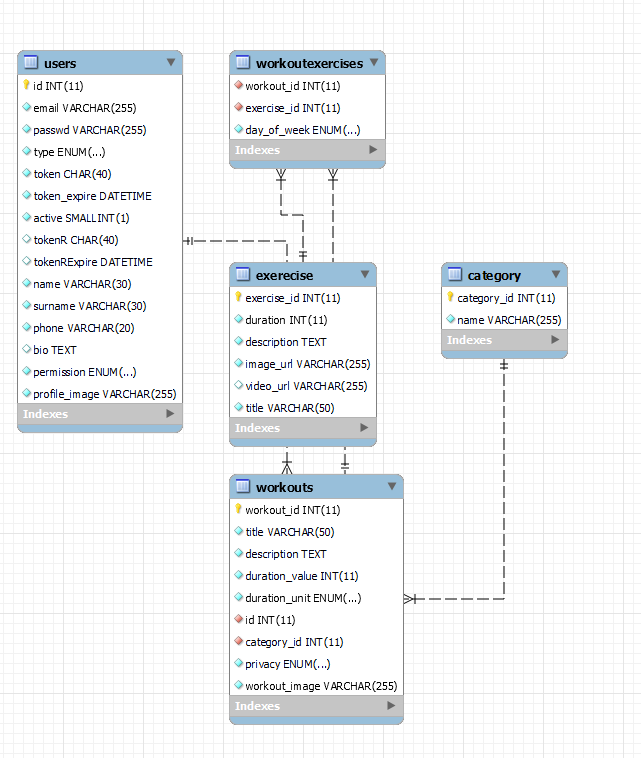
**Skripta koja fetchuje treninge na osnovu zadatih parametara**

*document*.addEventListener('DOMContentLoaded', () => {  
 const searchInput = *document*.getElementById('searchInput');  
 const filterButtons = *document*.getElementById('filterButtons');  
 const workoutContainer = *document*.getElementById('workoutContainer');  
 const resetFiltersButton = *document*.getElementById('resetFilters');  
  
 let selectedCategory = '';  
  
 const fetchWorkouts = async (query = '', category = '') => {  
 try {  
 const response = await fetch(`${*ROOT\_URL*}/api/publicWorkouts?query=${query}&category=${category}`);  
 const workouts = await response.json();  
 renderWorkouts(workouts);  
 } catch (error) {  
 *console*.error('Error fetching workouts:', error);  
 }  
 };  
  
 //Renders workout cards  
 const renderWorkouts = (workouts) => {  
 workoutContainer.innerHTML = '';  
 workouts.forEach(workout => {  
 const workoutCard = `  
 <div class="card train train-hover shadow" style="width: 18rem; text-decoration: none;">  
 <a href="${*ROOT\_URL*}/trainings/workout?id=${workout.workout\_id}">  
 <img  
 src="${*ROOT\_URL*}/assets/images/Train/${workout.workout\_image}"  
 class="card-img-top train-img" alt="text">  
 <div class="card-body">  
 <h2 class="text-center text-dark">${workout.title}</h2>  
 <p class="text-dark">${workout.description.split('.')[0]}.</p>  
 <div class="d-flex flex-wrap gap-1 ">  
 <p class="badge rounded-pill text-bg-warning fs-6">${workout.category\_name}</p>  
 </div>  
 <div class="d-flex gap-1 justify-content-end align-items-center my-auto">  
 <i class="bi bi-alarm-fill text-dark"></i>  
 <div class="d-flex gap-2">  
 <span class="text-dark fw-bold">Duration</span>  
 <span class="text-dark">${workout.duration\_value} ${workout.duration\_unit}</span>  
 </div>  
 </div>  
 <div class="d-flex gap-2 justify-content-end">  
 <span class="text-dark">Made by</span>  
 <a href="${*ROOT\_URL*}/home/profile?id=${workout.email}" class="link-secondary">${workout.name} ${workout.surname}</a>  
 </div>  
 </div>  
 </a>  
 </div>  
 `;  
 workoutContainer.insertAdjacentHTML('beforeend', workoutCard);// Inserts card into container  
 });  
 };  
 // Fetches all workouts based on search input changes  
 searchInput.addEventListener('input', () => {  
 const query = searchInput.value;  
 fetchWorkouts(query);  
 });  
 // Fetches workouts based on selected filter  
 filterButtons.addEventListener('click', (event) => {  
 if (event.target.classList.contains('filter')) {  
 selectedCategory = event.target.dataset.category;  
 fetchWorkouts(searchInput.value, selectedCategory);  
 }  
 });  
 // Resets filter buttons  
 resetFiltersButton.addEventListener('click', () => {  
 selectedCategory = '';  
 fetchWorkouts(searchInput.value);  
 });  
 // Fetches all workouts on initial load  
 fetchWorkouts();  
});

**Skripta koja vrši proveru podataka pri registraciji na klijenskoj strani**

*document*.getElementById('registerForm').addEventListener('submit', function(event) {  
 const name = *document*.getElementById('name').value.trim();  
 const surname = *document*.getElementById('surname').value.trim();  
 const email = *document*.getElementById('email').value.trim();  
 const phone = *document*.getElementById('phone').value.trim();  
 const password = *document*.getElementById('passwd').value.trim();  
 const passwordR = *document*.getElementById('passwdR').value.trim();  
 let errorMessage = '';  
 if (!name) {  
 errorMessage += 'Name is required.<br>';  
 }  
 if (!surname) {  
 errorMessage += 'Surname is required.<br>';  
 }  
 if (!email) {  
 errorMessage += 'Email is required.<br>';  
 }  
 if (!phone) {  
 errorMessage += 'Phone is required.<br>';  
 }  
 if (!password) {  
 errorMessage += 'Password is required.<br>';  
 } else if (password.length < 8) {  
 errorMessage += 'Password must be at least 8 characters long.<br>';  
 }  
 if (!passwordR) {  
 errorMessage += 'Confirm password is required.<br>';  
 } else if (password !== passwordR) {  
 errorMessage += 'Passwords do not match.<br>';  
 }  
 if (errorMessage) {  
 event.preventDefault();  
 const alertDiv = *document*.createElement('div');  
 alertDiv.className = 'alert alert-danger';  
 alertDiv.innerHTML = errorMessage;  
 const form = *document*.getElementById('registerForm');  
 const existingAlert = form.querySelector('.alert');  
 if (existingAlert) {  
 existingAlert.remove();  
 }  
 form.insertBefore(alertDiv, form.firstChild);  
 }  
});

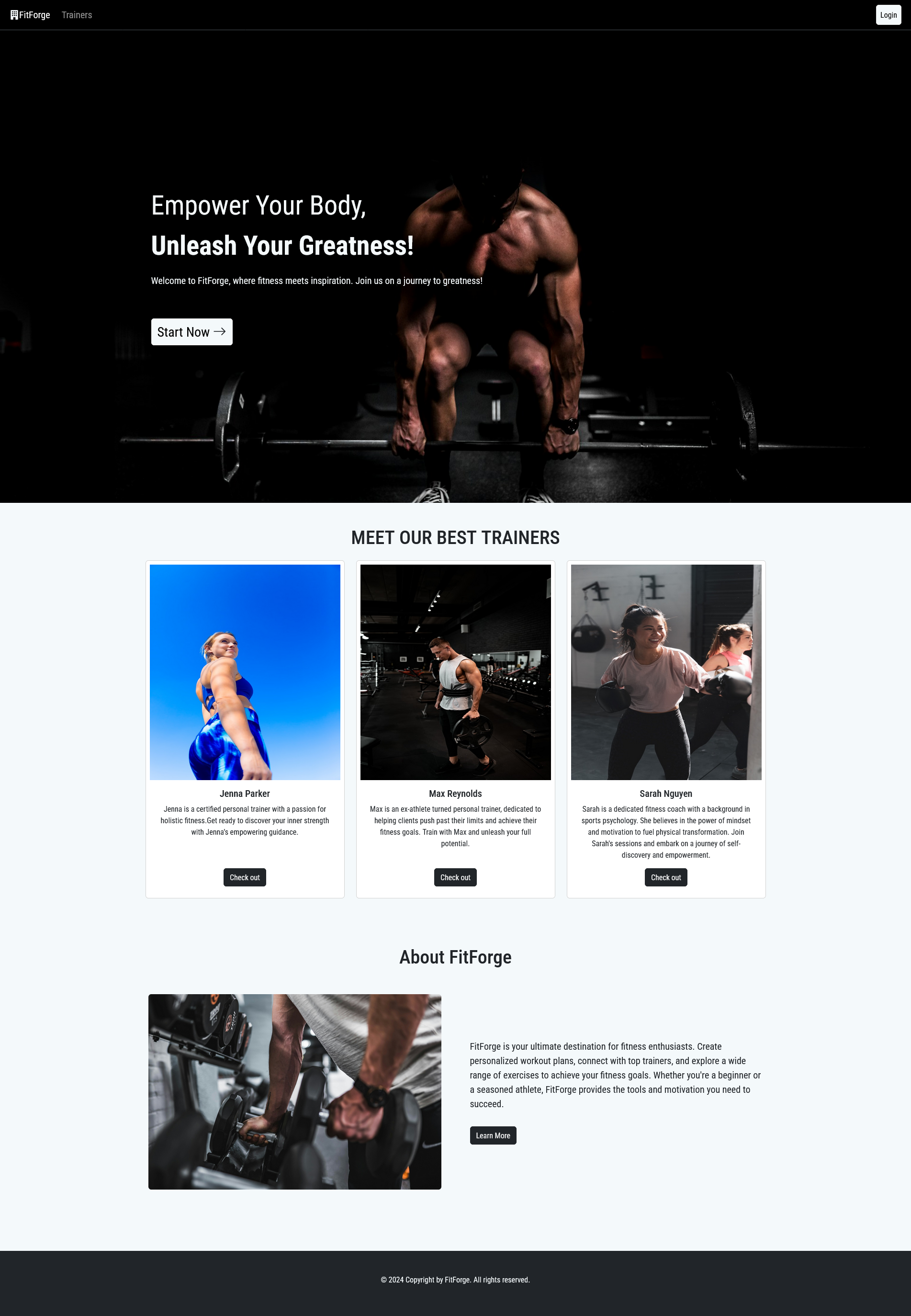
# Struktura baze podataka (MySQL Workbench)

