

**VÁCI SZAKKÉPZÉSI CENTRUM
BORONKAY GYÖRGY
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM**

VIZSGAREMEK

Fajta Jordán, Mártai Balázs

2024.

VÁCI SZAKKÉPZÉSI CENTRUM
BORONKAY GYÖRGY
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM



VIZSGAREMEK

FitNet

Konzulens: Wiesel Csaba

Készítette: Fajta Jordán
Mártai Balázs

Hallgatói nyilatkozat

Alulírottak, ezúton kijelentjük, hogy a szakdolgozat saját, önálló munkánk, és korábban még sehol nem került publikálásra.

Fajta Jordán

Mártai Balázs

Konzultációs lap

Vizsgázók neve: Fajta Jordán, Mártai Balázs

Vizsgaremek címe: FitNet

Program nyújtotta szolgáltatások:

- Egyedi edzésterv generálás felhasználóknak
- Edzés naprakész követése
- Regisztráció
- Bejelentkezés
- Gyakorlatok ismertetése izomcsoportok szerint
- Fórum oldal, melyen a felhasználók kommunikálni tudnak
- Lejátszó felület a feladatok szemléltetéséhez
- Adminisztrációs felület

Sorszám	A konzultáció időpontja	A konzulens aláírása
1.	2023.10.17.	
2.	2023.11.21.	
3.	2023.12.13.	
4.	2024.01.16	
5.	2024.02.20.	
6.	2024.03.12	

A vizsgaremek beadható:

Vác, 2024.....

.....

Konzulens

A vizsgaremek átvettem:

Vác, 2024.....

.....

A szakképzést folytató
intézmény felelőse

Tartalomjegyzék

Hallgatói nyilatkozat.....	3
Konzultációs lap.....	4
Tartalomjegyzék	5
Témaválasztás	8
1 Fejlesztői dokumentáció	9
1.1 Alkalmazás ismertetése.....	9
1.2 Felhasznált programok, technológiák.....	9
1.2.1 PHP	9
1.2.2 JavaScript.....	9
1.2.3 MySQL és Apache	9
1.2.4 Ajax	10
1.2.5 REST API.....	10
1.3 Fejlesztői környezet	10
1.4 Adatszerkezet	11
1.4.1 Adatbázis alapadatai	11
1.4.2 Kapcsolatok	12
1.4.3 Táblák	13
1.5 Használati esetmodell	22
1.5.1 Felhasználói szerepek és funkciók.....	22
1.6 Modulok, menüpontok ismertetése.....	23
1.6.1 db.php	23
1.6.2 landing.php.....	24
1.6.3 user_register.php	26
1.6.4 user_login.php.....	29
1.6.5 introducing.php	32
1.6.6 submit.php	32
1.6.7 musclegroup.php.....	36
1.6.8 menu.php	38
1.6.9 dashboard.php	39
1.6.10 settings.php.....	42
1.6.11 inbox.php	47
1.6.12 forum.php	49
1.6.13 admin.php	51
1.6.14 statistic.php.....	53

1.6.15	user.php	54
1.6.16	filter_users.php	57
1.6.17	toggle_ban.php	58
1.6.18	banned.php	59
1.6.19	save_user.php	60
1.6.20	announcements.php	61
1.6.21	upload_announcements.php	62
1.6.22	delete_announcements_work.php	62
1.6.23	exercise_upload.php	63
1.6.24	get_categories_count.php	64
1.6.25	exercis_upload_work.php	65
1.6.26	filter_exercise.php	66
1.6.27	edit_exercise.php	66
1.6.28	save_user.php	68
1.6.29	delete_exercis.php	69
1.6.30	delete_exercise_work.php	70
1.6.31	topics_and_comments.php	70
1.6.32	delete_topic.php	70
1.6.33	delete_comment.php	71
1.6.34	logout.php	71
1.6.35	add_comment.php	72
1.6.36	avatar_upload.php	73
1.6.37	check_access.php	74
1.6.38	create_topic.php	74
1.6.39	delete_topic.php	75
1.6.40	delete_comment.php	76
1.6.41	fetch_week_data.php	76
1.6.42	functions.php	77
1.6.43	submit_comment.php	78
1.6.44	get_completed_dates.php	79
1.6.45	get_exercise_details.php	80
1.6.46	get_exercise.php	81
1.6.47	regenerate_workout.php	82
1.6.48	save_completed_workout.php	82
1.6.49	search.php	83
1.6.50	update_workout_consent.php	83

1.7	Tesztelés, tesztesetek.....	85
1.7.1	Regisztráció tesztelés	85
1.7.2	Bejelentkezés tesztelés	86
1.7.3	Felhasználói adatváltoztatás tesztelés	86
1.7.4	Feladat feltöltés tesztelés	87
1.7.5	Szemelyre szabott feladatsor generálás tesztelés	88
1.7.6	Feladatsor lejátszásához szükséges modul tesztelés	88
1.8	Továbbfejlesztési lehetőségek	89
1.8.1	A weboldal továbbfejlesztése	89
2	Felhasználói dokumentáció.....	90
2.1	Alkalmazás ismertetése.....	90
2.2	Rendszerkövetelmények	90
2.3	Szükséges beállítások	91
2.4	Alkalmazás használata.....	91
3	Irodalomjegyzék	109
4	Mellékletek.....	110

Témaválasztás

A FitNet webalkalmazás témaválasztásának indoklása az egészséges életmód népszerűsítése és támogatása. Az egészséges életmód kialakítása és fenntartása számos előnnyel jár, mint például a jobb fizikai és mentális egészség, a nagyobb energiaszint és a jobb életminőség. A FitNet webalkalmazás lehetőséget biztosít az emberek számára, hogy könnyen hozzáférjenek edzéstervhez, motivációs tartalmakhoz és közösségi támogatáshoz az egészséges életmód eléréséhez és fenntartásához. Az egészséges életmód népszerűsítése különösen fontos a modern életmódban, ahol a mozgásszegény életmód és az egészségtelen táplálkozási szokások elterjedtek. Egy fitness webalkalmazás segíthet az embereknek megtalálni azokat az edzésformákat és táplálkozási szokásokat, amelyek a legjobban megfelelnek az egyéni igényeiknek és céljainknak. Ezáltal egy ilyen alkalmazás nemcsak segít az embereknek elérni és megtartani a kívánt testi állapotot, hanem hozzájárul az egészségesebb és boldogabb életmód kialakításához és fenntartásához.

Készítők feladatai:

Fajta Jordán – Frontend

- Felhasználói felület tervezése és kialakítása
- Html/Css/Javascript fejlesztése
- Reszponzivitás biztosítása
- Interaktivitás
- Tesztelés és hibajavítás
- Design kialakítása
- Kommunikáció csapattal
- Témával kapcsolatos információ és kép gyűjtés

Mártai Balázs – Backend

- Alkalmazás logika megvalósítása
- Adatbázis kezelés
- Felhasználói autentikáció és engedélyezés
- Teljesítmény optimalizálása
- Tesztelés és hibajavítás
- Kommunikáció csapattal

1 Fejlesztői dokumentáció

1.1 Alkalmazás ismertetése

A fitness webalkalmazás egy online platform, amely lehetővé teszi bármely felhasználó számára nem, életkor és testi állapotától független, hogy egészséges életmódot folytassanak és támogassák egészségüket és fitnessüket. Az alkalmazás különböző funkciókat kínál, beleértve az edzésterv készítését, a közösségi támogatást és a motivációs tartalmakat. A fejlesztőknek felelősek az alkalmazás teljes körű fejlesztéséért és karbantartásáért. PHP-t használnak a backend logika megvalósításához, és MySQL adatbázist használnak az adatok tárolásához és kezeléséhez. A frontend fejlesztés során a CSS keretrendszerként Tailwind CSS-t használják, és saját CSS és SCSS kódokat írnak az alkalmazás egyedi stílusának kialakításához. JavaScript segítségével interaktivitást és dinamizmust adnak az alkalmazásnak, míg a Gulp.js automatizálja a fejlesztési folyamatot. Templateket használnak az alkalmazás különböző részeinek strukturálásához és megjelenítéséhez, hogy hatékonyan kezeljék a frontend fejlesztést és a változásokat. Az oldalon található képek más tulajdonát képezik, szemléltetés céljából lettek felhasználva.

1.2 Felhasznált programok, technológiák

1.2.1 PHP

A PHP egy általános szerveroldali szkriptnyelv dinamikus weblapok készítésére. Az első szkriptnyelvek egyike, amely külső fájl használata helyett HTML oldalba ágyazható. A kódot a webszerver PHP feldolgozómodulja értelmezi, ezzel dinamikus weboldalakat hozva létre.

1.2.2 JavaScript

A JavaScript programozási nyelv egy objektumorientált, prototípus-alapú szkriptnyelv, amelyet a weboldalakon elterjedten használnak. A JavaScript esetében a futási környezet jellemzően egy webböngésző, illetve annak JavaScript-motorja.

1.2.3 MySQL és Apache

Az Apache HTTP Server nyílt forráskódú webkiszolgáló alkalmazás, szabad szoftver, mely kulcsfontosságú szerepet játszott a World Wide Web elterjedésében. A projekt célja olyan webszerver program létrehozása, karbantartása, fejlesztése, mely megfelel a gyorsan változó Internet követelményeinek.

A MySQL egy többfelhasználós, többszálú, SQL-alapú relációs adatbázis-kezelő szerver, mely az adatokat táblákban tárolja el. Az adatok logikailag rendezve vannak oszlopokba, sorokba, illetve egyes táblák között kapcsolatokat hozhatunk létre. A

szabályokat a fejlesztő hozza létre és elérheti, hogy az adatbázisban ne lehessenek duplikációk, hiányzó adatok, nem megfelelő adatok.

1.2.4 Ajax

Az AJAX interaktív webalkalmazások létrehozására szolgáló webfejlesztési technika. Segítségével a weblap kis mennyiségű adatot cserél a szerverrel a háttérben, így a lapot nem kell újratölteni minden egyes alkalommal, amikor a felhasználó módosít valamit. Ez növeli a honlap interaktivitását, sebességét és használhatóságát.

1.2.5 REST API

A REST szoftverarchitektúra típus, elosztott kapcsolat nagy, internet alapú rendszerek számára, amilyen például a világháló. A REST típusú architektúra kliensekből és szerverekből áll. A kliensek kérést indítanak a szerverek felé; a szerverek kéréseket dolgoznak fel és a megfelelő választ küldik vissza.

1.3 Fejlesztői környezet

A FitNet webalkalmazásunkhoz a következő fejlesztői környezeteket használtuk:

- **Fejlesztői IDE: Visual Studio Code**
 - A Visual Studio Code-ot választottuk IDE-ként mivel egy könnyű, keresztplatformos kódszerkesztő eszköz, amely számos funkciót kínál a fejlesztőknek, beleértve a kódszerkesztést, hibakeresést, verziókezelést és kiterjesztési lehetőségeket.
- **Verzió kezelő rendszer**
 - A Gitet használtuk a kódváltozások nyomon követésére és a csapatmunka megkönnyítésére. A kódot a GitHubon tároltuk és kezeltük.
- **Backend fejlesztői eszközök**
 - PHP: A backend logika megvalósításához használt nyelv.
 - MySQL: Az adatbázis-kezelő rendszer, amelyet az alkalmazás adatkezeléséhez használtunk.
 - Gulp.js: Automatizálási eszköz a fejlesztési folyamat könnyítésére és felgyorsítására.
- **Frontend fejlesztői eszközök**
 - HTML, CSS, JavaScript: A frontend rész fejlesztéséhez használatos alapvető technológiák.
 - Tailwind CSS: A CSS keretrendszer, amelyet az alkalmazás stílusának kialakításához használtunk.
 - SCSS: A CSS preprocessor, amely lehetővé tette a CSS kód strukturáltabbá és könnyebben karbantarthatóvá tételét.

Ezen eszközök és konfigurációk segítségével hatékonyan fejlesztettük és karbantartottuk a fitness webalkalmazást a Visual Studio Code fejlesztői környezetben.

1.4 Adatszerkezet

1.4.1 Adatbázis alapadatai

MySQL kliens verziója: 5.7.26

Sql szerver: mysql

Adatbázis neve: mesterremek

Tárolómotor: InnoDB

Alapértelmezett illesztés: utf8mb4_general_ci

Sql-parancs:

```
CREATE DATABASE IF NOT EXISTS `mesterremek`  
DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4_general_ci;
```

1.4.2 Kapcsolatok



1.4.3 Táblák

1.4.3.1 Az „*announcements*” tábla

A rendszerüzeneteket és annak kiküldetésének idejét tárolja a tábla.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
announcementName	varchar(255)	Rendszerüzenet neve	
announcementsText	varchar(255)	Rendszerüzenet szövege	
date	datetime	Rendszerüzenet dátuma	

SQL-parancs:

```
CREATE TABLE `announcements` (  
  `id` int(11) NOT NULL,  
  `announcementName` varchar(255) NOT NULL,  
  `announcementsText` varchar(255) NOT NULL,  
  `date` datetime NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
ALTER TABLE `announcements`  
  ADD PRIMARY KEY (`id`);  
ALTER TABLE `announcements`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;  
--
```

1.4.3.2 A „comments” tábla

A felhasználók által elküldött kommenteket és annak adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
user_id	int(11)	Felhasználó azonosítója	FK
topic_id	int(11)	Topic azonosítója	FK
content	text	A komment tartalmát tárolja	
created_at	timestamp	Komment elkészítésének idejét tárolja	

SQL-parancs:

```
CREATE TABLE `comments` (  
  `id` int(11) NOT NULL,  
  `user_id` int(11) NOT NULL,  
  `topic_id` int(11) NOT NULL,  
  `content` text NOT NULL,  
  `created_at` timestamp NOT NULL DEFAULT current_timestamp()  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
ALTER TABLE `comments`  
  ADD PRIMARY KEY (`id`),  
  ADD KEY `topic_id` (`topic_id`),  
  ADD KEY `user_id` (`user_id`);  
ALTER TABLE `comments`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;  
ALTER TABLE `comments`  
  ADD CONSTRAINT `comments_ibfk_1` FOREIGN KEY (`topic_id`) REFERENCES `topics` (`id`),  
  ADD CONSTRAINT `comments_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`);  
--
```

1.4.3.3 Az „exercise_category” tábla

Segéd tábla ami azért felel, hogy az adott feladatokhoz tartozó id-hez kapcsolódjon, hogy melyik kategóriába tartozik.

Mező neve	Típusa	Leírás	FK/PK
workout_id	int(11)	Az edzés azonosítója	FK
category_id	int(11)	Kategória azonosítója	FK

SQL-parancs:

```
CREATE TABLE `exercise_category` (  
  `workout_id` int(11) NOT NULL,  
  `category_id` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
ALTER TABLE `exercise_category`  
  ADD KEY `workout_id` (`workout_id`),  
  ADD KEY `category_id` (`category_id`);  
ALTER TABLE `exercise_category`  
  ADD CONSTRAINT `exercise_category_ibfk_1` FOREIGN KEY (`workout_id`) REFERENCES `workout_exercises` (`id`) ON DELETE CASCADE,  
  ADD CONSTRAINT `exercise_category_ibfk_2` FOREIGN KEY (`category_id`) REFERENCES `workout_categories` (`id`) ON DELETE CASCADE;
```

1.4.3.4 A „notifications” tábla

A tábla a kiküldött rendszerüzenetek megtekintését vizsgálja a felhasználó részéről.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
announcements_id	int(11)	A rendszerüzenet azonosítója	FK
user_id	int(11)	Felhasználó azonosítója	FK
status	enum("read","unread")	A rendszerüzenet olvasottsági állapotát tárolja	

SQL-parancs:

```
CREATE TABLE `notifications` (  
  `id` int(11) NOT NULL,  
  `announcement_id` int(11) DEFAULT NULL,  
  `user_id` int(11) DEFAULT NULL,  
  `status` enum('read','unread') DEFAULT 'unread'  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
ALTER TABLE `notifications`  
  ADD PRIMARY KEY (`id`),  
  ADD KEY `announcement_id` (`announcement_id`),  
  ADD KEY `user_id` (`user_id`);  
ALTER TABLE `notifications`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;  
ALTER TABLE `notifications`  
  ADD CONSTRAINT `notifications_ibfk_1` FOREIGN KEY (`announcement_id`) REFERENCES `announcements` (`id`),  
  ADD CONSTRAINT `notifications_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`);
```

1.4.3.5 A „topics” tábla

A felhasználók által kiírt topicokat tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
user_id	int(11)	Felhasználó azonosítója	FK
title	varchar(255)	A topik címét tárolja	
content	text	A topic tartalmát tárolja	
created_at	timestamp	A topic létrehozásának az időpontját tárolja	

SQL-parancs:

```
CREATE TABLE `topics` (  
  `id` int(11) NOT NULL,  
  `user_id` int(11) NOT NULL,  
  `title` varchar(255) NOT NULL,  
  `content` text NOT NULL,  
  `created_at` timestamp NOT NULL DEFAULT current_timestamp()  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
ALTER TABLE `topics`  
  ADD PRIMARY KEY (`id`),  
  ADD KEY `user_id` (`user_id`);  
ALTER TABLE `topics`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;
```


1.4.3.6 A „user” tábla

A felhasználók alap adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
username	varchar(40)	A felhasználó által választott nevet tárolja	
email	varchar(40)	A felhasználó email címét tárolja	
password_hash	varchar(60)	A felhasználó letitkosított jelszavát tárolja	
reg_time	timestamp	A felhasználó regisztrálásának időpontját tárolja	
avatar	text	A felhasználó feltöltött profilképét tárolja	
is_banned	int(11)	A felhasználó oldalhoz való hozzáférését figyeli	
permission	int(11)	A felhasználó jogosultságát kezeli	
first_register	tinyint(11)	Figyeli hogy a felhasználó új felhasználó-e	
generated_workout	text	Legenerált edzést tárolja	
workout_consent	tinyint(1)	A felhasználó edzés igénybevételéhez való saját felelősség elfogadását figyeli	

SQL-parancs:

```
CREATE TABLE `user` (  
  `id` int(11) NOT NULL,  
  `username` varchar(40) NOT NULL,  
  `email` varchar(40) NOT NULL,  
  `password_hash` varchar(80) NOT NULL,  
  `reg_time` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),  
  `avatar` text NOT NULL,  
  `is_banned` int(11) DEFAULT NULL,  
  `permission` int(11) NOT NULL,  
  `first_register` tinyint(1) NOT NULL DEFAULT 1,  
  `generated_workout` text NOT NULL,  
  `workout_consent` tinyint(1) NOT NULL DEFAULT 0  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_hungarian_ci;  
ALTER TABLE `user`  
  ADD PRIMARY KEY (`id`);  
ALTER TABLE `user`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;
```

1.4.3.7 A „user_data” tábla

A felhasználó személyes adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
user_id	int(11)	Egyedi belső azonosító	PK
surname	varchar(100)	A felhasználó vezetéknévét tárolja	
firstname	varchar(100)	A felhasználó keresztnévét tárolja	
weight	int(11)	A felhasználó súlyát tárolja	
height	int(11)	A felhasználó magasságát tárolja	
adress	text	A felhasználó lakcímét tárolja	
gender	int(1)	A felhasználó identitását tárolja	
bithday	date	A felhasználó születési idejét tárolja	
phone	text	A felhasználó telefonszámát tárolja	
intention	int(2)	A felhasználó célját tárolja	
desired_time	int(2)	A felhasználó által kívánt hetek számát tárolja	

SQL-parancs:

```
CREATE TABLE `user_data` (  
  `user_id` int(11) NOT NULL,  
  `surname` varchar(100) NOT NULL,  
  `firstname` varchar(100) NOT NULL,  
  `weight` int(11) NOT NULL,  
  `height` int(11) NOT NULL,  
  `address` text NOT NULL,  
  `gender` int(1) NOT NULL,  
  `birthday` date NOT NULL,  
  `phone` text NOT NULL,  
  `intention` int(2) NOT NULL,  
  `desired_time` int(2) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
ALTER TABLE `user_data`  
  ADD PRIMARY KEY (`user_id`);  
ALTER TABLE `user_data`  
  MODIFY `user_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;  
ALTER TABLE `user_data`  
  ADD CONSTRAINT `user_data_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`) ON DELETE CASCADE;
```

1.4.3.8 A „workout_body_parts” tábla

Azt tárolja, hogy a kiválasztott edzés melyik testrészre legyen

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Elsődleges belső azonosító	
body_parts	varchar(50)	Az edzésre kiválasztott testrész	

SQL-parancs:

```
CREATE TABLE `workout_body_parts` (  
  `id` int(2) NOT NULL,  
  `body_parts` varchar(50) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_hungarian_ci;  
ALTER TABLE `workout_body_parts`  
  ADD PRIMARY KEY (`id`);  
ALTER TABLE `workout_body_parts`  
  MODIFY `id` int(2) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;
```

1.4.3.9 A „workout_categories” tábla

Az edzés kategóriákat tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
categories	varchar(50)	Az edzés kategóriákat tárolja	

SQL-parancs:

```
CREATE TABLE `workout_categories` (  
  `id` int(2) NOT NULL,  
  `categories` varchar(50) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_hungarian_ci;  
ALTER TABLE `workout_categories`  
  ADD PRIMARY KEY (`id`);  
ALTER TABLE `workout_categories`  
  MODIFY `id` int(2) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;
```

1.4.3.10 A „workout_completed” tábla

Az elvégzett edzéseket figyeli és tárolja azt, hogy mikor fejezték be a felhasználók az edzést.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
user_id	int(11)	A felhasználó azonosítója	FK
completion_date	date	Az elvégzett edzés dátumát tárolja	

SQL-parancs:

```
CREATE TABLE `workout_completed` (  
  `id` int(11) NOT NULL,  
  `user_id` int(11) NOT NULL,  
  `completion_date` date NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
ALTER TABLE `workout_completed`  
  ADD PRIMARY KEY (`id`),  
  ADD KEY `user_id` (`user_id`);  
ALTER TABLE `workout_completed`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;  
ALTER TABLE `workout_completed`  
  ADD CONSTRAINT `workout_completed_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`);
```

1.4.3.11 A „workout_exercises” tábla

Az edzés feladat adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
nameOfExercise	varchar(255)	A feladat nevét tárolja	
descriptionOfExercise	varchar(255)	A feladat leírását tárolja	
shortdescriptionOfExercise	varchar(255)	A feladat rövid leírását tárolja	
imageOfExercise	text	A feladathoz kiválasztott képet tárolja	
body_part_id	int(2)	A testrész azonosítója	FK

SQL-parancs:

```
CREATE TABLE `workout_exercises` (  
  `id` int(11) NOT NULL,  
  `nameOfExercise` varchar(255) NOT NULL,  
  `descriptionOfExercise` varchar(255) NOT NULL,  
  `shortdescriptionOfExercise` varchar(255) NOT NULL,  
  `imageOfExercise` text NOT NULL,  
  `body_part_id` int(2) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;  
ALTER TABLE `workout_exercises`  
  ADD PRIMARY KEY (`id`),  
  ADD KEY `body_part_id` (`body_part_id`);  
ALTER TABLE `workout_exercises`  
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;  
ALTER TABLE `workout_exercises`  
  ADD CONSTRAINT `workout_exercises_ibfk_1` FOREIGN KEY (`body_part_id`)  
    REFERENCES `workout_body_parts` (`id`) ON DELETE CASCADE;  
COMMIT;
```

1.5 Használati esetmodell

A FitNet alkalmazásunkban az alábbi szerepkörök vannak definiálva:

- **Látogató**
 - Az alkalmazásba még nem regisztrált felhasználó, aki csak böngészi az oldalt és megtekinti a nyilvános tartalmakat.
- **Felhasználó**
 - Regisztrált felhasználó, aki hozzáfér az alkalmazáshoz és kihasználja annak funkcióit, mint például az edzésterv készítése, a táplálkozási tanácsok elérhetősége, a közösségi funkciók használata stb.
- **Adminisztrátor**
 - Az adminisztrátor rendelkezik kiterjedt jogosultságokkal az alkalmazásban. Feladatai közé tartozik az új felhasználók regisztrációjának jóváhagyása, a tartalomkezelés, a hirdetések kezelése stb.

1.5.1 Felhasználói szerepek és funkciók

- **Látogató**
 - Regisztrációra való hivatkozás és regisztrációba lépés.
 - Böngészés az alkalmazás nyilvános tartalma között.
- **Felhasználó**
 - Bejelentkezés az alkalmazásba.
 - Edzésterv készítése és követése.
 - Köösségi funkciók használata, például kommentelés, posztolás stb.
- **Adminisztrátor**
 - Új felhasználók regisztrációjának jóváhagyása vagy elutasítása.
 - Meglévő felhasználók szerepkörének változtatása (pl: Admin jog kiosztása, bannolás)
 - Tartalomkezelés, például posztok vagy edzéstervszakaszok moderálása
 - Oldal statisztikáinak megtekintése
 - Rendszer üzenet kiküldése

1.6 Modulok, menüpontok ismertetése

1.6.1 db.php

Bevezetés

A **db.php** fájl felelős az adatbázis kapcsolódás kezeléséért az alkalmazásban. Ez a fájl tartalmazza a szükséges konfigurációs információkat a MySQL adatbázishoz való kapcsolódáshoz, valamint inicializálja és biztosítja a kapcsolatot az adatbázissal.

```
db.php
1  <?php
2  $servername = "localhost"; // A MySQL szerver neve
3  $username = "root"; // MySQL felhasználónév (alapértelmezett XAMPP beállítás)
4  $password = ""; // MySQL jelszó (alapértelmezett XAMPP beállítás)
5  $dbname = "mesterremek"; // Adatbázis neve
6
7  $conn = new mysqli($servername, $username, $password, $dbname);
8
9  if ($conn->connect_error) {
10     die("Hiba a kapcsolódás közben: " . $conn->connect_error);
11 }
12 ?>
```

Használat

A db.php fájlt az alkalmazás bármely pontján include-olni kell az adatbázissal való kommunikáció előtt. Az include PHP kóddal történik a következő módon:

```
<?php
include ("db.php");
```

Ezután az adatbáziskapcsolat az \$conn változón keresztül érhető el.

Konfiguráció

A db.php fájl tartalmazza az alapvető konfigurációs beállításokat az adatbáziskapcsolathoz. Ezek a következők:

- **\$servername:** A MySQL szerver elérési útja. Alap esetben ez "localhost", de módosítható más szerverekhez történő kapcsolódás esetén.

- **\$username:** A MySQL felhasználóneve. Alapértelmezetten "root" a XAMPP beállítások esetén.
- **\$password:** A MySQL jelszava. Alapértelmezetten üres string a XAMPP beállítások esetén.
- **\$dbname:** Az adatbázis neve, amellyel kapcsolatot kell létesíteni.

1.6.2 landing.php

Bevezetés

Az oldalunk főoldala a landing.php. Itt kínáljuk a lehetőséget regisztrációra vagy bejelentkezésre, és a felhasználók számára lehetőséget biztosítunk a gyakorlatok közötti böngészésre és az oldal érdekességeinek elolvasására. Ez az a hely, ahol minden látogató számára elérhetővé válnak azok a tartalmak és funkciók, amelyek segítségével könnyedén csatlakozhatnak közösségünkhöz és kihasználhatják az oldal nyújtotta előnyöket.

Menüsáv

Az oldal tetején található menüsáv lehetővé teszi a felhasználók számára, hogy navigáljanak az oldalon. Tartalmazza az "Home", "About us" és "Exercises" menüpontokat.

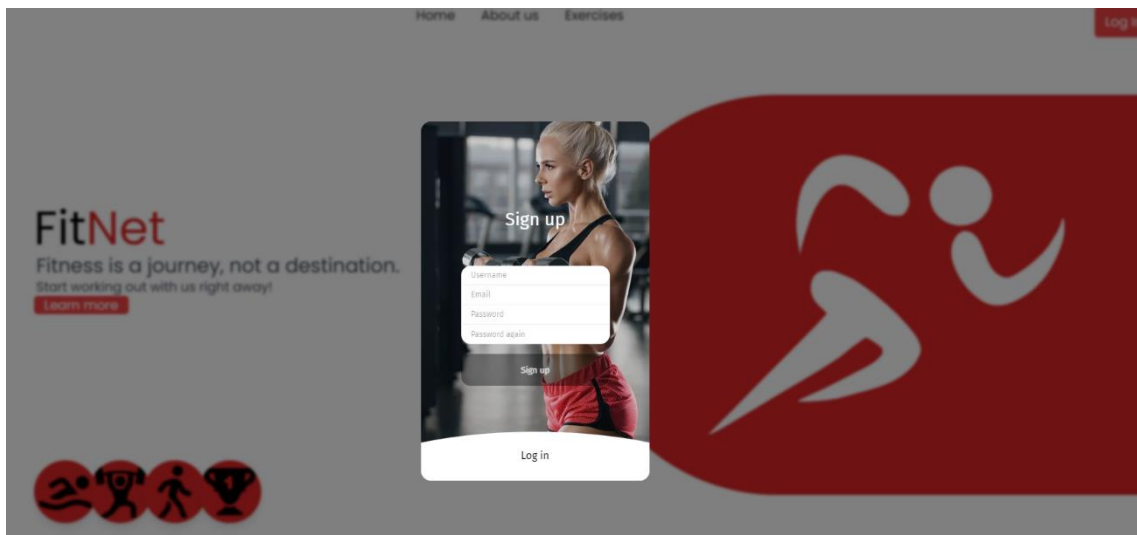
Home About us Exercises

- **Home:** Ez a főoldal, amely visszavezet az oldal tetejére.
- **About us:** Ez az "About us" szakaszhoz visz, ahol információkat találhatunk az oldalról.
- **Exercises:** Ez a menüpont a musclegroup.php oldalra irányít át, ahol tudunk böngészni különböző izomcsoportok gyakorlatai között.

Bejelentkezés és regisztráció

Az oldal lehetővé teszi a felhasználók számára, hogy bejelentkezzenek vagy regisztráljanak. Ha a felhasználó be van jelentkezve, az oldal átirányítja a felhasználót a vezérlőpultra.

A "Log In" gomb a bejelentkezési modal megnyitásáért felel, ahol a felhasználó bejelentkezhet vagy regisztrálhat. A "Log In" gombot a `login_page.js` scripttel aktiválják.



Session kezelés:

Az oldal elején megtalálható a session kezelése a PHP `session_start()` függvénnyel. Ez arra szolgál, hogy a felhasználói munkamenetet kezelje az oldalon. A `$_SESSION` tömböt használják az adatok tárolására és hozzáférésére a munkamenet során. Az if állítással ellenőrzik, hogy a felhasználó be van-e jelentkezve (`$_SESSION['loggedin'] === true`). Ha igen, akkor az oldalt átirányítják a `menu.php?page=dashboard` oldalra.

Bejelentkezés/regsiztrációs modal:

- Ha a felhasználó nincs bejelentkezve (`else` ág), akkor a "Log In" gombot jelenítik meg. Ennek a gombnak a lenyomására megjelenik egy modális ablak, ahol a felhasználó be tud jelentkezni vagy regisztrálni.
- A regisztráció és bejelentkezés űrlapjai POST metódussal küldik az adatokat a `user_register.php` és `user_login.php` PHP fájlokba.
- Az űrlapokhoz tartozó input mezők az adatok bekérésére szolgálnak (felhasználónév, e-mail, jelszó stb.).
- Az adatokat a szerver oldalon ellenőrzik, például hogy a jelszavak egyeznek-e (a "Password again" mezővel összevetve), vagy hogy az e-mail cím formailag megfelelő-e.

Felhasználók számának lekérdezése:

- Az adatbázis kapcsolatot az adatbázis szerverrel egy MySQL adatbázis felhasználói nevével, jelszójával és az adatbázis nevével hozzák létre (`$conn = new mysqli($host, $username, $password, $dbname);`).
- Az adatbázisban található felhasználók számát lekérjük egy SQL lekérdezéssel (`SELECT COUNT(id) AS total_users_count FROM user;`), majd megjelenítjük az oldal **About us** részén.

```

$totalUsersQuery = "SELECT COUNT(id) AS total_users_count FROM user";
$totalUsersResult = $conn->query($totalUsersQuery);
$totalUsersData = $totalUsersResult->fetch_assoc();
$totalUsersCount = $totalUsersData['total_users_count'];
?>
<div class="mb-12 md:mb-0">
  <h2 class="text-dark mb-4 text-3xl font-bold">
    <?php echo $totalUsersCount; ?>
  </h2>
  <h5 class="mb-0 text-lg font-medium text-neutral-500 dark:text-neutral-300">
    users
  </h5>
</div>

```

Lábléc:

- A láblécben négy ikon található, amelyek a közösségi médiára irányítanak.
- A lábléc alatt található menüpontok (**Home**, **About**, **Team**) is hiperhivatkozások, amelyek más oldalakra irányítanak.
 - **Home**: Az oldal tetejére irányít.
 - **About**: Az oldal **About us** részére navigál.
 - **Team**: A **team.php** oldalra irányít át, ahol a készítőkről tudhat meg többet a felhasználó.

1.6.3 user_register.php

A user_register.php a regisztrációs folyamatot valósítja meg a webalkalmazásunkban.

Adatbázis kapcsolódás

A **db.php** fájl tartalmazza az adatbázis kapcsolódási információkat és az ahhoz szükséges kódot.

```

// Adatbázis kapcsolódás
include ("db.php");

```

Űrlapból érkező adatok

Az űrlapról érkező felhasználónév (username), email cím (email), jelszó (password), és a jelszó megerősítése (passwordagain) POST kérések segítségével érkeznek.

```
// Űrlapból érkező adatok
$username = $_POST['username'];
$email = $_POST['email'];
$password = $_POST['password'];
$password_again = $_POST['passwordagain'];
```

Adatbázis ellenőrzése

Ellenőrzi, hogy van-e már ilyen felhasználónév vagy email az adatbázisban. Ehhez lekérdezi az adatbázist a megfelelő felhasználónévvel vagy email címmel.

```
// Ellenőrzés, hogy van-e már ilyen felhasználónév vagy email az adatbázisban
$check_sql = "SELECT * FROM user WHERE username = ? OR email = ?";
$check_stmt = $conn->prepare($check_sql);
$check_stmt->bind_param("ss", $username, $email);
$check_stmt->execute();
$check_result = $check_stmt->get_result();
```

Üres mezők ellenőrzése

Ellenőrzi, hogy a felhasználó által megadott mezők üresek-e. Ha bármelyik mező üres, akkor hibaüzenet jelenik meg.

```
//Üres mezők ellenőrzése
if ($password == "" || $username == "" || $email == "") {
    echo "<script>alert('Minden mezőt ki kell tölteni.');
```

Jelszó ellenőrzése és titkosítása (hashelése)

Ellenőrzi, hogy a két jelszómezőben megadott jelszavak megegyeznek-e. Ha minden ellenőrzés sikeres volt, akkor a jelszót hasheli a PHP `password_hash` függvény segítségével.

```
if ($password === $password_again) {
    // Jelszó hashelése
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);
```

Felhasználó beszúrása az adatbázisba

Ha nincs még ilyen felhasználó az adatbázisban, akkor beszúrja az új felhasználót az adatbázisba.

```
if ($check_result->num_rows > 0) {  
    // Már létezik ilyen felhasználónév vagy email cím  
    echo "<script>alert('A megadott felhasználónév vagy email cím már regisztrálva van.');    exit();  
} else {  
    // SQL lekérdezés a felhasználó beszúráshoz  
    $insert_sql = "INSERT INTO user (username, email, password_hash) VALUES (?, ?, ?)";  
    $insert_stmt = $conn->prepare($insert_sql);  
    $insert_stmt->bind_param("sss", $username, $email, $hashed_password);
```

Ha a felhasználó beszúrára kerül az adatbázisba, akkor PHPMailer segítségével kiküldünk a felhasználó E-mail címére egy üdvözlő üzenetet. A kiküldött E-mail elfogása Mailtrap segítségével történik.

Munkamenet indítása vagy folytatása

Ha a regisztráció sikeres volt, akkor a PHP session kezelését használja az új felhasználó bejelentkeztetésére.

```
if ($insert_stmt->execute()) {  
    session_start(); // Munkamenet indítása vagy folytatása  
    $_SESSION['loggedin'] = true;  
    $_SESSION['username'] = $username;  
    header("Location: ../user_pages/introducing.php");  
    exit();
```

Hibakezelés

Hibák esetén megjeleníti a megfelelő hibaüzenetet és visszairányítja a felhasználót a regisztrációs oldalra.

```
} else {  
    echo "<script>alert('Hiba történt a regisztráció során: " . $insert_stmt->error .  
        "');}  
  
$insert_stmt->close();
```

Adatbázis kapcsolat bezárása

A kapcsolatok és a lekérdezések után lezárja a kapcsolatot az adatbázissal.

```
$check_stmt->close();  
$conn->close();  
}
```

1.6.4 user_login.php

A user_login.php egy bejelentkeztető rendszert valósít meg. A felhasználók bejelentkezési adatait egy adatbázisban tárolja, és az üzeneteket szinkronizálja a PHP-szerverrel a HTTP-kérések által.