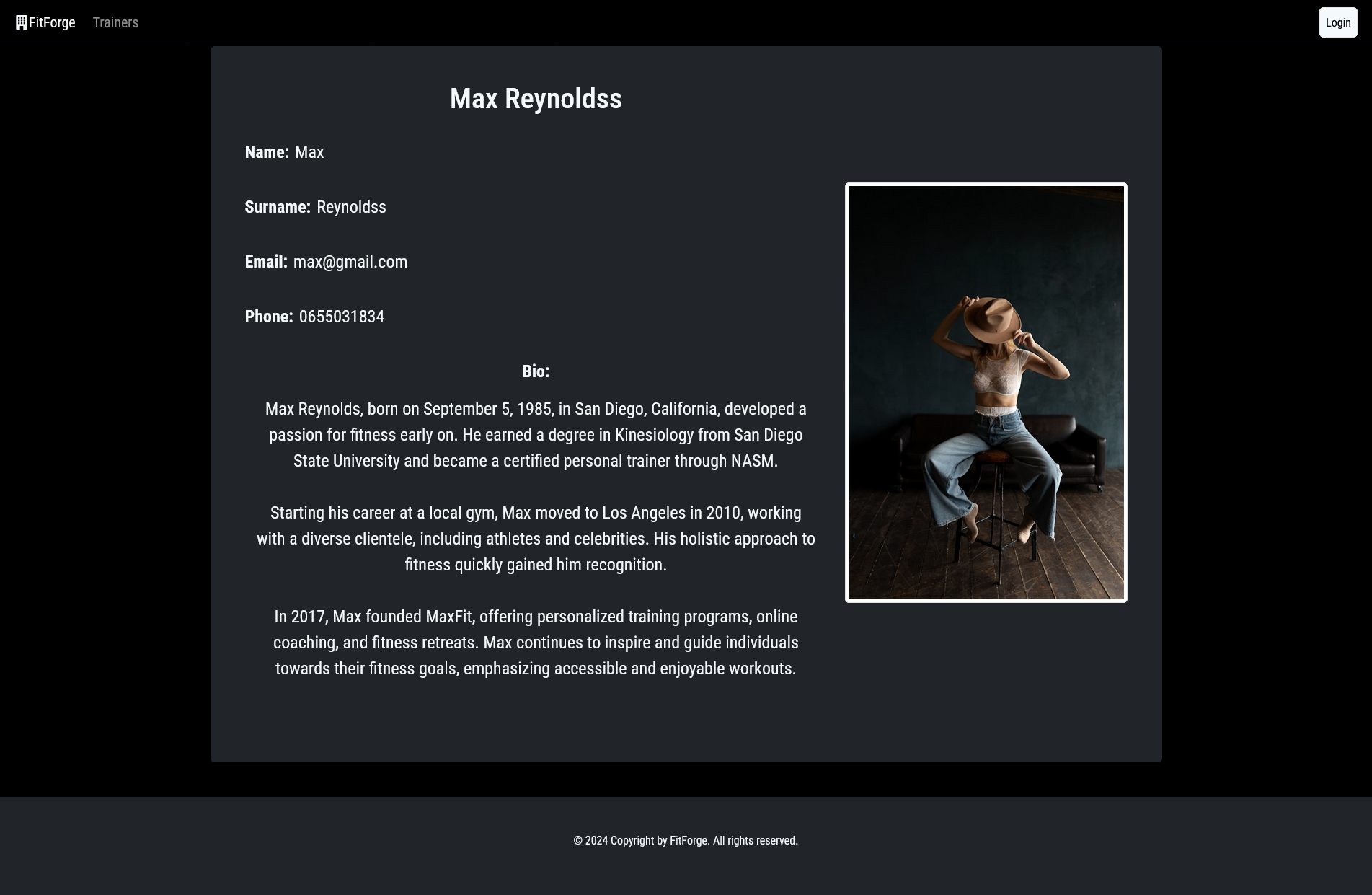


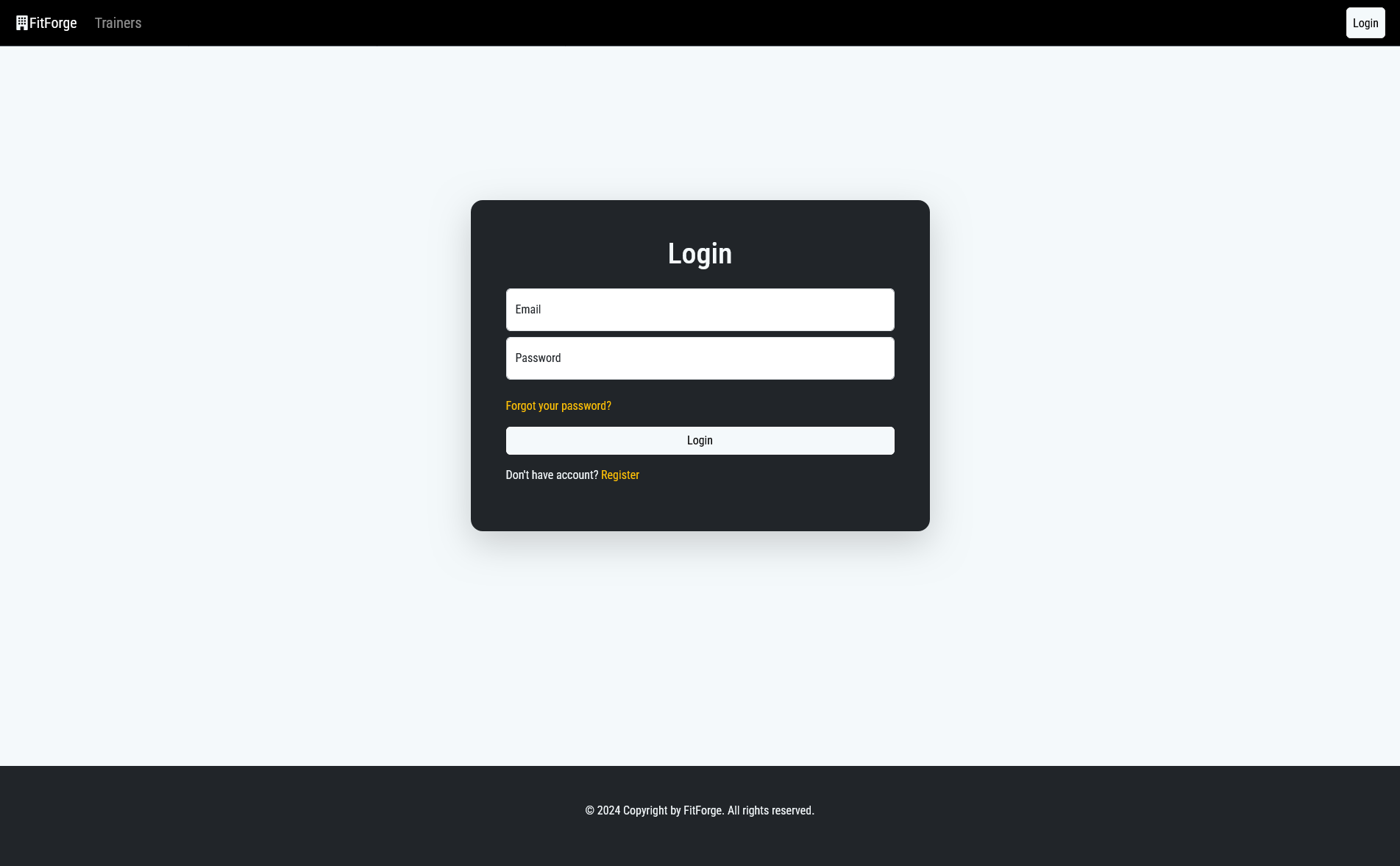
# Opis funkcionalnosti

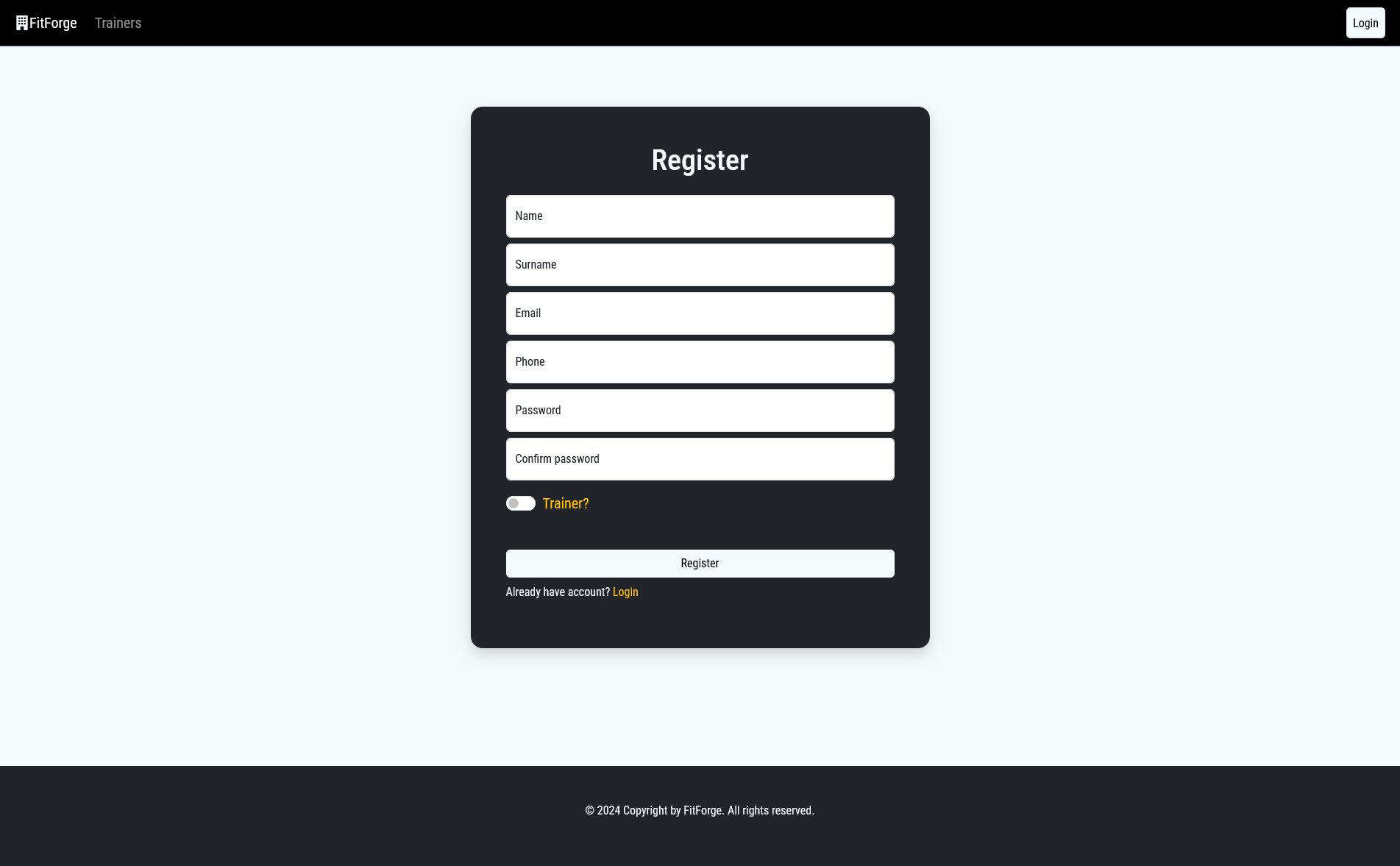
1. **Početna strana i forma za prijavljivanje**

* **Posetilac** dolazi na sajt gde ga dočekuje osnovne informacije o uslugama sajta, takođe ima mogućnost pogleda prijavljenih trenera
* **Opcija prijave**
* **Ukoliko korisnik nema nalog,** popunjava Register formu za prijavu nakon koje dobija mejl sa aktivacionim linkom na adresu unetu u fromu i tek onda može da pristupi svim mogućnostima sajta
* **Ukoliko korisnik ima nalog** dovoljna je prijava na login Formu