Adatbázis kapcsolódás

A **db.php** Az adatbázis kapcsolódást kezelő fájl, itt találhatóak az adatbázis kapcsolódási adatok és a kapcsolat inicializálása.

Session kezelése

A bejelentkeztetési állapotot a PHP session segítségével tárolja.

```
session_start();
```

Bejelentkezési ellenőrzése

Az űrlapról érkező adatokat ellenőrzi az adatbázisban tárolt adatokkal.

```
// Ellenőrizzük, hogy a bejelentkezési űrlap elküldése megtörtént-e  
if ($_SERVER["REQUEST_METHOD"] == "POST") {  
    // Az ellenőrzéshez használt adatbázis kapcsolódás  
    include ("db.php");  
  
    // Űrlapból érkező adatok  
    $email = $_POST['email'];  
    $password = $_POST['password'];  
  
    // Ellenőrzés az adatbázisban  
    $sql = "SELECT id, username, email, password_hash, is_banned, first_register FROM user WHERE email = ?";  
    $stmt = $conn->prepare($sql);  
    $stmt->bind_param("s", $email);  
  
    if ($stmt->execute()) {  
        $result = $stmt->get_result();  
        if ($result->num_rows == 1) {  
            $row = $result->fetch_assoc();  
        }  
    }  
}
```

Bejelentkezési státusz ellenőrzése, visszairányítás

Ellenőrzi, hogy a felhasználó már be van-e jelentkezve, és ha igen, hogy már kitöltötte-e az introducing oldalt. A megfelelő oldalra irányítja a felhasználót bejelentkezés vagy újra irányítás esetén.

```
// Ellenőrizzük, hogy a felhasználó már be van-e jelentkezve
if (isset($_SESSION['loggedin']) && $_SESSION['loggedin'] === true) {
    // Ha be van jelentkezve, ellenőrizzük a first_register értékét
    if ($_SESSION['first_register'] == 1) {
        // Ha a first_register értéke 1, azaz még nem töltötte ki az introducing.php-t
        header("Location: ../user_pages/introducing.php");
        exit;
    } else {
        // Ha a first_register értéke 0, azaz már kitöltötte az introducing.php-t
        header("Location: ../user_pages/menu.php");
        exit;
    }
}
```

Bejelentkezés és visszairányítás

Az űrlapról beérkező adatokat ellenőrzi az adatbázisban tárolt felhasználói adatokkal. A felhasználót az introducing oldalra vagy a menü oldalra irányítja, attól függően, hogy az új felhasználó, vagy már korábban bejelentkezett

```
if (password_verify($password, $row['password_hash'])) {
    session_start();
    $_SESSION['loggedin'] = true;
    $_SESSION['user_id'] = $row['id'];
    $_SESSION['username'] = $row['username'];
    $_SESSION['first_register'] = $row['first_register']; // first_register értékét elmentjük a session-be
    // Ellenőrizzük, hogy az új felhasználó
    if ($_SESSION['first_register'] == 1) {
        // Ha az új felhasználó, akkor az introducing.php-ra irányítjuk
        header("Location: ../user_pages/introducing.php");
        exit;
    } else {
        // Ha nem új felhasználó, akkor a menu.php-ra irányítjuk
        header("Location: ../user_pages/menu.php");
        exit;
    }
}
```

Session kezelés

Ha a bejelentkezés sikeres, létrehozza a session-t, és elmenti a szükséges adatokat, például a felhasználó azonosítóját és a bejelentkezési státuszt.

```
session_start();
$_SESSION['loggedin'] = true;
$_SESSION['user_id'] = $row['id'];
$_SESSION['username'] = $row['username'];
$_SESSION['first_register'] = $row['first_register'];
```

Hibakezelés

- **Hibás jelszó**

Ha a beírt jelszó nem egyezik meg az adatbázisban tárolttal, hibaüzenet jelenik meg, és a felhasználót visszairányítja a bejelentkező oldalra.

```
else {
    echo "<script>alert('Wrong password.');
```

- **Nincs ilyen felhasználó**

Ha az adatbázisban nem található megfelelő felhasználó, hibaüzenet jelenik meg, és a felhasználót visszairányítja a bejelentkező oldalra.

```
else {
    echo "<script>alert('There is no such register user.');
```

- **Bejelentkezési hibák**

Az adatbázissal történő kapcsolat vagy végrehajtás során felmerülő hibákat kiírja a böngészőbe.

```
else {
    echo "An error occurred during login: " . $stmt->error;
```

1.6.5 introducing.php

Az **introducing.php** oldal arra szolgál, hogy bemutassa az új felhasználókat az oldal funkcióival és megadja az oldal használatához szükséges adatokat. Ellenőrzi, hogy a felhasználó be van-e jelentkezve (`$_SESSION['loggedin'] === true`). Ha igen, lekéri az adatbázisból a felhasználó első regisztráció dátumát (**first_register** mező), és ellenőrzi, hogy ez az érték nem 0. Ha az érték nem 0, akkor megjeleníti az oldalt, amely az új felhasználók űrlapját tartalmazza az adataik megadásához. Ha a **first_register** értéke 0, akkor átirányítja a felhasználót a **dashboard.php** oldalra. Ha a felhasználó nincs bejelentkezve, átirányítja a felhasználót a **landing.php** oldalra.

Amikor a felhasználó a űrlapot elküldi, az oldal JavaScript kódja meghívja a **submitButton()** függvényt. Ez a függvény először meghívja a **submitForm()** függvényt, ami az űrlap adatait begyűjti, majd HTTP POST kérést indít a **submit.php** fájlra az adatokkal. Ha a kérés sikeres volt (HTTP státusz 200), akkor az oldal átirányítja a felhasználót a **menu.php** oldalra.

1.6.6 submit.php

A **submit.php** fájl az űrlapról érkező adatokat fogadja és feldolgozza. Ezeket az adatokat az adatbázisba menti, valamint frissíti a felhasználó adatait. Emellett generál egy edzéstervet a felhasználó számára és átirányítja a felhasználót a menü oldalra a sikeres feldolgozás után, vagy hiba esetén visszairányítja az előző oldalra.

Adatbázis kapcsolat

Az `include("db.php")` sorral a **db.php** fájl tartalmát betöltjük, amely tartalmazza az adatbáziskapcsolatot.

Munkamenet indítása vagy folytatása

`session_start()` függvény segítségével elkezdjük vagy folytatjuk a munkamenetet.

Felhasználónév és POST kérésekből érkező adatok lekérése

Az aktuális adatok lekérdezése a munkamenetből és tárolása változóban.


```

$username = $_SESSION['username'];

$surname = $_POST['surname'];
$firstname = $_POST['firstname'];
$weight = $_POST['weight'];
$height = $_POST['height'];
$address = $_POST['address'];
$phone = $_POST['phone'];
$birthday = $_POST['birthday'];
$gender = $_POST['gender']; // Módosított sor
$intention = $_POST['intention']; // Módosított sor
$workouttime = $_POST['workouttime'];

```

Gender és Intention értékek beállítása

A `$gender` és `$intention` értékek alapján egyedi értékek beállítása a `$gender_value` és `$intention_value` változókban.

```

$gender_value = "";
switch ($gender) {
    case "Man":
        $gender_value = 0;
        break;
    case "Women":
        $gender_value = 1;
        break;
    case "Etc":
        $gender_value = 2;
        break;
    default:
        $gender_value = 2; // Alapértelmezett érték
}

$intention_value = "";
switch ($intention) {
    case "bulking":
        $intention_value = 0;
        break;
    case "cutting":
        $intention_value = 1;
        break;
    case "weight maintenance":
        $intention_value = 2;
        break;
    default:
        $intention_value = 0; // Alapértelmezett érték
}

```

Felhasználó azonosítása a user táblából

Az adatbázisból lekérdezzük az aktuális felhasználó azonosítóját a `$user_id` változóba.

```
$user_query = "SELECT id FROM user WHERE username = ?";
$user_stmt = $conn->prepare($user_query);
$user_stmt->bind_param("s", $username);
$user_stmt->execute();
$user_result = $user_stmt->get_result();
$user_row = $user_result->fetch_assoc();
$user_id = $user_row['id'];

$user_stmt->close();
```

Felhasználó adatainak beszúrása a user_data táblába

Az űrlapról érkező adatok beszúrása az adatbázisba.

```
$insert_query = "INSERT INTO user_data (user_id, surname, firstname, weight,
height, address, gender, birthday, phone, intention, desired_time)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
$insert_stmt = $conn->prepare($insert_query);
$insert_stmt->bind_param("issiiissii", $user_id, $surname, $firstname, $weight,
$height, $address, $gender_value, $birthday, $phone, $intention_value, $workouttime);
```

Sikeres beszúrás esetén

Ha a beszúrás sikeres, akkor frissítjük a felhasználó bejegyzését a `first_register` értékének beállításával, majd generáljuk az edzéstervet a `generateWorkouts` függvény segítségével és végül átirányítjuk a felhasználót a menü oldalra. Ha hiba történik az adatok beszúrása közben, akkor egy JavaScript üzenetet jelenítünk meg, és visszairányítjuk a felhasználót az `introducing.php` oldalra.

```

if ($insert_stmt->execute()) {
    // Sikeres beszúrás esetén átirányítás
    $update_query = "UPDATE user SET first_register = 0 WHERE username = ?";
    $update_stmt = $conn->prepare($update_query);
    $update_stmt->bind_param("s", $username);
    $update_stmt->execute();
    $update_stmt->close();

    include("functions.php");
    // Edzésterv generálása
    generateWorkouts($username, $intention, $workouttime);

    header("Location: ../user_pages/menu.php");
    exit();
} else {
    // Hiba esetén visszatérés az előző oldalra
    echo "<script>alert('Hiba történt az adatok beszúrása során: " . $insert_stmt->error . "');
    window.location.href='../user_pages/introducing.php';</script>";
}

```

1.6.7 musclegroup.php

Az **musclegroup.php** lehetővé teszi a felhasználók számára, hogy különböző izomcsoportokra kattintva megtekinthessék az azokhoz tartozó gyakorlatokat.

[Home](#) [About us](#) [Exercises](#)

Muscle Group Selector

ARMS

Biceps

Deltoids

Forearms

Triceps

BACK

Trapezius

Lats

CORE

Abs

Obliques

Pectorals

LEGS

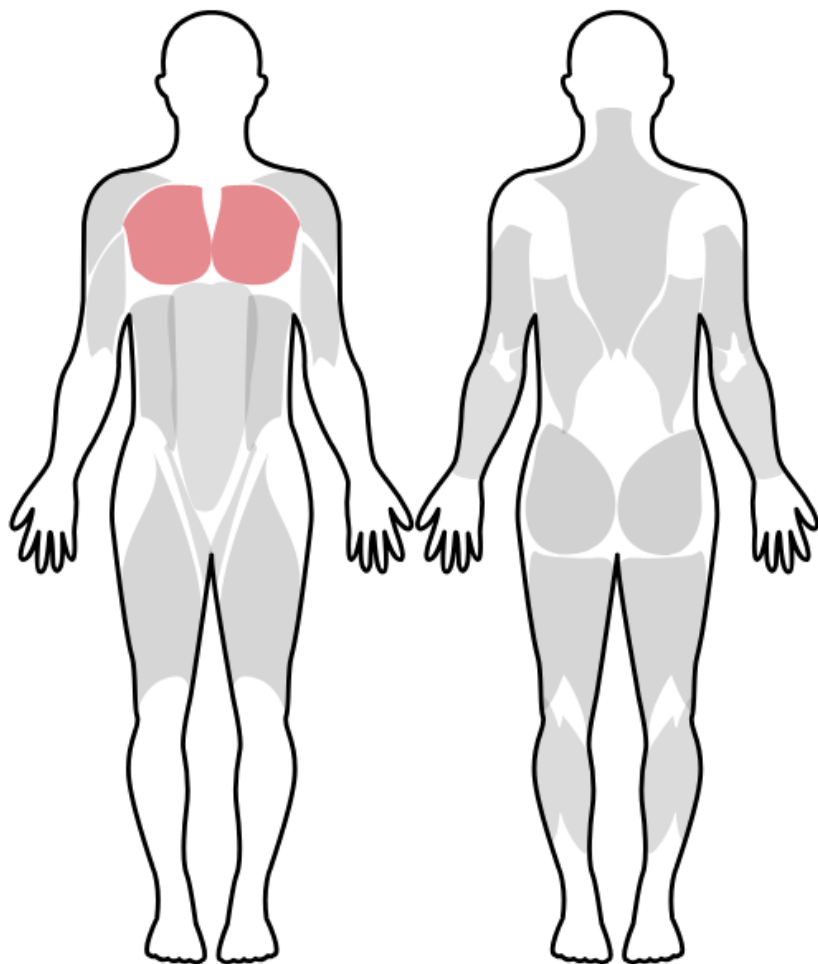
Adductors

Calves

Hamstrings

Glutes

Quads



Muscle group selector

A felhasználóknak lehetőségük van használni a bal oldalt megjelenő listaszerű navigációs menüt ahol az izomcsoportok nevei találhatóak, vagy oldal középső részén található diagramot, amelyen az izomcsoportok ikonjai vannak ábrázolva. A nevekre és ikonokra kattintva a felhasználók átléphetnek az adott izomcsoport gyakorlatainak oldalára.

Izomcsoportok oldalai

Ezeket az oldalakat érhetjük el a muscle group selector hivatkozásain keresztül

1. **Biceps:** Az "Biceps" nevére vagy ikonjára kattintva átléphetünk a "biceps.php" oldalra.
2. **Deltoids:** Az "Deltoids" nevére vagy ikonjára kattintva átléphetünk a "shoulder.php" oldalra.
3. **Trapezius:** Az "Trapezius" nevére vagy ikonjára kattintva átléphetünk a "trapezius.php" oldalra.
4. **Lats:** Az "Lats" nevére vagy ikonjára kattintva átléphetünk a "lats.php" oldalra.
5. **Triceps:** Az "Triceps" nevére vagy ikonjára kattintva átléphetünk a "triceps.php" oldalra.
6. **Forearms:** Az "Forearms" nevére vagy ikonjára kattintva átléphetünk a "forearms.php" oldalra.
7. **Glutes:** Az "Glutes" nevére vagy ikonjára kattintva átléphetünk a "glutes.php" oldalra.
8. **Hamstrings:** Az "Hamstrings" nevére vagy ikonjára kattintva átléphetünk a "hamstrings.php" oldalra.
9. **Calves:** Az "Calves" nevére vagy ikonjára kattintva átléphetünk a "calves.php" oldalra.
10. **Adductors:** Az "Adductors" nevére vagy ikonjára kattintva átléphetünk a "adductors.php" oldalra.
11. **Quads:** Az "Quads" nevére vagy ikonjára kattintva átléphetünk a "quads.php" oldalra.
12. **Abs:** Az "Abs" nevére vagy ikonjára kattintva átléphetünk a "abs.php" oldalra.
13. **Obliques:** Az "Obliques" nevére vagy ikonjára kattintva átléphetünk a "obliques.php" oldalra.
14. **Pectorals:** Az "Pectorals" nevére vagy ikonjára kattintva átléphetünk a "pectorals.php" oldalra.

1.6.8 menu.php

A **menu.php** oldal a FitNet alkalmazás főmenü felülete, amely lehetőséget biztosít a regisztrál vagy bejelentkezett felhasználók számára különböző szakaszok elérésére a felhasználói jogosultságok és beállítások alapján.

Include-olt fájlok

- **db.php**: Fájl az adatbáziskapcsolat létrehozásához.
- **check_access.php**: Fájl a felhasználói bejelentkezési állapot és hozzáférési engedélyek ellenőrzésére.
- **dashboard.php**, **exercises.php**, **inbox.php**, **users.php**, **upgrade.php**, **settings.php**, **contact.php**, **forum.php**: Fájlok a különböző alkalmazásrészletek vagy oldalak megjelenítéséhez, dinamikusan beillesztve a felhasználó választása alapján.
- **logout.php**: Fájl a felhasználó kijelentkezési funkcionalitásának kezelésére.
- **admin.php**: Fájl az adminisztrációs panelhez, amely hozzáférhető az adminisztrátori jogosultsággal rendelkező felhasználók számára.

Fő Komponensek:

1. **Felhasználói Hitelesítés:**
 - Ellenőrzi, hogy a felhasználó be van-e jelentkezve (**\$_SESSION['loggedin']**) és átirányít a bejelentkezési oldalra, ha nem átirányít a bejelentkezési oldalra.
 - Felhasználói információkat kér le, beleértve a felhasználónevet, a profilképet és az olvasatlan értesítések számát.
2. **Oldalsáv Navigáció:**
 - Összecsukható oldalsáv navigációs menü a különböző alkalmazásrészletek eléréséhez.
 - A szakaszok közé tartozik a Dashboard, Settings, Inbox, Forum és Admin panel (ha a felhasználónak van adminisztrátori jogosultsága).
3. **Dinamikus Oldalbetöltés:**
 - Dinamikusan betölti a tartalmat a kiválasztott oldal alapján (**\$_GET['page']**).
 - Különböző PHP fájlokat tartalmaz a különböző alkalmazásrészletek vagy oldalak számára.
4. **Felhasználói Felület:**
 - Megjeleníti a felhasználó profilképét, felhasználónevét, és ha be van jelentkezve, egy kijelentkezési gombot.
 - A Tailwind CSS osztályok használatával érzékeny tervezést valósít meg.

Javascript függvények

1. `toggleSidebar()` és `hamburger()` függvények:

- Ezek a JavaScript függvények felelősek a sidebar (oldalsáv) állapotának váltásáért és a hamburger menü megjelenítéséért.
- A `toggleSidebar()` függvény módosítja az oldalsáv CSS transzformációját, hogy megjelenítse vagy elrejtse azt.
- A `hamburger()` függvény az oldalsáv megnyitását vagy bezárását kezeli kattintásra.

2. `toggleSidebar()` függvény működése:

- A függvény először meghatározza az oldalsáv elemet az `id` alapján (`default-sidebar`).
- Ezután a `classList.toggle()` metódust használja az `-translate-x-full` osztály hozzáadására vagy eltávolítására az oldalsávhoz.
- Az `-translate-x-full` osztály a Tailwind CSS-ben az oldalsáv elrejtésére szolgál a teljes szélesség mentén.

```
function toggleSidebar() {  
  const sidebar = document.getElementById("default-sidebar");  
  sidebar.classList.toggle("-translate-x-full");  
}
```

3. `hamburger()` függvény működése:

- Ez a függvény egyszerűen meghívja a `toggleSidebar()` függvényt, hogy váltson az oldalsáv megjelenítése és elrejtése között.

```
function hamburger() {  
  toggleSidebar();  
}
```

1.6.9 dashboard.php

A `dashboard.php` oldal a FitNet platform felhasználóinak testmozgás Irányítópultja. Strukturált nézetet biztosít a felhasználók gyakorlati rutinjáról, hetekre és napokra bontva. A felhasználók navigálhatnak az edzésütemterven, megtekinthetik az egyes gyakorlatok részleteit, és nyomon követhetik haladásukat.

Funkciók

1. Edzésütemterv megjelenítése:

- A felhasználók láthatják az edzésütemtervet hetekre és napokra bontva.
- Minden napot egy kattintható gomb reprezentál, amely megjeleníti a dátumot.
- Pihenőnapok vizuálisan elkülönítettek, és kiválaszthatatlanok.

2. **Modális ablak a gyakorlatok részleteihez:**

- Az egyes gyakorlatok részleteit megjelenítő modális ablak megnyitása a nem pihenőnapokon történő gombkattintással.
- A felhasználók egyenként gyakorlatok között navigálhatnak a "Previous" és "Next" gombok segítségével.
- A modális ablak olyan információkat nyújt, mint a gyakorlat neve, leírása és egy kép.

3. **Navigáció és interakció:**

- A felhasználók a billentyűparancsok segítségével (Nyílombok) navigálhatnak a gyakorlatok között.
- A modális ablak bezárható az "X" gombra kattintva vagy az Escape billentyű megnyomásával.
- A haladás automatikusan mentésre kerül az edzés rutin befejezésekor.

4. **Edzési hozzájárulás:**

- A felhasználóknak hozzájárulást kell adniuk az edzésütemterv hozzáféréséhez.
- Ha a hozzájárulás nem kerül megadásra, egy üzenet jelenik meg, amely magyarázza a megfelelő edzés végrehajtásának fontosságát, valamint egy hozzájáruló gomb.

Javascript függvények

1. `fetchWeeksAndDays()`

- Az `fetch_week_data.php` fájlról JSON adatokat kér le, amelyek az edzésütemterv hetekre és napokra bontott részleteit tartalmazzák.
- Az adatok alapján gombokat hoz létre a hetekhez és napokhoz, amelyeket a felhasználók kattintásokkal navigálhatnak.
- A pihenőnapokat vizuálisan elkülöníti és letiltja a kiválasztást.

2. `fetchExercises(data)`

- Az `get_exercises.php` fájlról lekéri az adott dátumhoz tartozó gyakorlatok részleteit.
- A gyakorlatok adatait JSON formátumban kapja meg, majd megjeleníti a modális ablakban.

3. `displayDays()`

- A `displayDays()` függvény felelős a napok megjelenítéséért a felhasználó számára az adott héten.
- A függvény az előző napokat, majd a kapott adatok alapján létrehoz gombokat a napokhoz
- Ellenőrzi, hogy a nap befejezett-e, és ha igen, zöldre színezi a gombot
- A pihenőnapokat szürkére színezi és letiltja a kiválasztást
- Egyes napokra kattintva megjeleníti a hozzájuk tartozó gyakorlatokat vagy pihenőnapokat

4. `showExercise(index)`

- A `showExercise(index)` függvény felelős az egyes gyakorlatok részleteinek megjelenítésért a modális ablakban
- A függvény megkapja a gyakorlat indexét, majd az adott index alapján megjeleníti a gyakorlat nevét, leírását, képeit
- Ellenőrzi, hogy van-e gyakorlat az adott indexen, és ha nem, megjeleníti a „No exercises available” üzenetet
- Engedélyezi vagy letiltja a „Previous” és „Next” gombokat a gyakorlatok közötti navigáláshoz
- A `nextExercise()` a gyakorlatok befejezésekor automatikusan menti a haladást az adatbázisba

5. **prevExercise()** és **nextExercise()**
 - Az előző és a következő gyakorlatok megjelenítését végzi a modalitásban.
6. **closeModal()**
 - Bezárja a modális ablakot, és frissíti a gomb szövegét és háttérszínét, ha szükséges.
7. **saveCompletedWorkout()**
 - Az adatok mentése az adatbázisba az edzés befejezésekor.
 - AJAX kérést küld a szervernek a befejezett edzés dátumával.
8. **acceptConsent()**
 - A felhasználói hozzájárulást kezeli az edzésütemterv eléréséhez.
 - Az **update_workout_consent.php** fájlt hívja meg, hogy frissítse a felhasználó hozzájárulását.

Fájlok

- **fetch_week_data.php**: Az edzésütemterv JSON adatának lekérése és feldolgozása.
- **get_completed_dates.php**: A befejezett dátumok lekérése az adatbázisból a vizuális jelzéshez.
- **get_exercises.php**: Gyakorlat részleteinek lekérése a kiválasztott dátum alapján.
- **get_exercise_details.php**: Részletes információk lekérése egyes gyakorlatokról.
- **save_completed_workout.php**: A befejezett edzés adatának mentése az adatbázisba.
- **update_workout_consent.php**: A felhasználói hozzájárulás kezelése az edzésütemterv eléréséhez.

1.6.10 settings.php

A **setting.php** a felhasználók beállításainak kezelésére szolgáló PHP fájl.

Csatlakozás adatbázishoz bejelentkezés ellenőrzése

Az adatbázis csatlakozás a **db.php**-val történik. Ellenőrizzük, hogy a felhasználó be van-e jelentkezve, ha nem akkor átirányítjuk a bejelentkezési oldalra.

```
<?php
// Először csatlakozunk az adatbázishoz
include ("db.php");

// Ellenőrizzük, hogy be van-e jelentkezve a felhasználó
if (!isset($_SESSION['username'])) {
    // Ha nincs bejelentkezve, átirányítjuk a bejelentkezési oldalra vagy hibaüzenetet jelenítünk meg
    header("Location: login.php");
    exit();
}
```

Felhasználói adatok lekérése

A bejelentkezett felhasználóhoz tartozó adatok lekérdezése az adatbázisból. Az adatokat változókba mentjük a későbbi felhasználás céljából.

Adatok frissítése

Ha a felhasználó az "Update" gombra kattintott, az újonnan megadott adatokat frissítjük az adatbázisban. Az adatok frissítésekor az űrlapból érkező POST adatokat használjuk.

Űrlap megjelenítése, felhasználói interakció

Az űrlap segítségével a felhasználó megváltoztathatja adatait. A meglévő adatokat az űrlap elemeinek értékei között jelenítjük meg. A felhasználó az űrlapon keresztül módosíthatja adatait, feltöltheti avatárját vagy edzéstervét újra generálhatja.

Surname:

Citrom

First name:

Cecil

Weight:

71

kg

Height:

191

cm

Choose your gender:

☒ Man

☐ Woman

☐ Etc.

Change your phone number:

+

06301857345

Change your birthday:

2000. 01. 21.

Set your intention:

bulking

Update

Upload your avatar: (Only JPG, JPEG, PNG)

Fájl kiválasztása

Nincs fájl kiválasztva

Upload

Regenerate workout:

For how much week would you like to generate?

☐ Regenerate workout

Regenerate

Javascript függvények

A `settings.js` felelős az űrlap interaktivitásának biztosításáért.

Funkciói

Az űrlapban lévő súly és magasság input mezők értékeit beállítja az adatbázisból származó értékek alapján. Figyeli a súly és magasság input mezők értékének változását, és frissíti a mellettük lévő szám mezők értékét az aktuális értékükkel. Az input range értékét arányosan beállítja a mellette lévő számmező értékével az input mezők skálája alapján.

```
var heightInput = document.getElementById('height');
var weightInput = document.getElementById('weight');

// Az adatbázisból származó értékek beállítása
document.getElementById('height-data').value = heightInput.value;
document.getElementById('weight-data').value = weightInput.value;

// A csúszka értékének beállítása a mellette lévő számmező értékével
heightInput.addEventListener('input', function() {
    updateTextInput(heightInput.value, 'height-data');
});

weightInput.addEventListener('input', function() {
    updateTextInput(weightInput.value, 'weight-data');
});

function updateTextInput(val, elID) {
    document.getElementById(elID).value = val;

    // Az input range értékét arányosan beállítjuk a beírt érték alapján
    var rangeInput = (elID === 'height-data') ? document.getElementById('height')
    : document.getElementById('weight');
    var min = parseInt(rangeInput.min);
    var max = parseInt(rangeInput.max);

    // Az arányos érték kiszámítása és beállítása
    var scaledValue = (val - min) / (max - min) * 300; // Az 100 a range skálájának maximuma
    rangeInput.value = scaledValue;
}
```

1.6.11 inbox.php

A `inbox.php` lehetővé teszi a felhasználók számára, hogy megtekintsék az értesítéseiket a FitNet alkalmazásból. Az oldalon megjelennek az értesítések, amelyeket az adatbázisból lekérdeznek és megjelenítenek.

Adatbáziskapcsolat

Az adatbázishoz való csatlakozás a `db.php`-val történik.

Értesítések kezelése

Ellenőrzi, hogy a felhasználó átkattintott-e az üzenetre. Ha igen, frissíti az üzenet státuszát "read"-re az adatbázisban.

```
// Ellenőrizzük, hogy a felhasználó átkattintott-e az üzenetre
if (isset($_GET['id'])) {
    // Ha átkattintott, akkor lekérjük az üzenet azonosítóját
    $announcementId = $_GET['id'];

    // Frissítjük az üzenet státuszát "read"-re az adatbázisban
    $update_status_query = "UPDATE notifications
SET status = 'read' WHERE announcement_id = $announcementId";
    mysqli_query($conn, $update_status_query);
}
```

HTML generálása

Megjeleníti az értesítéseket, ha vannak, vagy kiírja, ha nincsenek értesítések.

```
if ($result->num_rows > 0) {
    echo "<div>";
    echo "<h1 class='text-xl'>Announcements:</h1>";
    while ($row = $result->fetch_assoc()) {
        echo "<div style='background-color: #f9f4f4; padding: 15px; border-radius: 10px; margin-top: 10px; text-align: center;'>";
        echo "<h2 style='text-align: center;'>Announcement: " . $row['announcementName'] . "<h2>";
        echo "<h2 style='text-align: center;'>Date: " . $row['date'] . "</h2>";
        echo "<div style='text-align: center; color: red; ' class='text-xl'>" . $row['announcementsText'] . "</div></div>";
    }
    echo "</div>"; // Fő div lezárása
} else {
    echo "There are no announcements currently.";
}
```

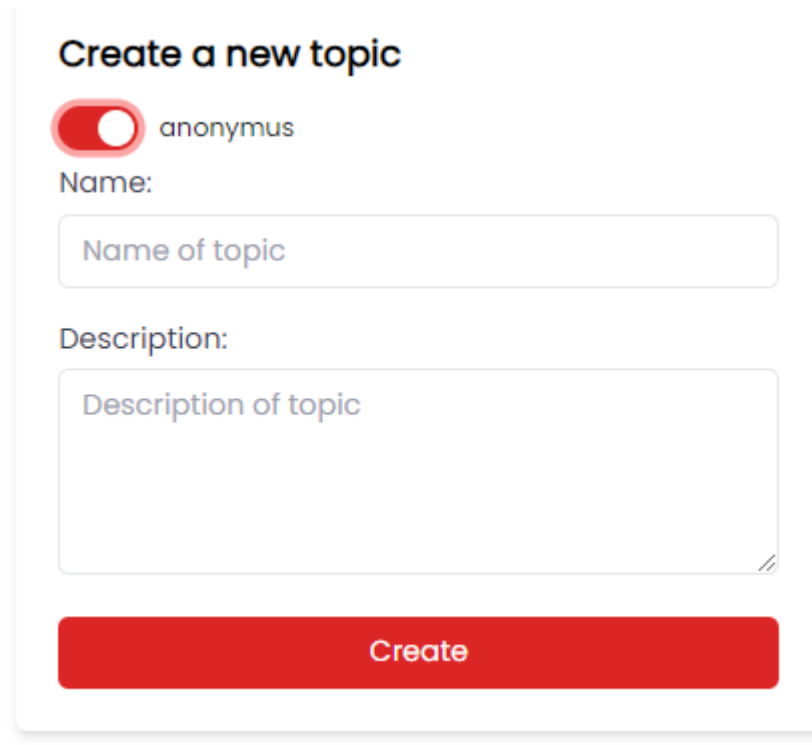
Üzenetek lekérdezése adatbázisból

Lekérdezi az összes értesítést az adatbázisból, beleértve az azonosítót, az értesítés címét, szövegét és dátumát.

```
// Fogadott üzenetek lekérdezése az adatbázisból
$sql = "SELECT id, announcementName, announcementsText,
date FROM announcements";
$result = $conn->query($sql);
```


1.6.12 forum.php

A fórum. lehetővé teszi felhasználók számára, hogy különböző témákról beszéljenek, kérdéseket tegyenek fel és megválaszoljanak mások kérdéseit. A felhasználók új témákat hozhatnak létre, kommentelhetnek a létező témákhoz és törölhetik saját kommentjeiket vagy témáikat.



The image shows a web form titled "Create a new topic". It features a red toggle switch labeled "anonymus" which is currently turned on. Below this, there is a "Name:" label followed by a text input field with the placeholder "Name of topic". Underneath that is a "Description:" label followed by a larger text area with the placeholder "Description of topic". At the bottom of the form is a prominent red button with the word "Create" in white text.

Adatbázis kapcsolat

A **db.php**-val történik az adatbázis kapcsolódás.

Fórum témák listázása és megjelenítése

A \$query változóban lekérjük az összes témát a topics táblából, majd egy while ciklussal kilistázzuk azokat az oldalon. A felhasználó neve is megjelenik, ha be van jelentkezve, különben "anonymous" lesz a neve.

```
// Ellenőrizzük, hogy vannak-e topicok
if (mysqli_num_rows($result) > 0) {
    // Topicok kilistázása
    while ($row = mysqli_fetch_assoc($result)) {
        $user_id = $row['user_id'];
        $topicId = $row['id'];

        if ($user_id != 0) {
            $user_query = "SELECT username FROM user WHERE id = $user_id";
            $user_result = mysqli_query($conn, $user_query);
            $user_row = mysqli_fetch_assoc($user_result);
            $username = $user_row['username'];
        } else {
            $username = 'anonymous';
        }
    }
}
```

Hozzászólások listázása és megjelenítése

A témához kapcsolódó hozzászólásokat is kilistázzuk. Ha a felhasználó saját hozzászólását nézi, lehetősége van a törlésre is.

```
$comment_query = "SELECT comments.*, user.username
FROM comments
LEFT JOIN user ON comments.user_id = user.id
WHERE topic_id = $topicId
ORDER BY comments.created_at DESC";

$comment_result = mysqli_query($conn, $comment_query);
```

Javascript függvények

1. **toggleForm()**: Ez a függvény váltja a "topicForm" div megjelenését és eltűnését a "New topic" gombra való kattintáskor. Ha a forma látható, akkor elrejtje azt, és fordítva.
2. **change_text()**: Ez a függvény változtatja meg a "topicForm" formban az "anonymus" checkboxhoz tartozó szöveget, hogy megjelenítse vagy elrejtse a felhasználónevet. Ha az eredeti szöveg még nem lett elmentve (originalText), elmenti azt, majd megváltoztatja a szöveget a "newText"-re vagy vissza az eredetire.
3. **toggleComments(topicId)**: Ez a függvény váltja a témához kapcsolódó kommentek megjelenítését vagy elrejtését. Ha a kommentek láthatók, elrejtje azokat, ha nem, akkor megjeleníti. A függvény továbbá megjeleníti vagy elrejtje a teljes témaleírást a kattintásnak megfelelően.

4. **submitComment(topicId, username)**: Ez a függvény küldi el az új hozzászólást a szervernek AJAX kéréssel. Az űrlapból kinyert szöveget és a témához tartozó azonosítót küldi el, majd a szerver választát frissíti az oldalon.
5. **deleteComment(commentId)** és **deleteTopic(topicId)**: Ezek a függvények felelősek a kommentek és témák törléséért. Mindkettő megerősítő ablakot jelenít meg a felhasználónak a törlés előtt, majd AJAX kérést küld a szervernek a törlés végrehajtásához.

1.6.13 admin.php

Az **admin.php** adminisztrációs felületet biztosít a FitNet weboldalon. Ez az oldal kezeli az adminisztrátori funkciókat és lehetőségeket.

Felhasználói Bejelentkezés Ellenőrzése

Az **admin.php** megvizsgálja, hogy a felhasználó be van-e jelentkezve (`$_SESSION['loggedin'] === true`). Ha igen, akkor megjeleníti a felhasználónevet és a "Log Out" gombot, különben a "Log In" gomb jelenik meg.

Avatar kezelése

Az avatar elérési útját lekéri az adatbázisból és megjeleníti a felhasználó képének megfelelően. Ha a felhasználónak nincs avatarja (`$row['avatar'] == ''`), akkor az alapértelmezett kép (`fitnet.png`) jelenik meg. Ha van avatarja, akkor a feltöltött kép jelenik meg az elérési út alapján.

```
$username = $_SESSION['username'];
$sql = "SELECT avatar FROM user WHERE username = '$username'";
$result = $conn->query($sql);
$row = $result->fetch_assoc();

if ($row['avatar'] == "") {
    $default = "fitnet.png";
    $avatarPath = '../uploads/defaults/' . $default;
} else if ($result->num_rows > 0) {
    $avatarPath = '../uploads/' . $row['avatar']; // Kép elérési útjának beállítása
}
```

Oldalnavigáció és oldalbetöltés

Az oldalnavigáció a GET változó (`$_GET['page']`) alapján történik, ami a **admin.php** URL-jében van megadva. A kiválasztott oldal betöltése PHP include segítségével történik. A megfelelő PHP fájlt betölti a switch utasítás alapján.

```

$page = isset($_GET['page']) ? $_GET['page'] : 'statistics';

// Az oldal tartalmának betöltése
switch ($page) {
    case 'statistics':
        include ('statistics.php');
        break;
    case 'exercise_upload':
        include ('exercise_upload.php');
        break;
    case 'users':
        include ('users.php');
        break;
    case 'announcements':
        include ('announcements.php');
        break;
    case 'exercise_delete':
        include ('delete_exercise.php');
        break;
    case 'exercise_edit':
        include ('edit_exercise.php');
        break;
    case 'topics_and_comments':
        include ('topics_and_comments.php');
        break;
    case 'moderation':
        include ('moderation.php');
        break;
    default:
        include ('statistics.php');
}

```

Navigációs menü ismertetése

- **statistics:** Statisztikai adatokat jelenít meg az oldalról.
- **upload exercises:** Gyakorlatok feltöltésére szolgáló felület.
- **users:** Felhasználók kezelése (lista, módosítás, törlés stb.).
- **announcements:** Rendszer üzenetek kezelése (hozzáadás, szerkesztés, törlés stb.).
- **delete exercise:** Gyakorlatok törlése.
- **edit exercise:** Gyakorlatok szerkesztése.
- **topics and comments:** Fórumtémák és hozzászólások kezelése.
- **moderation:** Moderálási funkciók (tovább fejlesztés alatt).
- **return to user page:** Visszanavigálja az admint a felhasználói felületre.