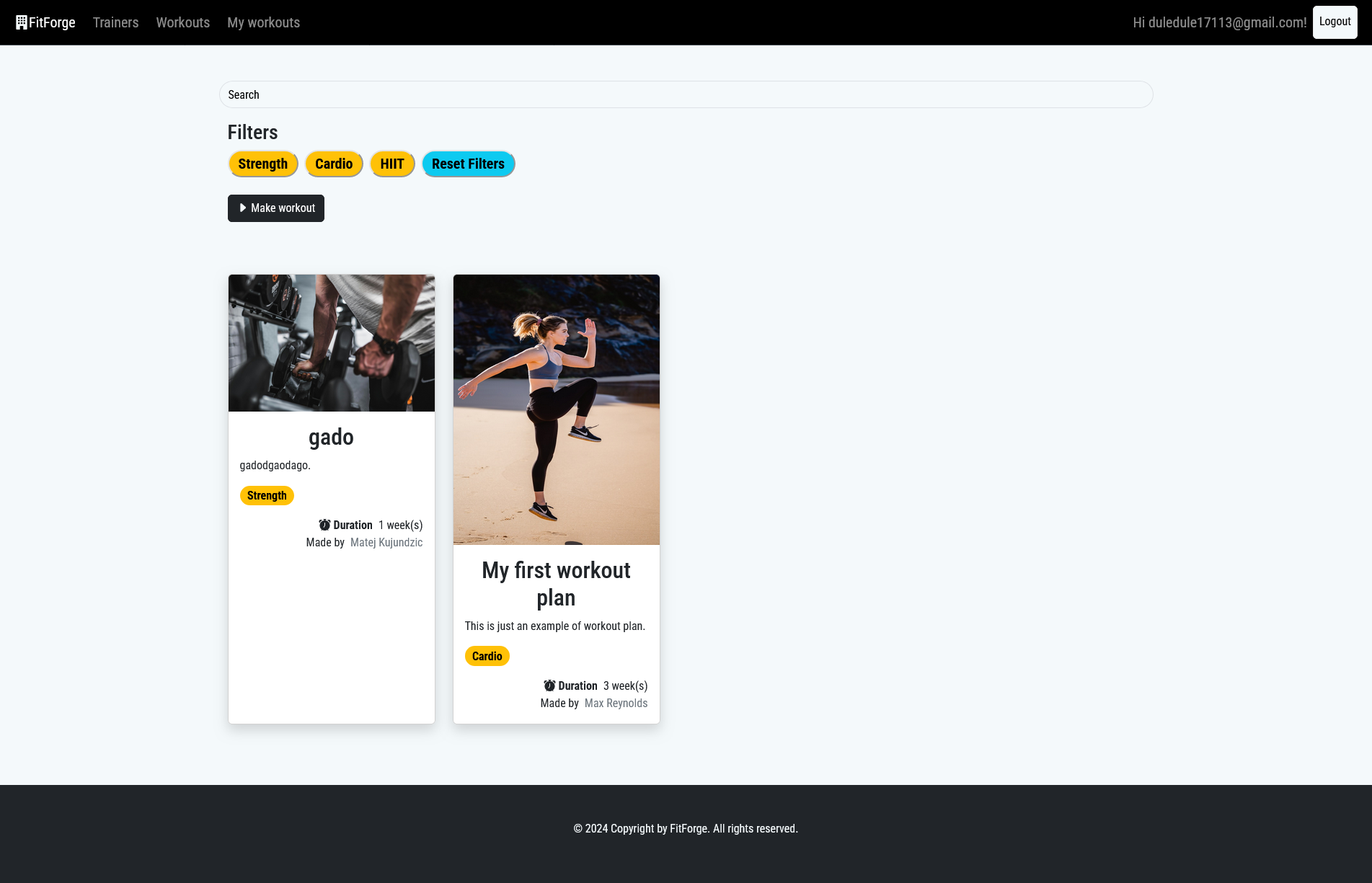


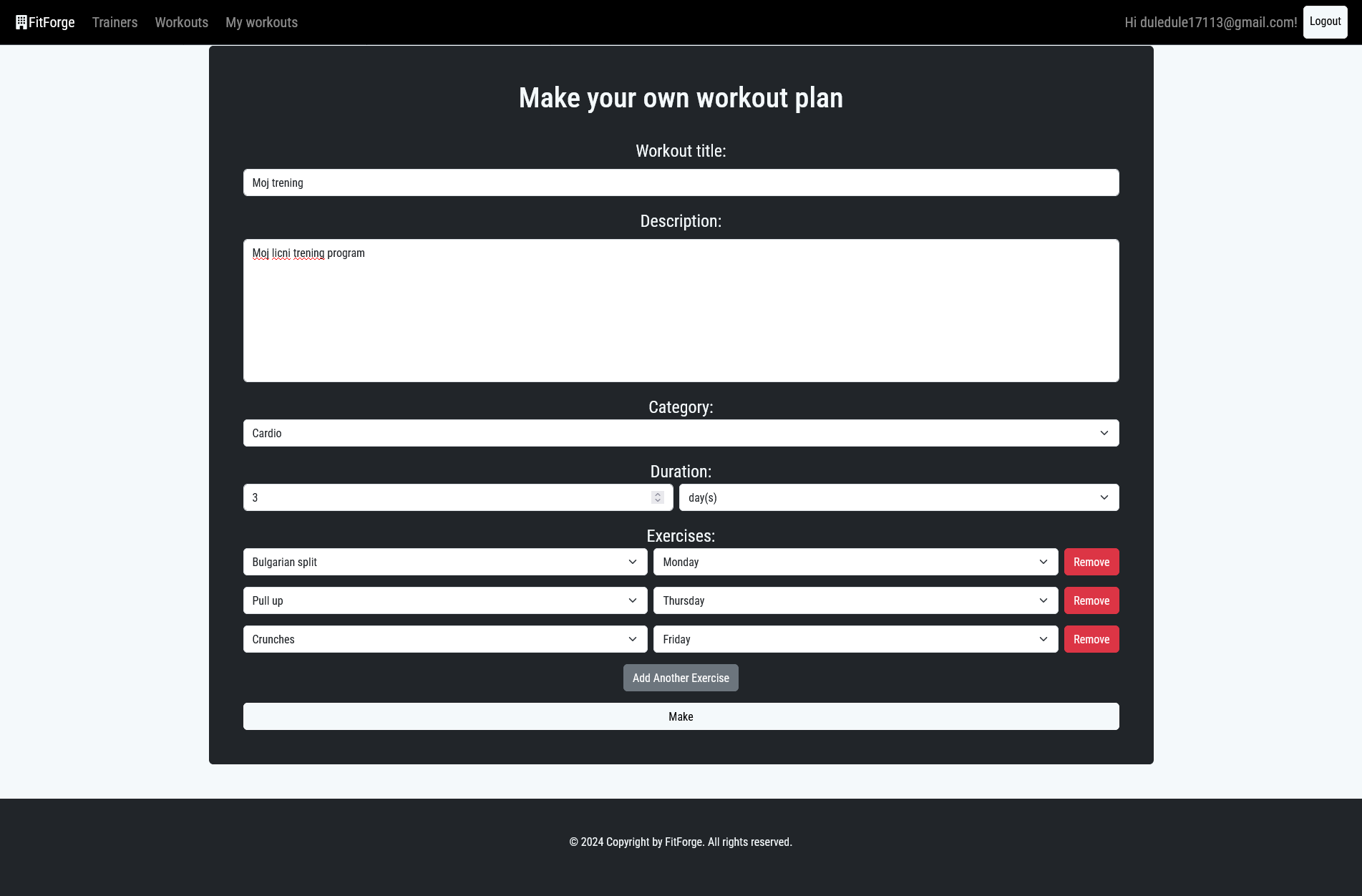


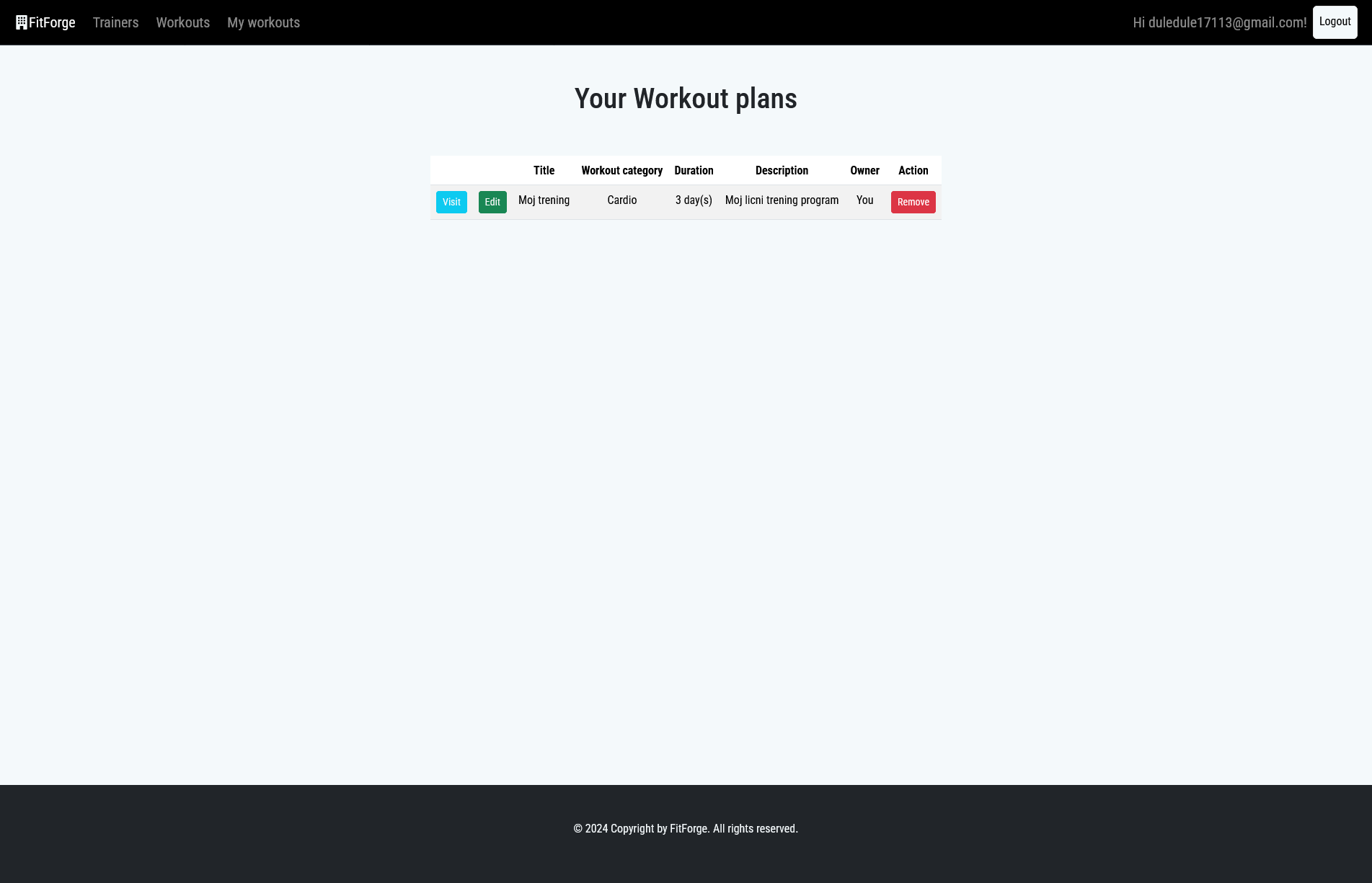
Obe forme vrše temelju proveru podataka kako na serverskoj tako i na klijentskoj strani, postoji zaštita od SQL injectiona i za rad sa bazom se koristi PDO ekstenzija.

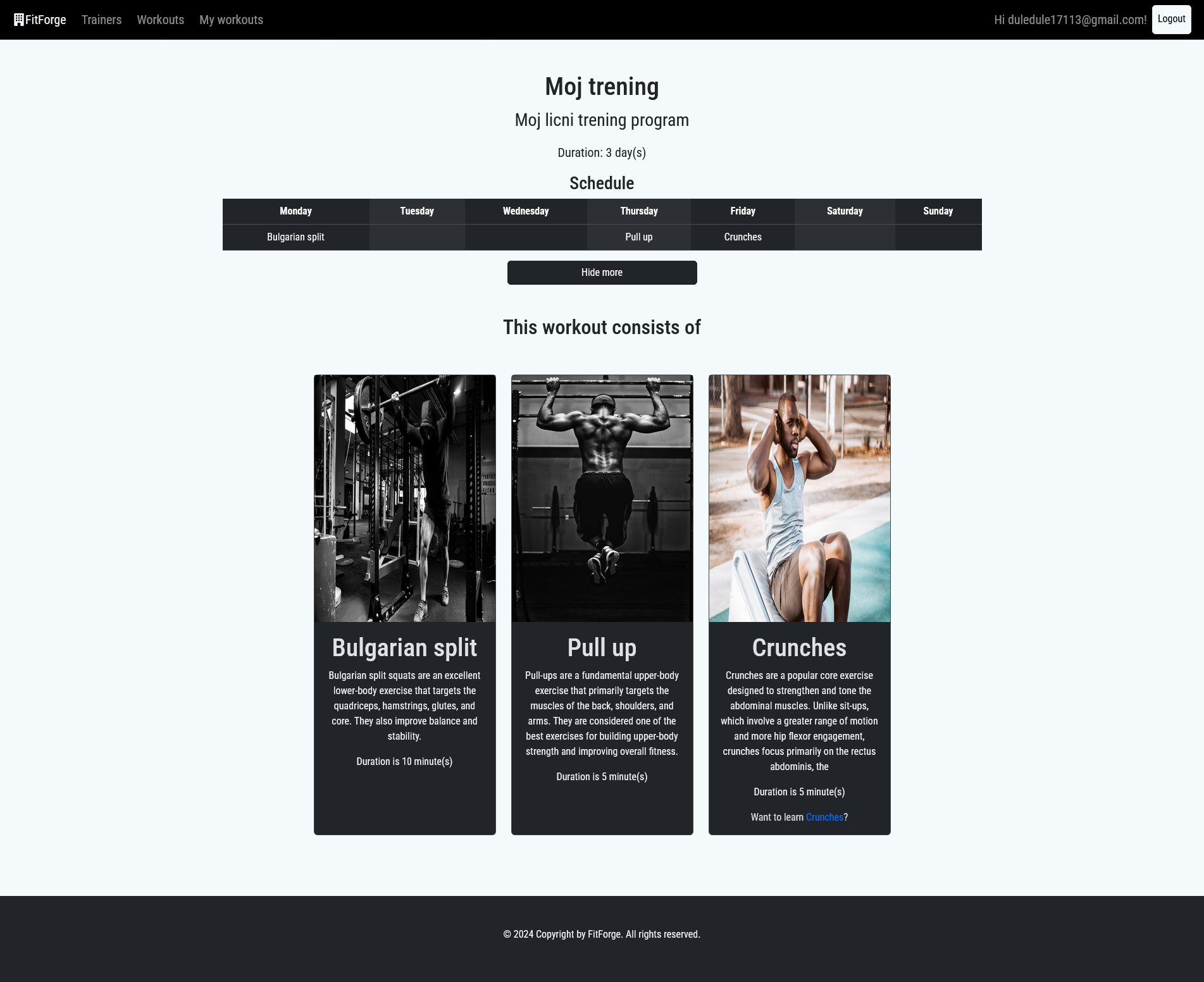
1. **Uspešna registracija korisnika**

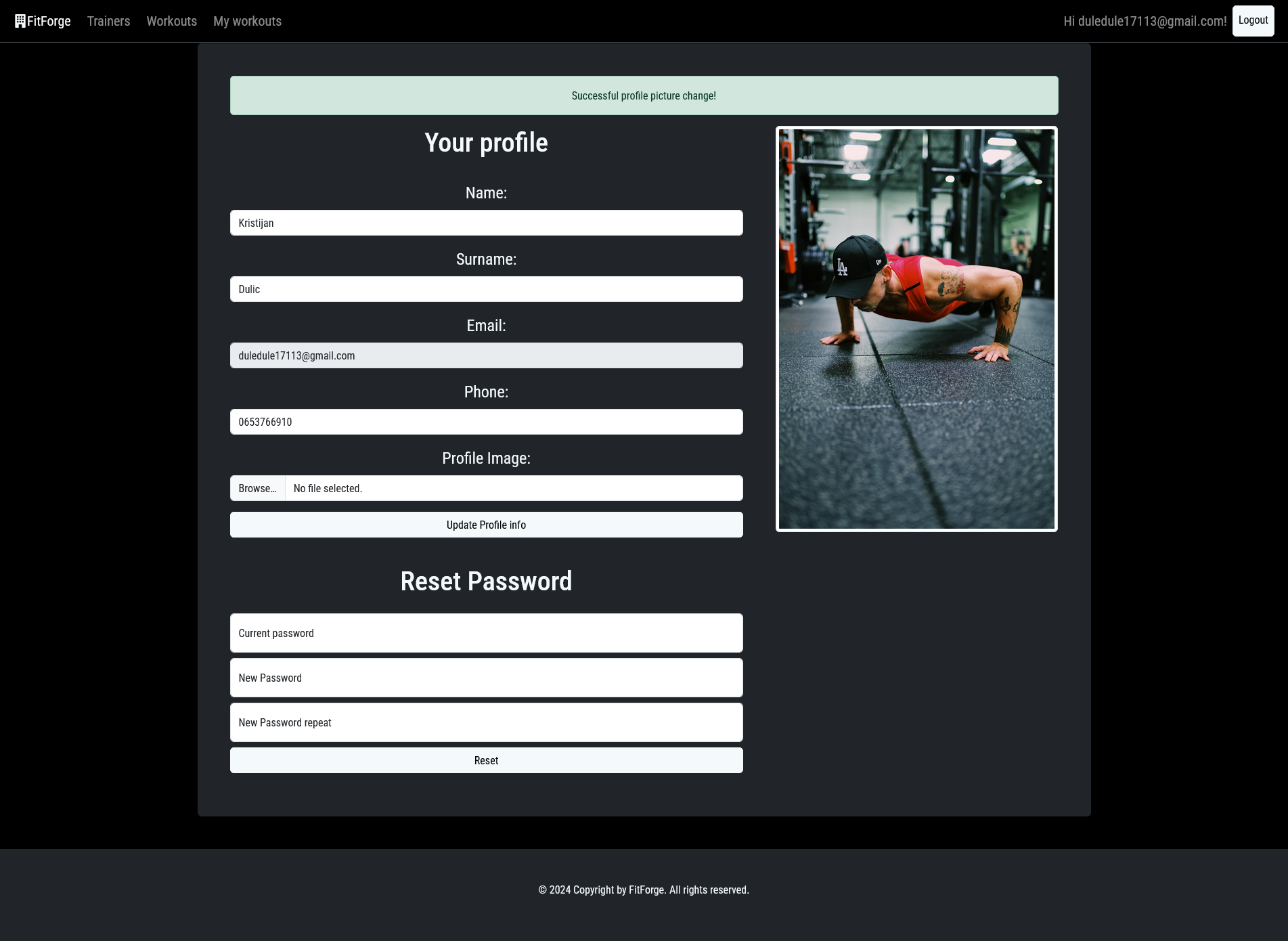
* **Korisnik** dobija jasan pregled svih dostupnih trening programa koje može da pretražuje opciom Search ili sa filter dugmadima
* **Korisnik** je u mogućnosti da kreira svoj lični trening program
* **Pregled svojih trening programa** je takođe moguć odlaskom na tab My Workouts
* **Korisniku** se još i otvara mogućnost promene svojih profilnih podataka takođe može da menja i lozinku