Javascript függvények

Oldalsáv Megjelenítése és Elrejtése

A **toggleSidebar()** függvény segítségével az oldalsávot meg lehet jeleníteni vagy elrejtteni. A **hamburger()** függvény az oldalsávot megjeleníti vagy elrejt, amikor a hamburger ikonra kattintanak.

```
function toggleSidebar() {
    const sidebar = document.getElementById("default-sidebar");
    sidebar.classList.toggle("-translate-x-full");
}

function hamburger() {
    toggleSidebar();
}
```

1.6.14 statistic.php

A `statistics.php` a FitNet webalkalmazás statisztikai adatainak megjelenítését végzi, beleértve az új felhasználók számát az elmúlt 24 órában, az összes felhasználó számát és a szerver uptime-ot.

Funkciók

1. **Új Felhasználók Megjelenítése:** Lekérdezi az új felhasználók számát az elmúlt 24 órában a `user` táblából, majd megjeleníti ezt az adatot a felhasználóhoz tartozó kártyában.

```
$newUsersQuery = "SELECT COUNT(id) AS new_users_count
FROM user WHERE reg_time >= NOW() - INTERVAL 1 DAY";
$newUsersResult = $conn->query($newUsersQuery);
$newUsersData = $newUsersResult->fetch_assoc();
$newUsersCount = $newUsersData['new_users_count'];
```

2. **Összes Felhasználó Megjelenítése:** Lekérdezi az összes felhasználó számát a `user` táblából, majd megjeleníti ezt az adatot a felhasználóhoz tartozó kártyában.

```
$totalUsersQuery = "SELECT COUNT(id) AS total_users_count FROM user";
$totalUsersResult = $conn->query($totalUsersQuery);
$totalUsersData = $totalUsersResult->fetch_assoc();
$totalUsersCount = $totalUsersData['total_users_count'];
```

3. **Szerver Uptime Megjelenítése:** Számolja az eltelt időt a szerver utolsó indításától, majd megjeleníti ezt az adatot az uptime-hoz tartozó kártyában.

```

if (!file_exists($serverStartTimeFile)) {
    // Ha nem létezik, vagy a tartalma üres, akkor létrehozuk és beleírjuk a jelenlegi időt
    file_put_contents($serverStartTimeFile, time());
}

// Szerver indításakor és leállításakor nullázzuk az eltelt időt
if (isset($_SERVER['SERVER_NAME']) && $_SERVER['SERVER_NAME'] === 'localhost') {
    // Csak localhost esetén történjen a szerver időnullázás, ahol a fejlesztés zajlik
    $current_time = time();
    file_put_contents($serverStartTimeFile, $current_time);
}

// Szerver start time lekérdezése
$serverStartTime = (int) file_get_contents($serverStartTimeFile);

// Az uptime kiszámolása
$uptimeInSeconds = time() - $serverStartTime;

// Az uptime különböző formátumokban megjelenítése
$uptimeDays = floor($uptimeInSeconds / (60 * 60 * 24));
$uptimeHours = floor(($uptimeInSeconds % (60 * 60 * 24)) / (60 * 60));
$uptimeMinutes = floor(($uptimeInSeconds % (60 * 60)) / 60);

```

1.6.15 user.php

A **user.php** felhasználói beállítások felületet nyújt, ahol az adminisztrátorok megtekinthetik, szűrhetik, szerkeszthetik és kezelhetik a felhasználói fiókokat a FitNet rendszerben. Ez az oldal lehetővé teszi az adminisztrátorok számára olyan tevékenységek végrehajtását, mint például a felhasználók keresése felhasználónév alapján, felhasználói adatok szerkesztése és felhasználók tiltása/feloldása.

Fájlstruktúra

- **filter_users.php**: Szerveroldali szkript, amely az AJAX kéréseket kezeli a felhasználók szűrésére felhasználónév alapján.
- **toggle_ban.php**: Szerveroldali szkript, amely az AJAX kéréseket kezeli a felhasználók tiltásának/feloldásának kapcsolásához.
- **save_user.php**: Szerveroldali szkript, amely az AJAX kéréseket kezeli az szerkesztett felhasználói adatok mentéséhez.


Funkcionalitás

1. **Felhasználók Keresése**: Az adminisztrátorok felhasználókra kereshetnek, beírva egy felhasználónevet a keresőmezőbe, majd az "Keresés" gombra kattintva.

```
function filterUsers() {
    var username = document.getElementById('username').value;

    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("cardContainer").innerHTML = this.responseText;
        }
    };
    xhttp.open("GET", "filter_users.php?username=" + username, true);
    xhttp.send();
}
```

2. **Felhasználói Kártyák Megtekintése:** A felhasználói információkat kártyákban jelenítik meg, beleértve a felhasználónevet, avatart, e-mail címet, regisztrációs dátumot és műveleteket (felhasználói adatok szerkesztése, felhasználó tiltása/feloldása).



Tibor

Email: tibi@gmail.com

Registration: 2024-04-16 20:52:33

Edit user data
Ban

Username: Tibor
Email: tibi@gmail.com
Permission: 1

Save
Back

3. **Lapozás:** A felhasználókat oldalakra osztják fel a nagy felhasználói listák könnyebb navigálása érdekében.
4. **Felhasználói Adatok Szerkesztése:** Az adminisztrátorok szerkeszthetik a felhasználói adatokat, például felhasználónév, e-mail és jogosultsági szint. Az szerkesztett adatok AJAX kérés segítségével mentésre kerülnek.

```

function editUser(username, email) {
    var editForm = document.getElementById('editForm-' + username);
    editForm.style.display = 'block';

    var editUsernameInput = document.getElementById('editUsername-' + username);
    var editEmailInput = document.getElementById('editEmail-' + username);
    var editPermissionInput = document.getElementById('editPermission-' + username);

    editUsernameInput.value = username;
    editEmailInput.value = email;
    editPermissionInput.value = userData[username].originalPermission;
}

function saveUser(username, event) {
    event.preventDefault();

    var editUsernameInput = document.getElementById('editUsername-' + username);
    var editEmailInput = document.getElementById('editEmail-' + username);
    var editPermissionInput = document.getElementById('editPermission-' + username);

    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            console.log(this.responseText);
        }
    };
    xhttp.open("POST", "save_user.php", true);
    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhttp.send("username=" + editUsernameInput.value + "&email=" + editEmailInput.value + "&permission=" + editPermissionInput.value);
    alert("Save successful!");
    filterUsers();
}

```

5. **Tiltás Státusz Váltása:** Az adminisztrátorok tilthatják vagy feloldhatják a felhasználókat az adott felhasználó kártyáján található megfelelő gomb megnyomásával. Ez a művelet AJAX kéréssel történik.

```

var userData = userData || {};
function toggleBan(username, isBanned) {
    var confirmation = confirm("Are you sure you want to " +
        (isBanned == 1 ? "unban" : "ban") + " the user?");
    if (confirmation) {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                console.log(this.responseText);
                filterUsers();
            }
        };
        xhttp.open("POST", "toggle_ban.php", true);
        xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xhttp.send("username=" + username + "&isBanned=" + isBanned);
    }
}

```


1.6.16 filter_users.php

A `filter_users.php` lehetővé teszi a felhasználók keresését és szűrését a rendszerben tárolt adatok alapján.

Működés

1. **Adatbázis kapcsolat inicializálása:** Az `include("db.php")` sor importálja az adatbáziskapcsolatot biztosító fájlt.
2. **Szűrési feltétel meghatározása:** Az oldal megvizsgálja, hogy van-e keresési paraméter a GET kérésben. Ha van, a felhasználónév alapján keresés lesz végrehajtva a `username` mezőben.

```
if (isset($_GET['username']) && !empty($_GET['username'])) {  
    $username = $_GET['username'];  
    $whereClause = " WHERE username LIKE '%$username%'";  
}
```

3. **Felhasználók lekérése az adatbázisból:** Az SQL lekérdezés összeállítja a megfelelő SELECT parancsot a megadott szűrési feltételek alapján, majd lefuttatja azt az adatbázison.

```
$sql_select_users = "SELECT id, username, avatar, email,  
    is_banned, permission, reg_time FROM user" . $whereClause;  
$result_users = $conn->query($sql_select_users);
```

4. **Felhasználók megjelenítése:** Ha a lekérdezés eredménye nem üres, akkor minden talált felhasználót megjelenítünk. Minden felhasználóhoz egy kártyát rendelünk, amely tartalmazza a felhasználó adatait és különböző műveletek gombokat.
5. **Adatbázis kapcsolat bezárása:** Az adatbáziskapcsolat bezárása a művelet végén.

6.

1.6.17 toggle_ban.php

A toggle_ban.php a felhasználók kitiltását vagy kitiltásának feloldásának műveletét kezeli az adatbázisban. A toggle_ban.php fogad POST kéréseket, amelyek tartalmazzák a felhasználó nevét (**username**) és az új kitiltási állapotot (**isBanned**). Az adatbázisban a megfelelő felhasználó kitiltásának vagy feloldásának frissítése után visszajelzést ad a művelet sikerességéről.

```
<?php
session_start();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST['username'];
    $isBanned = $_POST['isBanned'];

    // Az adatbázis kapcsolódás
    include("db.php");

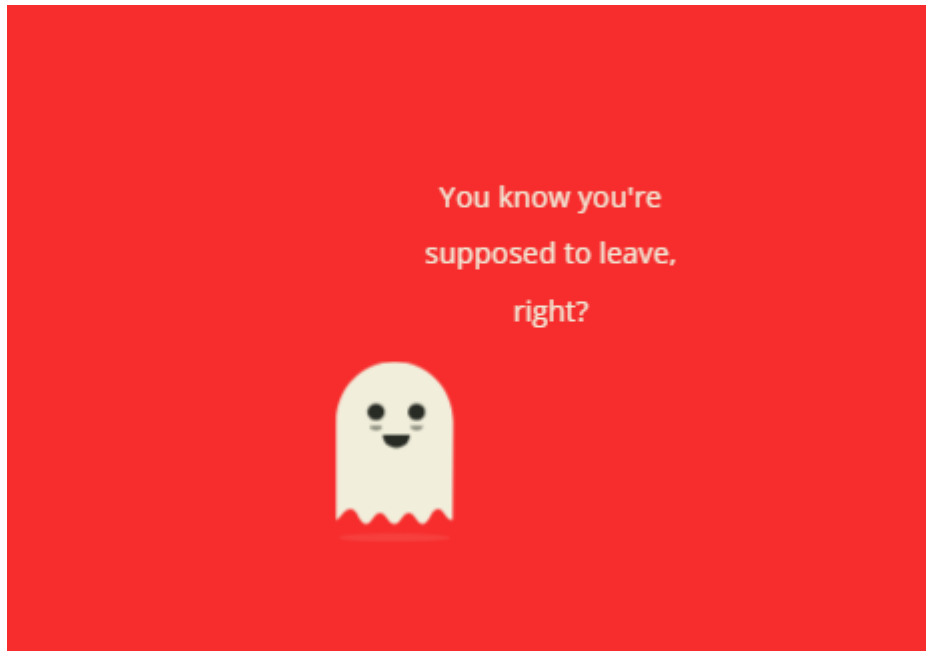
    // Az adatbázisban a felhasználó kitiltásának/feloldásának frissítése
    $stmt = $conn->prepare("UPDATE user SET is_banned = ? WHERE username = ?");
    $newBanStatus = ($isBanned == 1) ? 0 : 1;
    $stmt->bind_param("is", $newBanStatus, $username);
    $stmt->execute();
    $stmt->close();

    $conn->close();

    // Példa: sikeres üzenet küldése
    echo "Ban status toggled successfully!";
    echo "<script><alert>Ban successful!</script>";
} else {
    header("Location: ../admin_pages/admin.php");
    exit;
}
?>
```

1.6.18 banned.php

A `banned.php` oldal arra szolgál, hogy értesítést küldjön azoknak a felhasználóknak, akiket kitiltottak a FitNet alkalmazásban. Az oldal vizuálisan vonzó üzenetet jelenít meg, animált elemekkel kiegészítve, hogy fokozza a felhasználói élményt, és kifejezze a helyzet súlyát.



Javascript funkcionalitás

- **DOM Manipuláció:** JavaScript-et használnak a DOM elemek manipulálására az animációk céljából.
- **Animáció Időtartama:** Az animációk időtartama dinamikusan kiszámítódik a CSS-ben megadott egyéni tulajdonság alapján.

```
// target the elements in the DOM used in the project
const ghost = document.querySelector(".ghost");
const heading = document.querySelector("h1");
const paragraph = document.querySelector("p");
// for the length of the timeout, consider the
// --animation-duration custom property and add a small delay
// retrieve properties on the root element
const root = document.querySelector(":root");
const rootStyles = getComputedStyle(root);
// retrieve the animation-duration custom property
// ! this is specified as "40s", in seconds,
// so parse the number and include it in milliseconds
const animationDuration = parseInt(rootStyles.getPropertyValue("--animation-duration")) * 1000;
```

1.6.19 save_user.php

A **save_user.php** felelős a felhasználói adatok frissítéséért az adatbázisban a POST kérés során kapott információk alapján. Kezeli a felhasználónév, e-mail cím és jogosultsági szint frissítését a bejelentkezett felhasználó számára.

Működés

1. **Munkamenet indítása:** A munkamenet elindításával kezdődik "session_start()", hogy fenntartsa a felhasználói munkamenet adatokat.
2. **Kérés-módszer ellenőrzése:** A szkript megvizsgálja a kérés módszerét.
3. **Adatbázis csatlakozás:** A beágyazott db.php végzi az adatbázis kapcsolódást.

```
<?php
session_start();
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $editedUsername = $_POST["username"];
    $editedEmail = $_POST["email"];
    $editedPermission = $_POST['permission'];

    include("db.php");

    $user_id = $_SESSION["user_id"];
    // Update the user table with the new username and email
    $sql_update_user = "UPDATE user SET username = '$editedUsername',
    email = '$editedEmail', permission = '$editedPermission' WHERE id = '$user_id'";

    if ($conn->query($sql_update_user) === true) {
        $_SESSION['username'] = $editedUsername;
        $_SESSION['email'] = $editedEmail;
        $_SESSION['permission'] = $editedPermission;
        echo "User data saved successfully!";
    } else {
        echo "Error updating user data: " . $conn->error;
    }

    $conn->close();
} else {
    // Handle invalid request method
    http_response_code(405);
    echo "Invalid request method";
}
?>
```

1.6.20 announcements.php

A FitNet `announcements.php` oldala egy platformként szolgál közlemények küldésére és törlésére. Az adminoknak lehetőséget biztosít új közlemények létrehozására és meglévő közlemények listájának megtekintésére, valamint azok törlésére.

Announcements küldése

- Az adminok új közlemények nevét, szövegét és dátumát adhatják meg.
- Bemeneti mezők:
 - Közlemény neve (`announcementName`)
 - Közlemény szövege (`announcementInput`)
 - A közlemény dátuma és ideje (`announcementDate`)
- Az adatok beküldésekor a POST kérés segítségével továbbítják az adatokat a `upload_announcements_work.php` fájlba.

Announcements törlése és listája

- Az oldal megjeleníti a már meglévő közlemények listáját.
- Minden közleményhez egy törlés gomb is tartozik.
- A törlés gombra kattintva a közleményt törlik a `delete_announcements_work.php` fájl segítségével.

```
<?php
include ("db.php");

$sql = "SELECT id, announcementsText, announcementName, date FROM announcements";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        echo "<form action='delete_announcements_work.php' method='get'>";
        echo "<input type='hidden' name='announcementId' value='" . $row['id'] . "'>";
        echo "<div style='background-color: #f9f4f4; padding: 15px; border-radius: 10px; margin-top: 10px;'>";
        echo "<h2 style='text-align: center;'>Announcement: " . $row['announcementName'] . "</h2>";
        echo "<h2 style='text-align: center;'>Date: " . $row['date'] . "</h2>";
        echo "<div style='text-align: center; color: red; ' class='text-x1'>" . $row['announcementsText'] . "</div>";
        echo "<input type='submit' value='Delete announcement' class='button'>";
        echo "</div>";
        echo "</form>";
    }
} else {
    echo "<p>There is no announcement to delete.</p>";
}
?>
```

1.6.21 upload_announcements.php

Az `upload_announcements.php` a FitNet rendszerben felelős a felhasználók által beküldött új közlemények kezeléséért és azok adatbázisba történő mentéséért. Emellett értesítéseket is generál a közlemény létrehozása után, hogy a felhasználók tudjanak a friss hírekről.

Működés

- Az oldal a `session_start()` függvénnyel elindítja a munkamenetet és importálja az adatbázis kapcsolódást a `db.php` fájlból.
- Az űrlapból érkező adatokat változóba menti, mint például `announcementName`, `announcementInput` és `announcementDate`.
- Az SQL lekérdezéssel az adatokat beszúrja az `announcements` táblába.
- Ha sikeres volt a beszúrás, a felhasználókhöz kapcsolódó értesítéseket is hozzáadja a `notifications` táblához.
- A sikeres vagy sikertelen folyamat végén a felhasználót visszairányítja az adminisztrációs oldalra (`admin.php?page=announcements`), és megjelenít egy értesítő üzenetet az eredményről.

1.6.22 delete_announcements_work.php

A `delete_announcements_work.php` oldal a FitNet rendszerben felelős a közlemények és azokhoz kapcsolódó értesítések törléséért az adatbázisból. Ez az oldal az adminisztrációs felületen keresztül hozzáférhető, és lehetőséget biztosít az adminoknak a közlemények és értesítéseik kezelésére.

Működés

- Az oldal az `include("db.php");` sorral importálja az adatbázis kapcsolódást.
- Ellenőrzi, hogy az URL-en át lett-e adva az "announcementId" paraméter.
- Ha az "announcementId" paraméter át lett adva, végrehajtja a törlést mind a `notifications` táblából, mind pedig az `announcements` táblából.
- Ha a törlés sikeres volt, JavaScript kódot futtat, amely frissíti az oldalt és megjelenít egy sikeres törlésről szóló üzenetet.
- Ha bármelyik törlési művelet sikertelen volt, a JavaScript kód hibaüzenettel tér vissza.
- Ha az "announcementId" paraméter nem lett átadva vagy hibás, egy "Unknown id or wrong." üzenet jelenik meg.

1.6.23 exercise_upload.php

Az `exercise_upload.php` fájl egy HTML űrlapot tartalmaz, amely lehetővé teszi felhasználók számára, hogy gyakorlatokat töltsenek fel az adatbázisba.