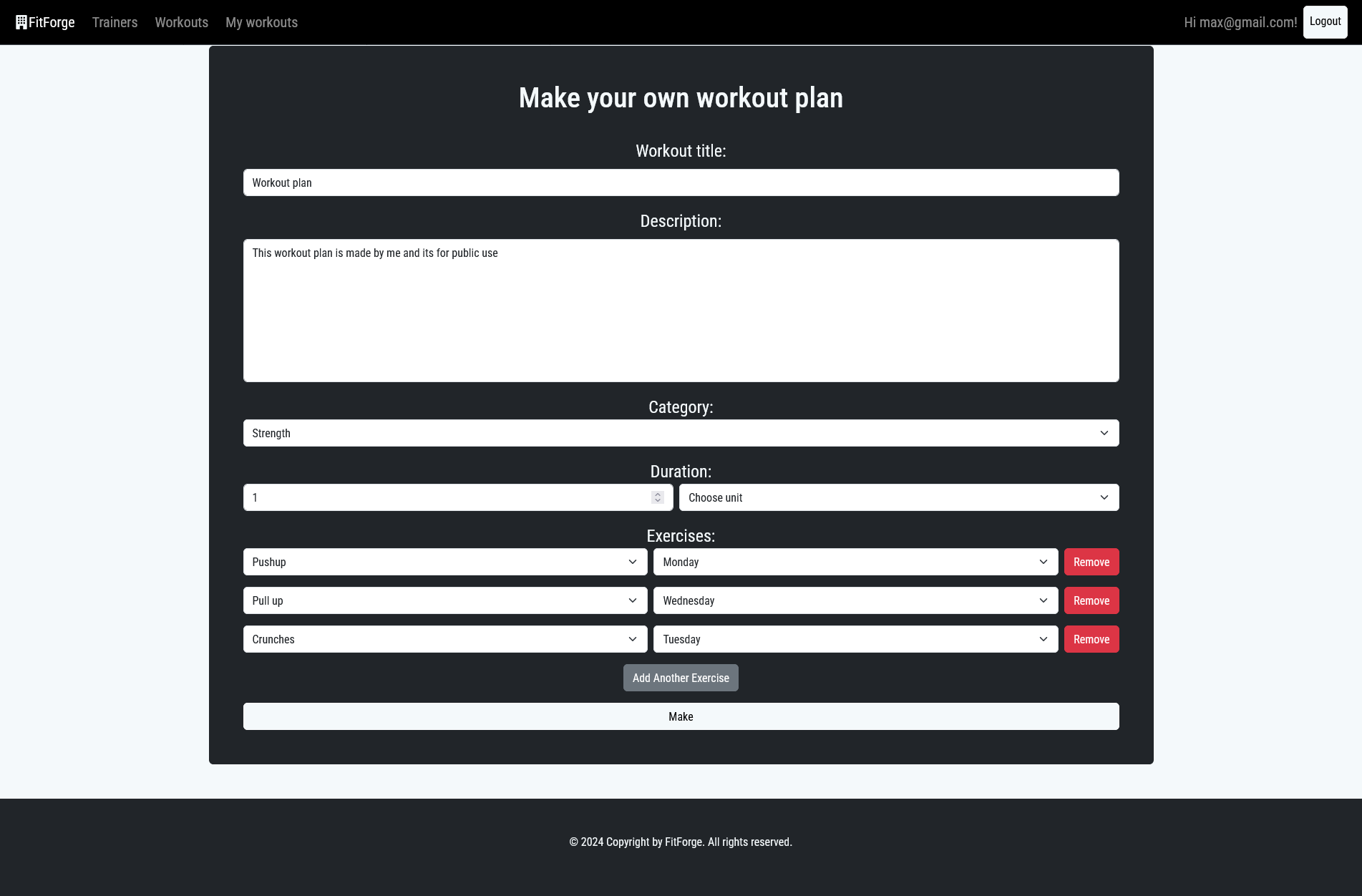


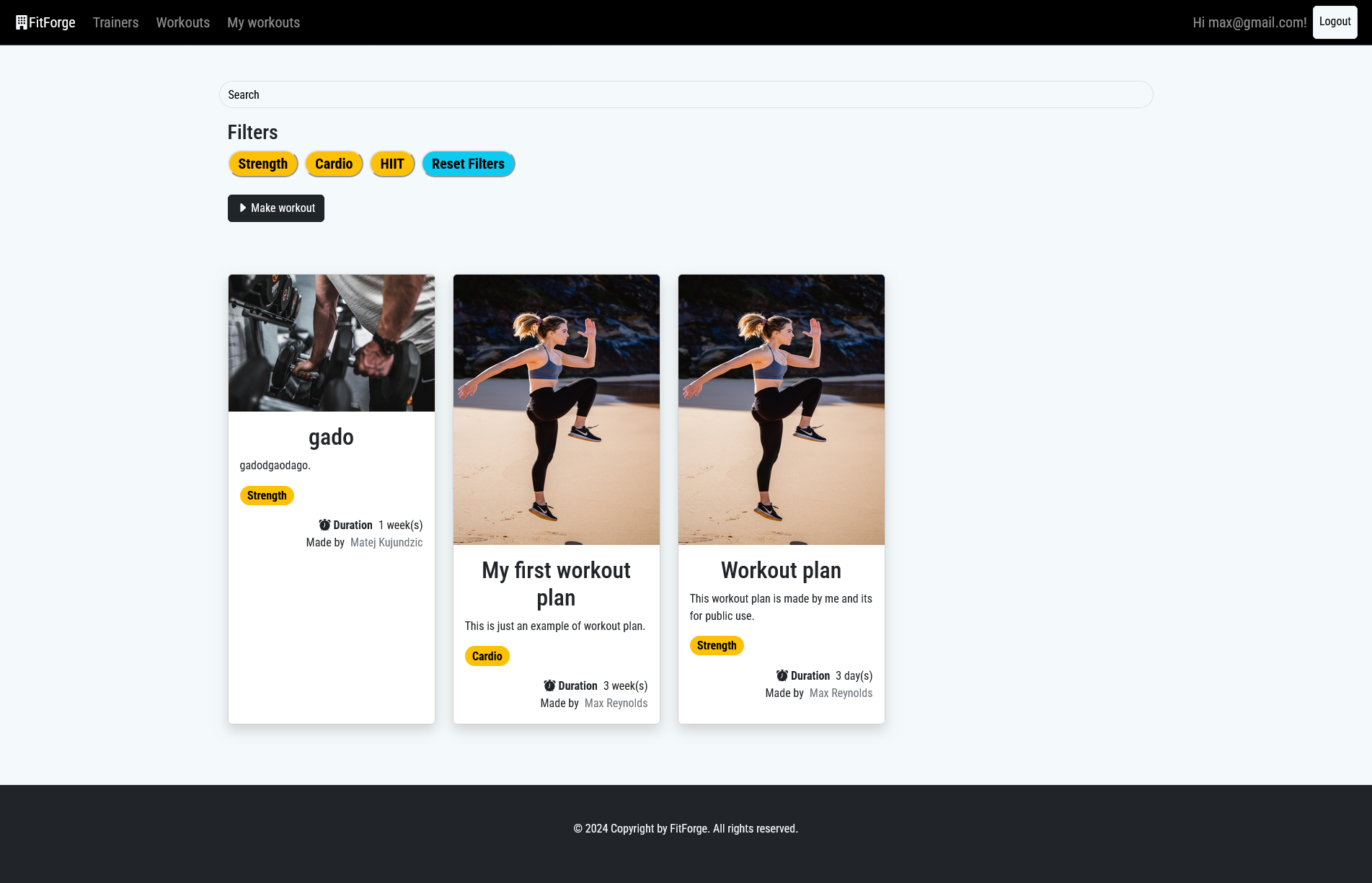


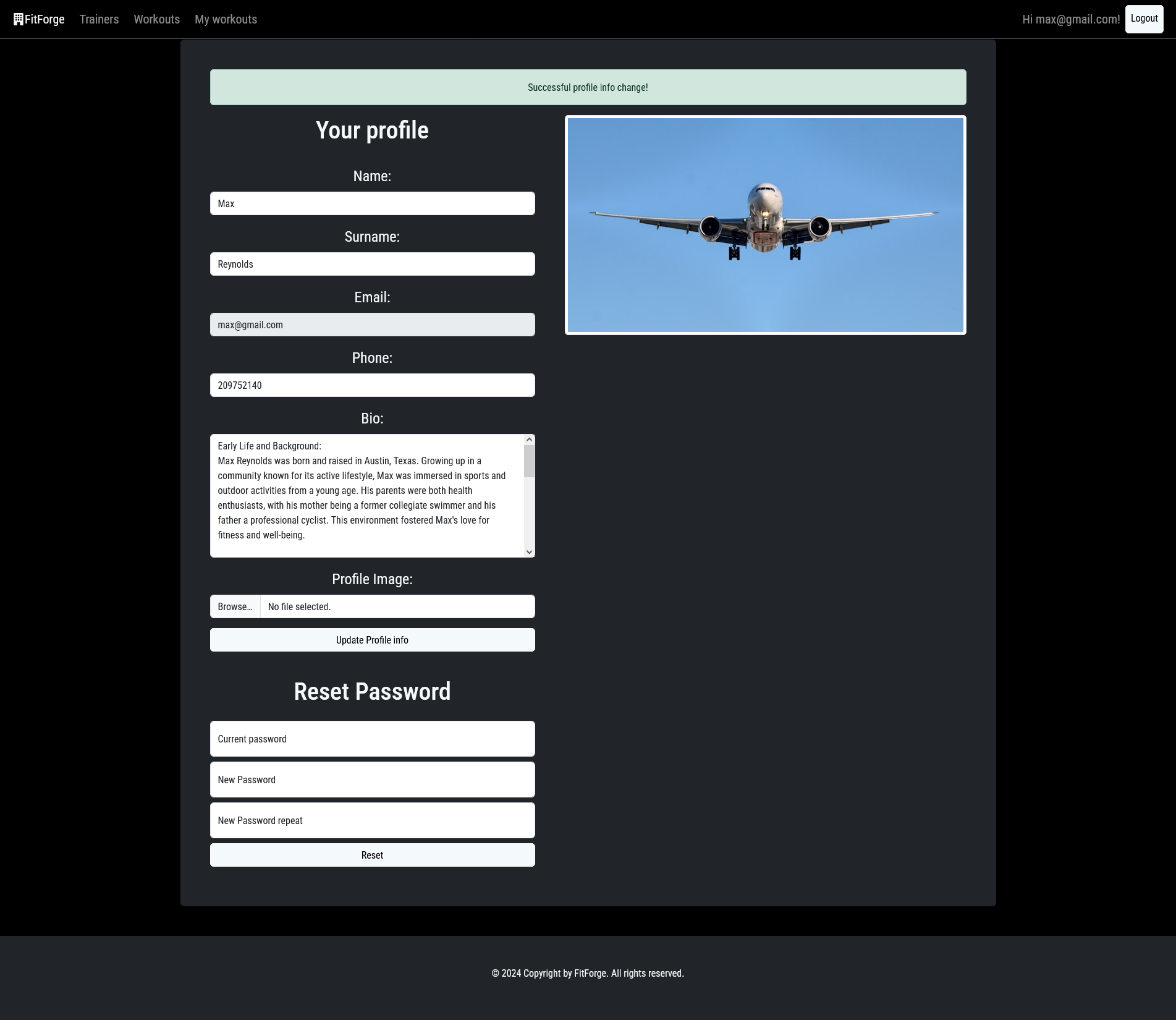
Dobavljanje rezultata pretrage obavljeno je sa Fetch API tehnikom sa samostalno definisanog API endpointa, unos podataka prilikom promene profilnih podataka i prilikom kreiranja treninga se proverava i na serverskoj i na klientskoj strani.

1. **Uspešna registracija korisnika - trenera**

* Nakon što se korisnik uspešno prijavio kao **trener** može da kreira trening programe koji su dostupni za javnu upotrebu
* **U mogućnosti je da menja svoje profilne podatke** ali za razliku od običnih korisnika, trenerima je preporučeno da dodaju i svoju biografiju

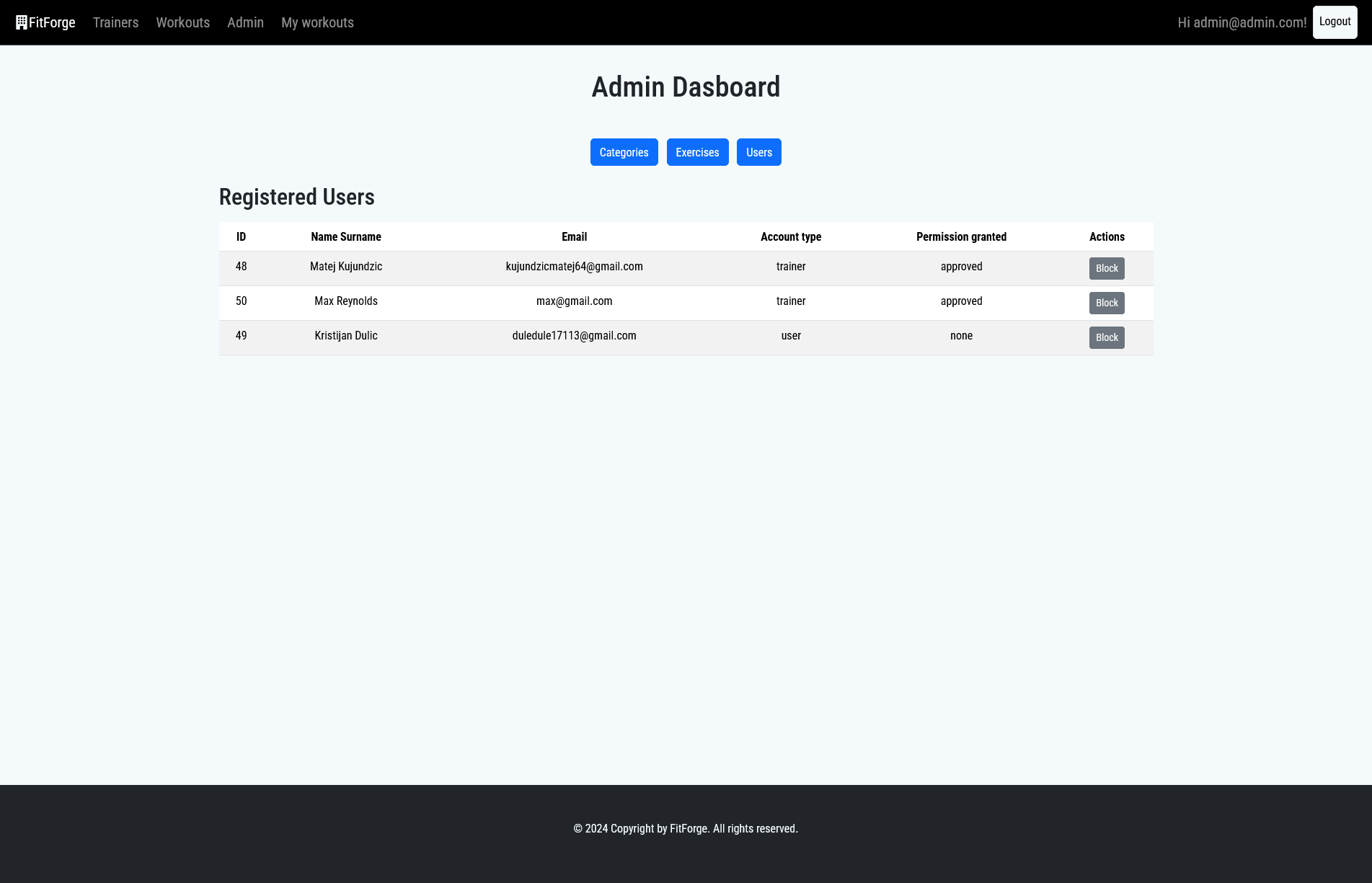
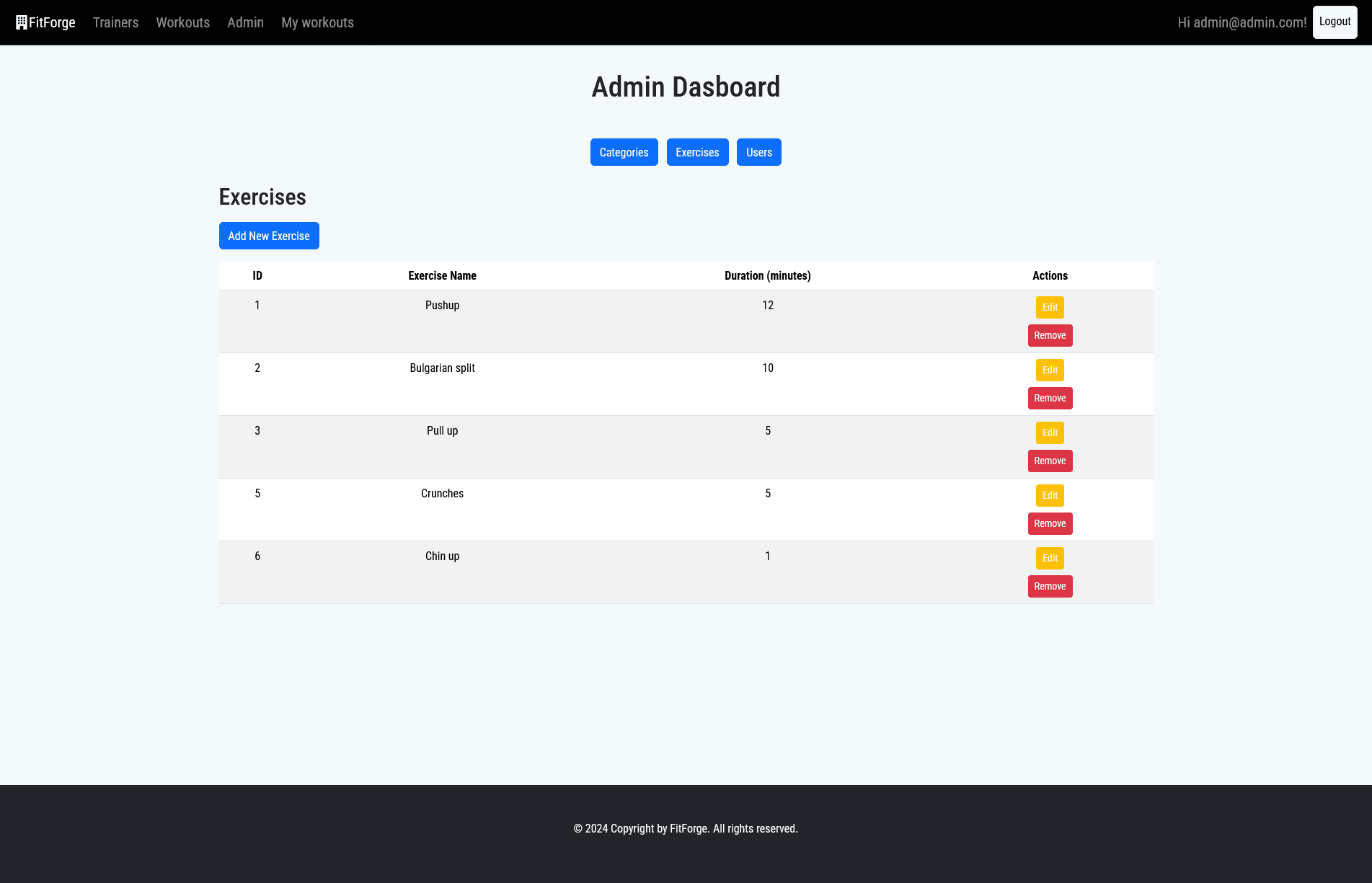
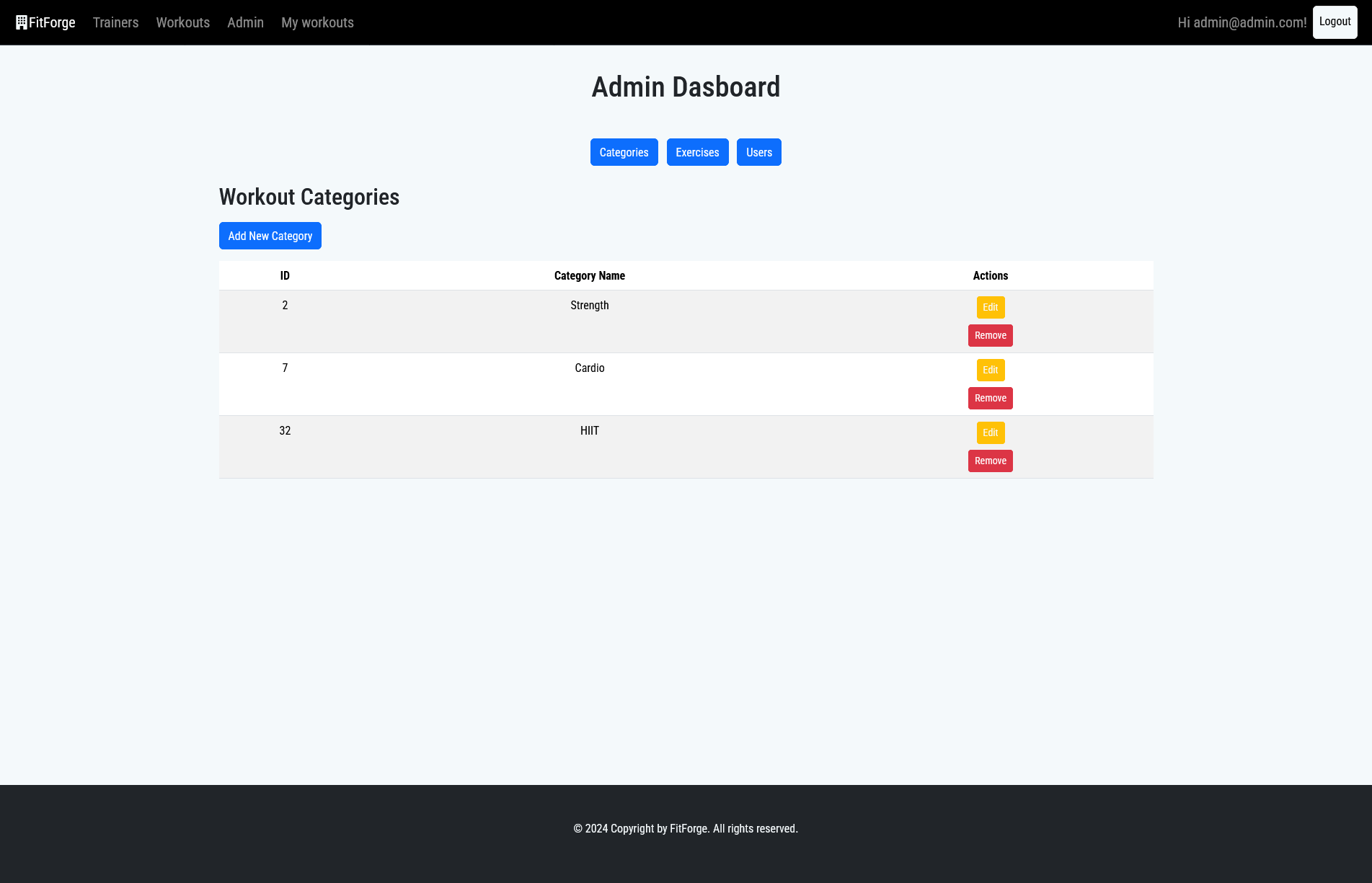