Exercise upload

Name of exercise:

Description of exercise:

Short description of exercise:

Category of the workout: Weight lifting ▼ +

Body part of the workout: Upper body ▼

Picture of exercise: Fájl kiválasztása Nincs fájl kiválasztva

Upload

Javascript függvények

1. `addSelect(className):`

- Ez a függvény felelős az új kategória vagy testrész mező hozzáadásáért az űrlaphoz.
- Paraméterként megkapja a mezők osztálynevét, amelyekhez hozzá kell adni az új mezőt.
- AJAX hívást használ a `get_categories_count.php` fájlhoz, hogy meghatározza, hány további mező adható hozzá.
- Ellenőrzi, hogy van-e még elérhető kategória, és ha van, akkor hozzáad egy új mezőt az űrlaphoz.
- Frissíti az újonnan hozzáadott mező kiválasztható lehetőségeit, hogy ne lehessen többször ugyanazt választani.

2. `updateOptions(newSelect):`

- Ez a függvény felelős a már kiválasztott lehetőségek eltárolásáért és eltávolításáért az új mezőből.
- Megkapja az új mezőt paraméterként, és eltávolítja belőle azokat az opciókat, amelyek már korábban kiválasztásra kerültek.

3. **createDeleteButton(selectElem):**
 - Dinamikusan létrehoz egy törlési gombot az adott mezőhöz.
 - Paraméterként megkapja a mezőt, amelyhez a törlési gombot hozzá kell adni.
 - Visszatér egy új törlési gomb elemmel, amely hozzá van rendelve a megadott mezőhöz.
4. **updateDeleteButton():**
 - Ez a függvény felelős a törlési gomb megjelenítésének vagy elrejtésének frissítéséért az űrlapon.
 - Ellenőrzi, hogy van-e több mező az adott típusból, és ha van, hozzáadja a törlési gombot az utolsó mezőhöz.
 - Ha nincs több mező az adott típusból, eltávolítja a törlési gombot az űrlapról.

1.6.24 get_categories_count.php

A **get_exercise_count.php** a kategóriák számának lekérdezését és visszaküldését végzi JSON formátumban.

Működése

1. **Adatbázis kapcsolat felépítése:** A szkript először csatlakozik az adatbázishoz a **db.php** fájl segítségével.
2. **Kategóriák számának lekérdezése:** Az adatbázisból lekéri a **workout_categories** táblában található kategóriák számát.
3. **Visszatérés JSON formátumban:** A kategóriák számát JSON formátumban küldi vissza a kliens oldalnak a **categoryCount** kulcs alatt.
4. **Adatbázis kapcsolat lezárása:** A szkript végén lezárja az adatbázis kapcsolatot.

```
<?php
include("db.php");

// Adatok lekérdezése a workout_category táblából
$sqlCount = "SELECT COUNT(*) as count FROM workout_categories";
$resultCount = $conn->query($sqlCount);
$rowCount = $resultCount->fetch_assoc();
$categoryCount = $rowCount['count'];

// Visszaküldjük a kategóriák számát JSON formátumban
echo json_encode(array('categoryCount' => $categoryCount));

// Adatbázis kapcsolat lezárása
$conn->close();
?>
```


1.6.25 exercis_upload_work.php

Az `exercise_upload_work.php` fájl egy szerveroldali PHP szkript, amely felelős a gyakorlatok feltöltését kezelő logika végrehajtásáért.

Funkciók

1. **Session indítása:** Az első sorban a `session_start()` függvény segítségével indítja el a munkamenetet, hogy a szkript később tárolhassa a feltöltött kép nevét.
2. **Űrlap adatok feldolgozása:** Ellenőrzi, hogy a kliens oldalról POST kérés érkezett-e az `upload` nevű gombra kattintva, és hogy a kérés tartalmazza-e a feltöltött fájlt.
3. **Feltöltött fájl kezelése:** Ellenőrzi a feltöltött fájl méretét és típusát. Ha megfelelőek, a fájlt áthelyezi a `../uploads/exercises/` mappába, és egyedi nevet ad neki.
4. **Űrlap adatok kezelése:** Feldolgozza az űrlapról érkező adatokat, mint például a gyakorlat neve, leírása, rövid leírása, kategóriája és testrésze.
5. **Adatok beszúrása az adatbázisba:** Beszúrja a gyakorlat adatait a `workout_exercises` táblába, majd a kapcsolódó kategóriákat az `exercise_category` táblába.
6. **Visszatérés az admin felületre:** Ha minden sikeres volt, az admin felületre irányítja a felhasználót, és sikeres feltöltésről értesítő üzenetet jelenít meg. Ha bármilyen hiba történt, hibaüzenetet jelenít meg és visszairányítja a felhasználót az admin felületre.
7. **Adatbázis kapcsolat lezárása:** Lezárja az adatbázis kapcsolatot a szkript végén.

1.6.26 filter_exercise.php

A `filter_exercise.php` felelős az edzésgyakorlatok szűréséért és megjelenítéséért az adatbázisból.

Funkciók

1. **Adatbázis kapcsolat:** Az első részben a `db.php` csatlakozik az adatbázishoz.
2. **Keresési feltételek ellenőrzése:** Ellenőrzi, hogy van-e keresési paraméter a GET kérésben (pl. `exercisename`), és ha van, létrehozza a megfelelő WHERE feltételt az SQL lekérdezéshez.
3. **Edzésgyakorlatok lekérdezése:** Összeállítja az SQL lekérdezést az edzésgyakorlatok lekérdezésére, figyelembe véve a keresési feltételeket.
4. **Edzésgyakorlatok megjelenítése:** Ha találhatóak edzésgyakorlatok a lekérdezés eredményében, megjeleníti azokat egy HTML struktúrában, beleértve a gyakorlatok nevét, leírását, rövid leírását és képét.
5. **Adatbázis kapcsolat lezárása:** A kód végén lezárjuk az adatbázis kapcsolatot (`$conn->close();`).

1.6.27 edit_exercise.php

Az `edit_exercises.php` egy PHP alapú weboldal, amely lehetővé teszi gyakorlatok szerkesztését az adatbázisban. Az oldal két fő üzemmódban működik:

1. **Gyakorlat szerkesztése:** Az oldal betöltésekor megjelenít egy legördülő listát az összes elérhető gyakorlatról. A felhasználó kiválaszthat egy gyakorlatot, és szerkesztheti annak adatait.
2. **Gyakorlat frissítése:** Ha a felhasználó elküldi a szerkesztett gyakorlat adatait, az oldal frissíti az adatbázisban tárolt információkat.

Edit Exercise

Select Exercise to Edit: benchpress ▼



Edit

Name of exercise:

benchpress

Description of exercise:

benchpress 10x8

Short description of exercise:

make benchpress

Category of the workout: Weight lifting ▼ +

Body part of the workout: Upper body ▼

Picture of exercise: Nincs fájl kiválasztva

Update

1.6.28 save_user.php

A `save_user.php` felelős az adatbázisban tárolt felhasználói adatok frissítéséért az adott felhasználó módosítása esetén.

Működés

1. **Session kezelése:** Az első lépés a session kezelésének megkezdése. A `session_start()` függvény meghívásával az aktuális munkamenet inicializálódik vagy folytatódik, amennyiben már meglévő session van jelen.
2. **HTTP POST ellenőrzése:** Az fájl ellenőrzi, hogy az érkező kérés típusa POST-e vagy sem. Ha az érkező kérés POST típusú, akkor a fájl folytatja a feldolgozást, különben hibát jelez a 405-es HTTP válasszal.
3. **Adatok feldolgozása:** A beérkező POST kérések alapján az új felhasználói adatokat (felhasználónév, email cím és jogosultság) eltároljuk a megfelelő változóknak.
4. **Adatbázis kapcsolat megnyitása:** A `db.php` fájl beillesztésével létrehozuk a kapcsolatot az adatbázissal.
5. **Felhasználói adatok frissítése:** Az SQL lekérdezés segítségével frissítjük a `user` táblát az új felhasználói adatokkal a megfelelő felhasználó azonosító alapján.
6. **Sikeres művelet kezelése:** Ha a lekérdezés sikeresen lefut, az új felhasználói adatokat elmentjük a session változóknak, és visszajelzünk a felhasználónak a sikeres mentésről.
7. **Hiba kezelése:** Amennyiben a lekérdezés sikertelen volt, hibát jelezünk vissza az adatbáziskapcsolat hibájával.
8. **Adatbázis kapcsolat bezárása:** Végül bezárjuk az adatbáziskapcsolatot a `close()` metódus meghívásával.

1.6.29 delete_exercis.php

Az `delete_exercise.php` lehetővé teszi az adminok számára, hogy töröljék az edzéshez társított gyakorlatokat egy adatbázisból. Az oldal lehetővé teszi az edzéshez rendelt gyakorlatok listájának megjelenítését, és lehetővé teszi a felhasználók számára, hogy kiválasszák és töröljék azokat.

Delete Exercise

Select Exercise to Delete: benchpress ▼



Delete

1.6.30 delete_exercise_work.php

A `delete_exercise_work.php` felelős az edzéshez társított gyakorlat törléséért az adatbázisból és a hozzá tartozó kép fizikai törléséért a fájlszerverről. Az oldal az `id` paraméter alapján azonosítja és törli a megfelelő gyakorlatot az adatbázisból.

Funkciók

1. **Adatbázisból való törlés:** Az oldal eltávolítja a megadott `id`-vel azonosított gyakorlatot az adatbázisból.
2. **Kép fizikai törlése:** Amennyiben a törölni kívánt gyakorlathoz tartozik kép, az oldal törli azt a fájlszerverről.
3. **Értesítés:** A felhasználót értesíti a művelet eredményéről JavaScript segítségével.

1.6.31 topics_and_comments.php

A `topics_and_comments.php` fájl egy olyan weboldal része, amelyen témákat és hozzájuk tartozó kommenteket lehet megtekinteni és kezelni.

Php kód

- A PHP kód az adatbázisból lekéri és megjeleníti a témákat és hozzájuk tartozó kommenteket.
- A `db.php` fájlban található adatbázis kapcsolatot hívja meg.
- A témák lekérdezése után a PHP kód végigmegy mindegyiken, megjelenítve a címet, a létrehozás dátumát, a felhasználó nevét vagy az "anonymous" szót, a témához tartozó rövid leírást és a hozzátartozó kommenteket.
- A kommenteket a felhasználó nevével és létrehozás dátumával együtt jeleníti meg, és minden komment mellett elérhető egy törlés gomb.

```
$comment_query = "SELECT comments.*, user.username
FROM comments
LEFT JOIN user ON comments.user_id = user.id
WHERE topic_id = $topicId
ORDER BY comments.created_at DESC";
```

1.6.32 delete_topic.php

A `delete_topic.php` oldal felelős a témák és azokhoz kapcsolódó kommentek törléséért az adatbázisból.

Fájl strukturája és működése

- A fájl elején a `db.php` fájl beilleszti az adatbázis kapcsolatot.
- Az oldal ellenőrzi, hogy a `topic_id` értéke be lett-e állítva a GET kérésben.
- Ha a `topic_id` be van állítva, akkor a következő lépések történnek:
 1. A `topic_id` értékét biztonságosan kezelik a `mysqli_real_escape_string()` függvény segítségével, hogy elkerüljék az SQL injekciókat.

2. Törlik a megadott témához tartozó összes kommentet az `comments` táblából.
3. Törlik magát a témát a `topics` táblából.
 - Az oldal visszajelzést ad a felhasználónak a törlés sikerességéről vagy sikertelenségéről.
 - Ha nem lett megadva `topic_id`, akkor az oldal hibát jelez.
 - Az `isset()` függvénnyel ellenőrzi, hogy a `topic_id` be lett-e állítva a GET kérésben.
 - Az SQL lekérdezések biztonságosak, mivel a felhasználói bemenetet megtisztítják a `mysqli_real_escape_string()` függvénnyel.
 - Az `echo` utasításokkal ad visszajelzést a felhasználónak a műveletek eredményéről.

1.6.33 delete_comment.php

A `delete_comment.php` felelős a kommentek törléséért az adatbázisból.

Fájl struktúrája és működése

- A fájl elején a `db.php` fájl beilleszti az adatbázis kapcsolatot.
- Az oldal ellenőrzi, hogy a `comment_id` érték át lett-e adva a GET kérésben.
- Ha a `comment_id` be lett adva, akkor a következő lépések történnek:
 - A `comment_id` értékét kinyerik a GET kérésből.
 - Végrehajtják a komment törlési műveletet az adatbázisban a `comments` táblából, az adott `comment_id`-re vonatkozóan.
 - Ellenőrzik, hogy a törlés sikeres volt-e.
 - Ha a törlés sikeres volt, átirányítják az adminisztrációs oldalra és visszajelzést adnak a felhasználónak a sikeres törlésről.
 - Ha a törlés nem sikerült, hibaüzenetet jelenítenek meg az SQL hibával együtt.
- A fájl PHP kódot tartalmaz, amely az adatbázis műveleteket végzi.
- Az `isset()` függvénnyel ellenőrzik, hogy a `comment_id` be lett-e adva a GET kérésben.
- Az SQL lekérdezés biztonságos, mivel nem kezel felhasználói bemenetet, és nincs lehetőség SQL injekcióra.
- Az `echo` utasításokkal visszajelzést adnak a felhasználónak a műveletek eredményéről, beleértve a sikeres törlést vagy a hibaüzenetet.

1.6.34 logout.php

A `logout.php` felelős a felhasználó kijelentkeztetéséért a webalkalmazásból.

Fájl struktúrája és működése

- A fájl elején megkezdődik a PHP session kezelése a `session_start()` függvénnyel, amely lehetővé teszi a session változók használatát.

- A `session_unset()` függvénnyel törlik a session változókat, hogy megszabaduljanak a felhasználóhoz kötődő adatoktól.
- A `session_destroy()` függvénnyel pedig megszüntetik a sessiont és az összes hozzá kapcsolódó adatot.
- Végül a `header()` függvénnyel az oldalátírányítás segítségével visszatérnek a bejelentkező oldalra, ami a `landing.php` oldal. Az `exit` függvénnyel azonnal leállítják a fájl végrehajtását, így nem folytatódik további kódvégrehajtás.

```
<?php
session_start();
session_unset();
session_destroy();
header("Location: ../user_pages/landing.php");
exit;
?>
```

1.6.35 add_comment.php

Az `add_comment.php` felelős új kommentek hozzáadásáért az adatbázishoz, és ezek megjelenítéséért az oldalon, amennyiben a felhasználó be van jelentkezve.

Működése

1. **Session kezelés:** Első lépésként a session kezelését megkezdi a `session_start()` függvény segítségével. Ez lehetővé teszi, hogy a felhasználó bejelentkezési állapotát tároljuk és kezeljük.
2. **Adatbázis kapcsolat beállítása:** Az oldal tartalmaz egy `db.php` fájlt, ami valószínűleg az adatbázis kapcsolódási adatokat tartalmazza, és ezt beolvassa az `include("db.php")` sorral.
3. **Bejelentkezés ellenőrzése:** Ellenőrzi, hogy a felhasználó be van-e jelentkezve. Ha a `$_SESSION['username']` be van állítva, akkor az oldal tovább lép.
4. **Paraméterek ellenőrzése:** Ellenőrzi, hogy a szükséges paraméterek (`topic_id` és `comment`) be vannak-e küldve a POST kérésben.
5. **Felhasználó ID lekérdezése:** Lekérdezi a felhasználó ID-ját az adatbázisból a felhasználónév alapján.
6. **Komment hozzáadása az adatbázishoz:** Az adatokat beszúrja a `comments` táblába az adatbázisba. A `user_id`, `topic_id`, `content` és `created_at` mezőket feltölti a megfelelő értékekkel. A `NOW()` függvény segítségével az aktuális dátum és idő kerül beillesztésre.
7. **Eredmény kiírása:** Ha sikeres volt a beszúrás az adatbázisba, akkor az új kommentet megjeleníti az oldalon egy `<div>` elembe rendezve. A komment tartalmazza a felhasználó nevét, a komment dátumát, magát a kommentet, valamint egy gombot a komment törléséhez. A komment törlésének gombja egy JavaScript függvényt hív meg, amely egy törlési funkciót kezdeményez. Ha a beszúrás sikertelen, akkor hibaüzenet jelenik meg.
8. **Hibakezelés:** Különböző hibás esetekre külön hibaüzeneteket ír ki, például ha hiányoznak a szükséges paraméterek, ha a felhasználó nincs bejelentkezve, vagy ha a felhasználó nem található az adatbázisban.
9. **Adatbázis kapcsolat lezárása:** Végül lezárja az adatbázis kapcsolatot a `mysqli_close($conn)` függvénnyel.

1.6.36 avatar_upload.php

Az `avatar_upload.php` felelős a felhasználó által kiválasztott avatar kép feltöltéséért és az adatbázisban való tárolásáért.

Működés

1. Első lépésként a fájl megkezdi a PHP session indítását a `session_start()` függvénnyel.
2. Ellenőrzi, hogy a `$_FILES['avatar']` és `$_POST['upload']` változók léteznek-e. Ha igen, folytatja a folyamatot, különben a felhasználót visszairányítja a `menu.php` oldalra.
3. Az adatbázis kapcsolatát meghívja az `include("db.php");` sorral.
4. Az `$_FILES` tömb segítségével kinyeri a feltöltött fájl nevét, méretét, ideiglenes nevét és hibakódját a `$_FILES['avatar']` tömbből.
5. A felhasználó nevét kinyeri az aktuális PHP sessionból: `$user = $_SESSION['username'];`.
6. Ellenőrzi, hogy volt-e hiba a feltöltött fájlban. Ha nem volt hiba (`$error === 0`), akkor folytatja a folyamatot, különben hibaüzenetet küld a `menu.php` oldalra.
7. Ellenőrzi, hogy a feltöltött fájl mérete nem haladja-e meg a megengedett maximumot (400000 byte). Ha túl nagy, akkor hibaüzenetet küld a `menu.php` oldalra.
8. Ellenőrzi, hogy a feltöltött fájl kiterjesztése megfelel-e az elfogadott kiterjesztéseknek (`jpg`, `jpeg`, `png`). Ha nem, akkor hibaüzenetet küld a `menu.php` oldalra.
9. Ha minden ellenőrzés sikeres volt, akkor a fájlt átnevezi egyedi azonosítóval, majd a `../uploads/` mappába helyezi a `move_uploaded_file()` függvénnyel.
10. A fájl nevét beilleszti az adatbázisba az aktuális felhasználóhoz tartozó rekordba az `UPDATE user SET avatar = '$new_img_name' WHERE username = '$user'` SQL parancs segítségével.
11. Ha az adatbázisba történő beszúrás sikeres volt, akkor frissíti a felhasználó sessionját az új kép nevével (`$_SESSION['avatar'] = $new_img_name;`) és visszairányítja a felhasználót a `menu.php` oldalra.
12. Ha az adatbázisba történő beszúrás sikertelen volt, akkor hibaüzenetet küld a `menu.php` oldalra.
13. Ha a fájl feltöltése közben bármilyen ismeretlen hiba történt, akkor hibaüzenetet küld a `menu.php` oldalra.
14. Ha a feltöltés gomb nem volt megnyomva, akkor visszairányítja a felhasználót a `menu.php` oldalra.