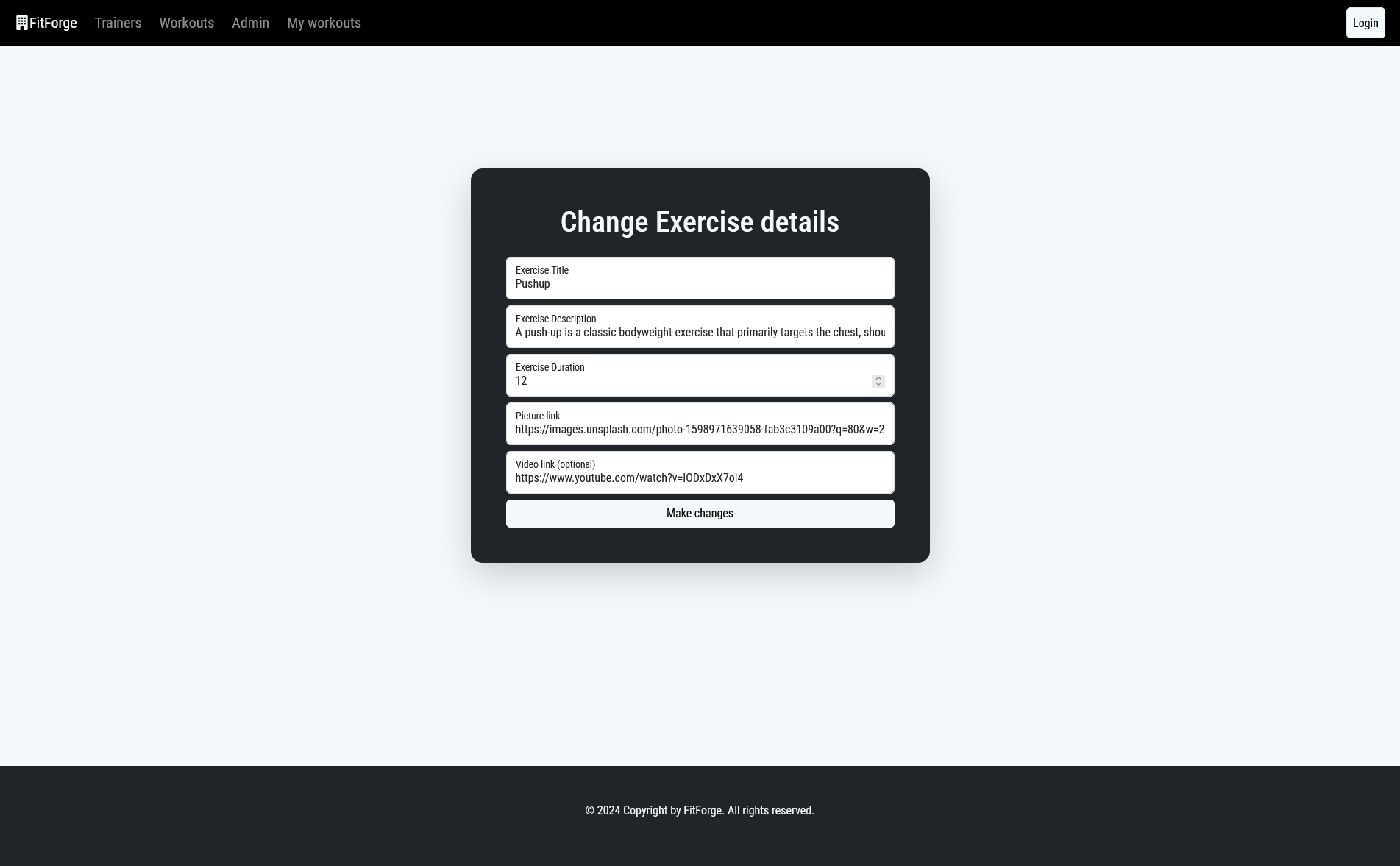




1. **Administratorski deo**

* **Admin** ima mogućnost pristupa administratorskom panelu
* **Može da odobrava ili zabranjuje** pristup logovanju na skorisnicima
* **Može da dodaje, briše i menja** kategorije kao i vežbe





Podaci se takođe fetchuju sa Fetch API-em, admin panel je zaštićen PHP sesijama koje zabranjuju pristup ovo delu sajta svima koji nisu registrovani kao admin.

# Korišćena literatura

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

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

W3. <https://www.php.net/>

W4. <https://getbootstrap.com/>

W5. <https://www.youtube.com/watch?v=q0JhJBYi4sw>

W6. <https://developer.mozilla.org/en-US/>