1.6.37 check_access.php

A `check_access.php` feladata, hogy ellenőrizze, hogy egy felhasználó be van-e jelentkezve, és hogy nincs-e kitiltva. Ha valamelyik feltétel nem teljesül, akkor átirányítja a felhasználót a megfelelő oldalra.

Működése

1. `session_start();`: Ez a függvény elindítja a munkamenetet, ami lehetővé teszi az adatok tárolását és hozzáférését a különböző oldalak között.
2. `if (!isset($_SESSION['loggedin']) || $_SESSION['loggedin'] !== true) {`: Ez a feltétel ellenőrzi, hogy a `$_SESSION` tömbben van-e `loggedin` kulcs, és ha van, akkor azt is ellenőrzi, hogy az értéke `true`-e. Ha ez nem teljesül (tehát a felhasználó nincs bejelentkezve), akkor átirányítja a felhasználót a bejelentkezési oldalra (`../user_pages/landing.php`), majd leállítja a további végrehajtást (`exit;`).
3. `if (isset($_SESSION['is_banned']) && $_SESSION['is_banned'] == 1) {`: Ez a feltétel ellenőrzi, hogy a `$_SESSION` tömbben van-e `is_banned` kulcs, és ha van, akkor azt is ellenőrzi, hogy az értéke `1`. Ha ez a feltétel teljesül (tehát a felhasználó kitiltva van), akkor átirányítja a felhasználót a kitiltott felhasználók oldalra (`../user_pages/banned.php`), majd leállítja a további végrehajtást (`exit;`).
4. A kód végére érve nincs további művelet, tehát ha egyik feltétel sem teljesült, akkor a felhasználó a `check_access.php` fájl hívása után marad, és folytathatja a műveleteket.

1.6.38 create_topic.php

A `create_topic.php` felelős az új fórumtéma létrehozásáért.

Működése

1. `session_start();`: Ez a függvény elindítja a munkamenetet, ami lehetővé teszi az adatok tárolását és hozzáférését a különböző oldalak között.
2. `include("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (`db.php`), amely tartalmazza az adatbázis kapcsolódásához szükséges információkat és az adatbázis kapcsolat létrehozását.
3. `if ($_SERVER["REQUEST_METHOD"] == "POST") {`: Ez a feltétel ellenőrzi, hogy a kérés típusa POST-e, vagyis hogy az űrlap elküldte-e az adatokat.
4. `if (isset($_SESSION['username'])) {`: Ez a feltétel ellenőrzi, hogy a felhasználó be van-e jelentkezve, azaz hogy van-e `username` kulcs a `$_SESSION` tömbben.
5. `$username = $_SESSION['username'];`: Az éppen bejelentkezett felhasználó nevét elmentjük a `$username` változóba.
6. `$query = "SELECT id FROM user WHERE username = '$username'";`: Lekérdezzük az adatbázisból az éppen bejelentkezett felhasználó azonosítóját (`id`) a felhasználónév alapján.
7. `if (mysqli_num_rows($result) == 1) {`: Ellenőrizzük, hogy a lekérdezés eredménye egyetlen sort adott-e vissza, vagyis hogy létezik-e ilyen felhasználó az adatbázisban.
8. `$row = mysqli_fetch_assoc($result);`: Elmentjük a lekérdezés eredményét egy asszociatív tömbbe.

9. `$user_id = $row['id'];`: Az éppen bejelentkezett felhasználó azonosítóját elmentjük a `$user_id` változóba.
10. `if(isset($_POST['anonymus']) && $_POST['anonymus'] == 'anonymus')`: Ellenőrizzük, hogy az űrlapból érkezett-e anonim jelölőnégyzet (**checkbox**), és ha igen, akkor az azonosítót nullázzuk (0).
11. Az új téma címét (**title**) és leírását (**description**) a POST kérésből olvassuk ki.
12. `if (mysqli_num_rows($result) == 0) {`: Ellenőrizzük, hogy az adott címmel már létezik-e téma az adatbázisban.
13. `if (mysqli_query($conn, $query)) {`: Ha az adott címmel még nem létezik téma az adatbázisban, akkor hozzáadjuk az új témát az adatbázishoz.
14. `header("Location: menu.php?page=forum");`: Sikeres létrehozás esetén a felhasználót visszairányítjuk a fórum főoldalára.
15. A különböző hiba esetekben (például: ha már létezik ilyen címmel téma az adatbázisban, ha a felhasználó nem található az adatbázisban vagy ha nem volt bejelentkezve) megfelelő üzenetet jelenítünk meg vagy átirányítjuk a felhasználót a megfelelő oldalra.
16. `mysqli_close($conn);`: Az adatbázis kapcsolat bezárása a kód végén.

1.6.39 delete_topic.php

A **delete_topic.php** PHP fájl feladata egy adott fórumtéma és hozzá tartozó kommentek törlése az adatbázisból.

Működése

1. `include("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (**db.php**), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
2. `if (isset($_GET['topic_id'])) {`: Ez a feltétel ellenőrzi, hogy a `$_GET` változóban van-e **topic_id** kulcs, vagyis hogy meg lett-e adva a törlendő fórumtéma azonosítója.
3. `$topicId = mysqli_real_escape_string($conn, $_GET['topic_id']);`: Az `mysqli_real_escape_string` függvénnyel biztonságosan kezeljük a `$_GET['topic_id']` változót, hogy elkerüljük az SQL injection támadásokat.
4. `DELETE FROM comments WHERE topic_id = $topicId`: Ez a query törli az adott **topic_id**-hez tartozó összes kommentet az adatbázisból.
5. `DELETE FROM topics WHERE id = $topicId`: Ez a query törli az adott **id**-jű témát az adatbázisból.
6. `if ($deleteCommentsResult && $deleteTopicResult) {`: Ellenőrizzük, hogy mindkét törlési művelet sikeres volt-e.
7. `echo "Topic and associated comments deleted successfully."`: Sikeres törlés esetén visszaküldünk egy megerősítő üzenetet.
8. Ha valamilyen hiba történt a törlés során (pl. nem volt megadva **topic_id**, hiba történt a törlés során), akkor az megfelelő hibaüzenetet jelenítünk meg.
9. `mysqli_close($conn);`: Az adatbázis kapcsolat bezárása a kód végén.

1.6.40 delete_comment.php

A `delete_comment.php` felelős egy adott komment törléséért az adatbázisból.

Működése

1. `session_start();`: Ez a függvény elindítja a munkamenetet, ami lehetővé teszi az adatok tárolását és hozzáférését a különböző oldalak között.
2. `include("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (`db.php`), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
3. `if (isset($_SESSION['username'])) {`: Ez a feltétel ellenőrzi, hogy a felhasználó be van-e jelentkezve, azaz van-e `username` kulcs a `$_SESSION` tömbben.
4. `$username = $_SESSION['username'];`: Az éppen bejelentkezett felhasználó nevét elmentjük a `$username` változóba.
5. `if (isset($_GET['comment_id'])) {`: Ellenőrizzük, hogy a `$_GET` változóban van-e `comment_id` kulcs, vagyis hogy meg lett-e adva a törlendő komment azonosítója.
6. `$comment_id = $_GET['comment_id'];`: Az `$_GET['comment_id']` változót elmentjük a `$comment_id` változóba.
7. `SELECT * FROM comments WHERE id = $comment_id AND user_id = (SELECT id FROM user WHERE username = '$username');`: Ez a query ellenőrzi, hogy a megadott komment azonosítóhoz tartozik-e komment az adatbázisban, és hogy az adott felhasználó írta-e azt.
8. `if (mysqli_num_rows($check_result) > 0) {`: Ellenőrizzük, hogy a komment megtalálható-e az adatbázisban és a felhasználóé-e.
9. `DELETE FROM comments WHERE id = $comment_id;`: Ez a query törli az adott `id`-jű kommentet az adatbázisból.
10. `if (mysqli_query($conn, $delete_query)) {`: Ellenőrizzük, hogy a törlési művelet sikeres volt-e.
11. Ha sikeres a törlés, akkor visszaküldünk egy megerősítő üzenetet.
12. Ha valamilyen hiba történt a törlés során (például: a komment nem található az adatbázisban, vagy a felhasználó nem rendelkezik megfelelő jogosultsággal), akkor megfelelő hibaüzenetet jelenítünk meg.
13. `mysqli_close($conn);`: Az adatbázis kapcsolat bezárása a kód végén.

1.6.41 fetch_week_data.php

A `fetch_week_data.php` PHP fájl felelős a felhasználó heti edzéstervének lekérdezéséért az adatbázisból, majd a kapott adatok JSON formátumban történő elküldéséért.

Működése

1. `session_start();`: Ez a függvény elindítja a munkamenetet, ami lehetővé teszi az adatok tárolását és hozzáférését a különböző oldalak között.

2. `include("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (`db.php`), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
3. `$username = $_SESSION['username'];`: Az éppen bejelentkezett felhasználó nevét elmentjük a `$username` változóba a munkamenetből.
4. `$sql = "SELECT generated_workout FROM user WHERE username = '$username'";`: Ezzel a SQL lekérdezéssel lekérdezzük az adott felhasználó heti edzéstervét a `user` táblából.
5. `$result = $conn->query($sql);`: A lekérdezést végrehajtjuk az adatbázison.
6. `if ($result->num_rows > 0) {`: Ellenőrizzük, hogy a lekérdezés eredménye nem üres-e.
7. `$row = $result->fetch_assoc();`: Az eredmény sorát elmentjük egy asszociatív tömbbe.
8. `$workoutData = json_decode($row['generated_workout'], true);`: Az adatbázisból kapott JSON formátumú edzéstervet dekódoljuk PHP tömbbé.
9. `header('Content-Type: application/json');`: Beállítjuk a válasz fejlécét, hogy JSON adatokat küldünk vissza.
10. `echo json_encode($workoutData);`: Elküldjük a heti edzéstervet JSON formátumban.
11. `$conn->close();`: Az adatbázis kapcsolatot lezárjuk a kód végén.

1.6.42 functions.php

A `functions.php` fájlban található `generateWorkouts` függvény felelős az edzésterv generálásáért és tárolásáért az adatbázisban.

Működés

1. `function generateWorkouts($username, $intention, $workouttime) {`: A `generateWorkouts` függvény paraméterei a felhasználónév (`$username`), az edzés szándéka (`$intention`) és az edzések időtartama hetekben (`$workouttime`).
2. `global $conn;`: Az `$conn` változó globális kontextusban való használata, mely az adatbázis kapcsolatot reprezentálja.
3. `$workouts = array();`: Ebben a tömbben tároljuk majd a generált edzéstervet hetek és napok szerint.
4. A `for` ciklus segítségével generáljuk meg a feladatsorokat a megadott időtartamra. Az edzések időtartama minden egyes hétnak az elején kezdődik újra.
5. A belső `for` ciklus segítségével minden héten generálunk 7 napos edzéstervet. Az első héten az első nap kihagyása, valamint minden harmadik nap pihenőnap beiktatása történik.
6. A `date` függvénnyel kiszámítjuk az aktuális nap dátumát a héten belül.
7. SQL lekérdezéseket hajtunk végre az edzésterv összeállításához. Az `upperBodyQuery` és `lowerBodyQuery` változóknak tároljuk azokat a lekérdezéseket, melyek az upper body és lower body gyakorlatokat kérdezik le a felhasználó szándéka (`$intention`) alapján. Ezek a lekérdezések véletlenszerűen választanak ki gyakorlatokat az adatbázisból.
8. Az eredményeket feldolgozzuk, majd összeállítjuk a generált feladatsort a `dayWorkout` tömbben.
9. A generált feladatsort hozzáadjuk a heti feladatsorokhoz a `$weekWorkouts` tömbbe.
10. Az így létrejött heti feladatsorokat hozzáadjuk a teljes feladatsorokat tartalmazó `$workouts` tömbhöz.

11. A generált feladatsorokat JSON formátumba alakítjuk az `json_encode` függvénnyel.
12. Az előkészített JSON adatokat frissítjük a felhasználó rekordjában az adatbázisban.
13. **UPDATE** SQL utasítással frissítjük a `user` táblában az adott felhasználó rekordját a generált edzéstervvel.
14. A `$update_stmt->execute();` sorral végrehajtjuk az SQL frissítő műveletet.

1.6.43 submit_comment.php

A `submit_comment.php` PHP fájl felelős az új komment felvételéért az adatbázisba egy adott témához.

Működése

1. `include("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (`db.php`), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
2. `if (isset($_POST['topic_id']) && isset($_POST['username']) && isset($_POST['comment']) && !empty($_POST['topic_id']) && !empty($_POST['username']) && !empty($_POST['comment']))` { Ez a feltétel ellenőrzi, hogy a POST kérés tartalmazza-e a szükséges adatokat: a téma azonosítóját (`topic_id`), a felhasználónevet (`username`) és a kommentet (`comment`). Valamint azt is ellenőrzi, hogy ezek nem üresek.
3. `$topic_id = $_POST['topic_id'];`: Az új kommenthez kapcsolódó téma azonosítóját elmentjük a `$topic_id` változóba.
4. `$username = $_POST['username'];`: Az új kommentet létrehozó felhasználó nevét elmentjük a `$username` változóba.
5. `$comment = $_POST['comment'];`: Az új komment szövegét elmentjük a `$comment` változóba.
6. `$check_query = "SELECT * FROM topics WHERE id = $topic_id";`: Ellenőrizzük, hogy az adott `topic_id`-vel rendelkező téma létezik-e az adatbázisban.
7. `if(mysqli_num_rows($check_result) > 0)` { Ellenőrizzük, hogy a lekérdezés eredménye nem üres-e, vagyis létezik-e ilyen azonosítójú téma.
8. `$insert_query = "INSERT INTO comments (topic_id, username, comment) VALUES ($topic_id, '$username', '$comment')";`: Ha a téma létezik, akkor létrehozunk egy SQL beszúrási lekérdezést az új komment hozzáadásához a `comments` táblához.
9. `mysqli_query($conn, $insert_query);`: Végrehajtjuk az SQL beszúrási lekérdezést az adatbázison, így felvisszük az új kommentet.
10. `mysqli_close($conn);`: Az adatbázis kapcsolat bezárása a kód végén.

1.6.44 get_completed_dates.php

A `get_completed_dates.php` felelős a felhasználó által befejezett edzések dátumainak lekérdezéséért az adatbázisból, majd ezek JSON formátumban való visszaadásáért.

Működése

1. `session_start();`: Ez a függvény elindítja a munkamenetet, ami lehetővé teszi az adatok tárolását és hozzáférését a különböző oldalak között.
2. `include ("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (`db.php`), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
3. `if ($_SERVER["REQUEST_METHOD"] == "GET")` {: Ez a feltétel ellenőrzi, hogy a kérés a GET metódust használja-e.
4. `if (!isset($_SESSION['username']))` {: Ellenőrizzük, hogy a felhasználó be van-e jelentkezve a munkamenetben.
5. `http_response_code(403); // Hozzáférés megtagadva`: Ha a felhasználó nincs bejelentkezve, akkor 403-as hibakódot küldünk vissza, és kilépünk a programból a "Unauthorized access" üzenettel.
6. `$username = $_SESSION['username'];`: Az éppen bejelentkezett felhasználó nevét elmentjük a `$username` változóba.
7. `$user_query = "SELECT id FROM user WHERE username = '$username'";`: SQL lekérdezés az adott felhasználó azonosítójának lekérésére a felhasználónév alapján.
8. `$user_result = $conn->query($user_query);`: A lekérdezést végrehajtjuk az adatbázison.
9. `if ($user_result->num_rows > 0)` {: Ellenőrizzük, hogy a lekérdezés eredménye nem üres-e.
10. `$row = $user_result->fetch_assoc();`: Az eredményt egy asszociatív tömbbe mentjük.
11. `$userId = $row['id'];`: Az adott felhasználó azonosítóját elmentjük a `$userId` változóba.
12. `$completed_dates_query = "SELECT completion_date FROM workout_completed WHERE user_id = '$userId'";`: SQL lekérdezés előkészítése a befejezett edzések dátumainak lekérdezésére a `workout_completed` táblából az adott felhasználó azonosítója alapján.
13. `$completed_dates_result = $conn->query($completed_dates_query);`: A lekérdezést végrehajtjuk az adatbázison.
14. `$completed_dates = [];`: Ebben a tömbben fogjuk tárolni a befejezett edzések dátumait.
15. `if ($completed_dates_result->num_rows > 0)` {: Ellenőrizzük, hogy a lekérdezés eredménye nem üres-e.
16. Az eredményből kiolvassuk a befejezett edzések dátumait és elmentjük őket a `$completed_dates` tömbbe.
17. `header('Content-Type: application/json');`: Beállítjuk a válasz fejlécét, hogy JSON adatokat küldünk vissza.
18. `echo json_encode($completed_dates);`: Elküldjük a befejezett edzések dátumait JSON formátumban.
19. `http_response_code(404); // Nem található a felhasználó`: Ha a felhasználó nem található az adatbázisban, akkor 404-es hibakódot küldünk vissza, és kilépünk a programból a "User not found" üzenettel.

20. `http_response_code(405);` // **Nem engedélyezett kérésí módszer**: Ha a kérés nem GET metódussal érkezett, akkor 405-ös hibakódot küldünk vissza, és kilépünk a programból a "Method Not Allowed" üzenettel.

1.6.45 `get_exercise_details.php`

A `get_exercises_details.php` fájl felelős a gyakorlat részleteinek lekérdezéséért az adatbázisból egy adott azonosító alapján, majd ezeket JSON formátumban küldi vissza a kliensoldalnak

Működése

1. `include ("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (`db.php`), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
2. `$id = $_GET['id'];`: Azonosító lekérése a GET kérésből, amelyet a JavaScript küldött át.
3. `$sql = "SELECT nameOfExercise, shortdescriptionOfExercise, imageOfExercise FROM workout_exercises WHERE id = $id";`: SQL lekérdezés előkészítése a gyakorlat részleteinek lekérdezésére az adott azonosító alapján.
4. `$result = $conn->query($sql);`: A lekérdezést végrehajtjuk az adatbázison.
5. `if ($result->num_rows > 0) {`: Ellenőrizzük, hogy a lekérdezés eredménye nem üres-e.
6. Az eredményből kiolvassuk a gyakorlat részleteit és elmentjük azokat egy asszociatív tömbbe.
7. Az asszociatív tömb tartalmát JSON formátumba alakítjuk.
8. `echo json_encode($exerciseDetails);`: Elküldjük a gyakorlat részleteit JSON formátumban a kliensoldalnak.
9. `else { echo "Exercise not found"; }`: Ha a lekérdezés eredménye üres, akkor kiírjuk a "Gyakorlat nem található" üzenetet.
10. `$conn->close();`: Az adatbázis kapcsolat bezárása a kód végén.

1.6.46 get_exercise.php

A `get_exercise.php` felelős az edzések lekérdezéséért az adatbázisból egy adott dátum alapján, majd ezeket JSON formátumban küldi vissza a kliensoldalnak.

Működése

1. `session_start();`: Ez a függvény elindítja a munkamenetet, ami lehetővé teszi az adatok tárolását és hozzáférését a különböző oldalak között.
2. `include ("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (`db.php`), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
3. `if (isset($_SESSION['username']))` { Ellenőrizzük, hogy van-e bejelentkezett felhasználó a munkamenetben.
4. `$username = $_SESSION['username'];`: Az éppen bejelentkezett felhasználó nevét elmentjük a `$username` változóba.
5. `$sql = "SELECT generated_workout FROM user WHERE username = '$username'";`: SQL lekérdezés előkészítése a generált edzések lekérdezésére az adott felhasználó alapján.
6. `$result = mysqli_query($conn, $sql);`: A lekérdezést végrehajtjuk az adatbázison.
7. `if (mysqli_num_rows($result) > 0)` { Ellenőrizzük, hogy a lekérdezés eredménye nem üres-e.
8. Az eredményből kiolvassuk a generált edzéseket és elmentjük azokat egy változóba.
9. `if (isset($_GET['date']))` { Ellenőrizzük, hogy a GET paraméterben van-e dátum, amit átadtak.
10. `$date = $_GET['date'];`: Az átadott dátumot elmentjük a `$date` változóba.
11. A generált edzések között keressük meg a megfelelő dátumot és elkészítjük a választ, csak az azonosítókat küldjük vissza az upper body és lower body gyakorlatokból.
12. `echo json_encode($response);`: Elküldjük a választ JSON formátumban a kliensoldalnak.
13. `exit();`: Kilépünk a programból a ciklusból, ha megtaláltuk a keresett dátumot.
14. Ha a dátum nem található a generált edzések között, akkor hibát jelzünk.
15. Ha a dátum paraméter nincs átadva, akkor szintén hibát jelzünk.
16. `else { echo json_encode(array("error" => "User not found in database")); }`: Ha a felhasználó nem található az adatbázisban, akkor hibát jelzünk.
17. `else { echo json_encode(array("error" => "User not logged in")); }`: Ha nincs bejelentkezett felhasználó, akkor szintén hibát jelzünk.
18. `mysqli_close($conn);`: Az adatbázis kapcsolat bezárása a kód végén.

1.6.47 regenerate_workout.php

A `regenerate_workout.php` fájl felelős az edzések újra generálásáért a felhasználói beállítások alapján.

Működése

1. `session_start();`: Ez a függvény elindítja a munkamenetet, ami lehetővé teszi az adatok tárolását és hozzáférését a különböző oldalak között.
2. `include ("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (`db.php`), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
3. `include ("functions.php");`: Ez a sor importálja a `functions.php` fájlt, amely tartalmazza a szükséges függvényeket, például a `generateWorkouts` függvényt.
4. `if (isset($_POST['regenerate'])) {`: Ellenőrizzük, hogy a POST kérés elküldte-e a "regenerate" adatot, ami jelzi az újragenerálás szándékát.
5. `$workouttime = $_POST['workouttime'];`: Az új edzésidő lekérése a POST kérésből.
6. `$username = $_SESSION['username'];`: Az éppen bejelentkezett felhasználó nevének lekérése a munkamenetből.
7. Felhasználó azonosítójának lekérése a felhasználónév alapján, majd a kívánt időtartam beállítása az adatbázisban.
8. Felhasználó szándékának lekérése az adatbázisból a felhasználó azonosítója alapján.
9. Ha a szándék sikeresen le lett kérdezve az adatbázisból, akkor az új edzések generálása a `generateWorkouts` függvénnyel a felhasználó szándékának és az új edzésidőnek megfelelően.
10. Az edzések újra generálása után az eddig teljesített edzések törlése az adatbázisból.
11. Végül a felhasználó vezérlőpultjára való átirányítás és a kód leállítása.

1.6.48 save_completed_workout.php

A `save_completed_workout.php` PHP fájl felelős az edzés teljesítésének elmentéséért az adatbázisban

Működése

1. `session_start();`: Ez a függvény elindítja a munkamenetet, ami lehetővé teszi az adatok tárolását és hozzáférését a különböző oldalak között.
2. `include ("db.php");`: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (`db.php`), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
3. Az adatok fogadása a POST kéréstől. Az éppen bejelentkezett felhasználó nevét és az edzés teljesítésének dátumát POST kérésből lekérjük.
4. Felhasználó azonosítójának lekérése a felhasználónév alapján az adatbázisból.
5. Ellenőrizzük, hogy az adott felhasználóhoz már hozzá lett-e rendelve ez a dátum az edzések között.
6. Ha még nem volt edzés ezen a dátumon az adott felhasználóhoz, akkor az adatokat elmentjük az adatbázisba.

7. Ha már volt edzés ezen a dátumon az adott felhasználóhoz, akkor hibaüzenetet küldünk vissza.
8. Ha a felhasználó nem található az adatbázisban, akkor szintén hibaüzenetet küldünk vissza.
9. Az adatbázis kapcsolat bezárása a kód végén.

1.6.49 search.php

A **search.php** felelős a felhasználók kereséséért az adatbázisban a megadott keresési kifejezés alapján.

Működése

1. **include("db.php");**: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (**db.php**), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
2. A keresési kifejezés lekérdezése a GET kérésből. A keresési kifejezést a "search" paraméter tartalmazza a GET kérésben. Ha nincs keresési kifejezés megadva, akkor üres string lesz az alapértelmezett érték.
3. Felhasználók lekérése az adatbázisból a keresési kifejezés alapján. A **LIKE** operátorral keresünk a felhasználók felhasználónevében olyan szavakat, amelyek tartalmazzák a keresési kifejezést.
4. Az eredmények formázása asszociatív tömb formátumba, hogy könnyen kezelhetőek legyenek.
5. A keresési eredmények JSON formátumba való átalakítása és kimenetként való kiírása.
6. Az adatbázis kapcsolat bezárása a kód végén, hogy ne foglalja a kapcsolatot feleslegesen.

1.6.50 update_workout_consent.php

Az **update_workout_consent.php** felelős az edzési beleegyezés frissítéséért az adatbázisban.

Működése

1. **include ("db.php");**: Ez a sor importálja az adatbázis kapcsolódáshoz szükséges fájlt (**db.php**), amely tartalmazza az adatbázis kapcsolódáshoz szükséges információkat és az adatbázis kapcsolat létrehozását.
2. Az első lépés az, hogy ellenőrizzük, hogy a kérés POST metódussal érkezett-e. Ha nem, akkor a kód nem fut tovább, és hibát ad vissza.
3. Ellenőrizzük, hogy a felhasználó be van-e jelentkezve. Ha nincs, akkor a kód hibát ad vissza (403 Forbidden) és kilép.
4. Felhasználó azonosító lekérése felhasználónév alapján az adatbázisból.
5. Ellenőrizzük, hogy az adott felhasználó rendelkezik-e már elfogadott edzési beleegyezéssel (ha a **workout_consent** értéke 1). Ha igen, akkor a kód hibát ad vissza (400 Bad Request) és kilép.
6. Ha a felhasználónak még nincs elfogadott edzési beleegyezése, akkor frissítjük a **workout_consent** értékét 1-re az adatbázisban.

7. Ha a frissítés sikeres, akkor a kód visszaad egy státuszkódot (200 OK) és egy üzenetet, hogy sikeresen frissült az edzési beleegyezés.
8. Ha bármilyen hiba történik az adatbázis műveletek során, akkor a kód hibát ad vissza (500 Internal Server Error) és kiírja az adatbázis hibaüzenetét.
9. Ha a felhasználó nem található az adatbázisban, akkor a kód hibát ad vissza (404 Not Found) és egy üzenetet, hogy a felhasználó nem található.
10. Ha a kérés nem POST metódussal érkezett, akkor a kód hibát ad vissza (405 Method Not Allowed).
11. Az adatbázis kapcsolat bezárása a kód végén, hogy ne foglalja a kapcsolatot feleslegesen.

1.7 Tesztelés, tesztesetek

A tesztelés szintjei a következők:

1. **Unit teszt:** a legkisebb egységek, modulok, objektumosztályok, alrendszerek tesztelése.
2. **Rendszer teszt:** az elkészült rendszer tesztelése annak érdekében, hogy igazoljuk, hogy megfelel a követelményeknek.
3. **Elfogadási teszt:** a megrendelő, illetve a felhasználó által megfogalmazott követelmények teljesülését vizsgáljuk. Ez a tesztelési fázis már általában a megrendelő bevonásával történik és dönti el, hogy a rendszer elfogadható vagy sem.
4. **Installációs teszt:** az elfogadás után a rendszert a végső működési környezetbe integráljuk és ott vizsgáljuk a helyes működést.

Ezek a tesztelési módszerek lettek elvégezve a programon. Az alábbiakban néhány fontosabb komponens tesztjét mutatjuk be.

1.7.1 Regisztráció tesztelés

Regisztráljunk egy példa felhasználót „TesztElek” névvel, „tesztElek123” jelszóval, „tesztelek1@mail.com” e-mail címmel.

A tesztesetek a következők:

1. **Eset:** Minden mező kitöltetlen marad

Várható kimenet: Egy figyelmeztető üzenet jelenik meg, ami közli, hogy minden mezőt ki kell tölteni.

2. **Eset:** A jelszó túl rövid (kevesebb, mint 8 karakter)

Várható kimenet: Egy figyelmeztető üzenet jelenik meg, ami közli, hogy a jelszó túl rövid.

3. **Eset:** A megadott felhasználónév vagy e-mail már regisztrálva van

Várható kimenet: Egy figyelmeztető üzenet jelenik meg, ami közli, hogy a megadott felhasználónév vagy e-mail cím már regisztrálva van.

4. **Eset:** A két jelszó nem egyezik meg

Várható kimenet: Egy figyelmeztető üzenet jelenik meg, ami közli, hogy a két megadott jelszó nem egyezik meg.

5. **Eset:** Sikeres regisztráció és e-mail küldés

Várható kimenet: A regisztráció sikeres, és az e-mail elküldésre kerül. A felhasználó átirányításra kerül az "introducing.php" oldalra.

1.7.2 Bejelentkezés tesztelés

A bejelentkezésnél használjuk az előzőleg beregisztrált felhasználói adatokat.

A tesztesetek a következők:

1. **Eset:** A felhasználó már bejelentkezett, de nem töltötte ki az „Introducing” oldalt

Várható kimenetel: A felhasználó átirányításra kerül az „Introducing.php” oldalra.

2. **Eset:** A felhasználó már bejelentkezett és kitöltötte az „Introducing” oldalt

Várható kimenetel: A felhasználó átirányításra kerül a „Menu.php” oldalra.

3. **Eset:** A felhasználó még nem jelentkezett be és megpróbálja elküldeni az űrlapot üresen

Várható kimenetel: Az oldal nem engedi az üres űrlap elküldését és figyelmezteti a felhasználót, hogy minden mezőt ki kell tölteni.

4. **Eset:** A felhasználó még nem jelentkezett be, de megpróbálja elküldeni a kitöltött űrlapot rossz jelszóval

Várható kimenetel: Egy figyelmeztető üzenet jelenik meg, ami közli, hogy a megadott jelszó hibás.

5. **Eset:** A felhasználó még nem jelentkezett be, de megpróbálja elküldeni az űrlapot kitöltve, de nem létező e-mail címmel

Várható kimenetel: Egy figyelmeztető üzenet jelenik meg, ami közli, hogy nincs ilyen regisztrált felhasználó.

6. **Eset:** A felhasználó sikeresen bejelentkezik

Várható kimenetel: A felhasználó átirányításra kerül az „Introducing.php” vagy a „Menu.php” oldalra attól függően, hogy kitöltötte-e már a felhasználó az „Introducing” oldalt.

1.7.3 Felhasználói adatváltoztatás tesztelés

A tesztesetek a következők:

1. **Eset:** A felhasználó nincs bejelentkezve, de próbálja elérni ezt az oldalt

Várható kimenetel: A felhasználó átirányításra kerül a „Landing.php” oldalra, ahol be tud jelentkezni.

2. **Eset:** A felhasználó be van jelentkezve és sikeresen megnyitja a beállítások oldalt

Várható kimenetel: Az oldal megjeleníti a felhasználó adatait az űrlapban és lehetővé teszi azok módosítását.

3. **Eset:** A felhasználó módosítja az adatait az űrlapon és frissíti azokat

Várható kimenetel: Az adatok sikeresen frissülnek az adatbázisban és az űrlap megfelelően frissíti a felhasználó által megadott adatokat.

4. **Eset:** A felhasználó feltölt egy profilképet

Várható kimenetel: A felhasználó által feltöltött kép sikeresen mentve lesz az adatbázisban és a megadott mappában.

5. **Eset:** A felhasználó újra generálja az edzéstervet

Várható kimenetel: A felhasználó új generált edzéstervet kap, mely sikeresen frissítve lesz az adatbázisban.

6. **Eset:** A felhasználó új jelszót kér

Várható kimenetel: A felhasználó sikeresen le lesz generálva és ki lesz küldve a felhasználó e-mail címére.

7. **Eset:** A felhasználó üresen hagyja az űrlapon az adatokat majd megpróbálja frissíteni azt.

Várható kimenetel: Az oldal figyelmezteti a felhasználót, hogy ki kell tölteni az űrlap mezőit.

1.7.4 Feladat feltöltés tesztelés

Töltsünk fel az edzésterv előállításához szükséges feladatot.

A tesztesetek a következők:

1. **Eset:** A felhasználó megpróbál feltölteni képet a feladathoz, de a kép fájl mérete túl nagy

Várható kimenetel: Az űrlap hibát jelez és figyelmezteti a felhasználót arra, hogy túl nagy.

2. **Eset:** A felhasználó megpróbál egy olyan fájlt feltölteni, mely nem megengedett típusú

Várható kimenetel: Az űrlap hibát jelez és figyelmezteti a felhasználót arra, hogy a fájl kiterjesztése nem megfelelő.

3. **Eset:** A felhasználó feltölt egy képet és teljesen kitölti hozzá az űrlapot

Várható kimenetel: A kép és az űrlap adatai sikeresen be lesznek szűrve az adatbázisba.

4. **Eset:** A felhasználó feltölt egy képet, de nem tölti ki hozzá teljesen az űrlapot

Várható kimenetel: Az űrlap hibát jelez és figyelmezteti a felhasználót arra, hogy minden mezőt ki kell tölteni.

5. **Eset:** Az űrlapon technikai hiba történik a fájl feltöltése közben

Várható kimenetel: Az űrlap hibát jelez és figyelmezteti a felhasználót arra, hogy technikai hiba történt.

6. **Eset:** A felhasználó nem POST kéréssel próbálja megnyitni az oldalt

Várható kimenetel: Az oldal visszairányítja a felhasználót az adminisztrációs oldalra.

1.7.5 Személyre szabott feladatsor generálás tesztelés

A felhasználó sikeresen kitölti az „Introducing” oldalt és kiválasztja, hogy mennyi időre generálja le a feladatsort.

A tesztesetek a következők:

1. **Eset:** A függvényt meghívják a szükséges paraméterekkel és adatbázisba történő beszúrás sikeres

Várható kimenetel: A felhasználónak szánt edzésterv sikeresen bekerül az adatbázisba és nincs hibaüzenet.

2. **Eset:** A függvényt meghívják, de érvénytelen paraméterekkel

Várható kimenetel: A függvény hibát jelez és visszatér az üresen generált edzéstervvel.

3. **Eset:** Az SQL lekérdezések során hiba történik

Várható kimenetel: A függvény hibát jelez és visszatér az üresen generált edzéstervvel.

1.7.6 Feladatsor lejátszásához szükséges modul tesztelés

A tesztesetek a következők:

1. **Eset:** Az oldal betöltésekor a felhasználónak megjeleníti az oldal az „Attention” feliratot és a beleegyezés gombot

Várható kimenetel: A felhasználó a gombra kattint és frissül az oldal és megjeleníti a felhasználó személyre szabott edzéstervét. Ha hiba történik, akkor az oldal a konzolon keresztül értesít.

2. **Eset:** A hetek gombjai megjelennek

Várható kimenetel: A felhasználó, ha rákattint valamelyik hét gombjára, akkor megjelenik a héthez tartozó napoknak a gombjai.

3. **Eset:** A napok gombjai megjelennek

Várható kimenetel: A felhasználó, ha rákattint valamelyik nap gombjára, akkor az ahhoz a naphoz tartozó feladatsorral megjelenik a modális ablak.

4. **Eset:** A felhasználó végig lepkéd minden gyakorlaton az adott napon

Várható kimenetel: Az utolsó gyakorlat végrehajtása után a „Next” gomb „End workout” gombra változik és a felhasználónak lehetősége van befejezni az edzést. Ezután a nap gombja zöldre változik jelezve, hogy az aznapi edzés kész. Ha nem tudja elmenteni az oldal azt, hogy az aznapi edzés készen van, akkor az oldal a konzolon keresztül értesít. Egy felugró ablak értesíti a felhasználót, hogy sikeresen végig csinálta az edzést.

1.8 Továbbfejlesztési lehetőségek

1.8.1 A weboldal továbbfejlesztése

A weboldal egyik legfontosabb felhasználói felülete a fórum oldalán a későbbiekben szeretnénk megcsinálni, hogy adminisztrátor aktívan moderálhassa a felhasználói hozzászólásokat jelentések alapján. A felhasználói oldalon létrehoznánk egy gombot, mellyel egy adott topic-ot vagy kommentet lehetne jelenteni.

A weboldal azon fő funkciója mellett, hogy edzéstervet generál szeretnénk a későbbiekben egy egyéni étrend generátort, mely a felhasználó által megadott tulajdonságok szerint generál egyéni étrendet.

Szeretnénk a későbbiekben biztosítani a felhasználónak azt, hogy nyomon tudja követni a haladását, erre egy új felületet hoznánk létre, melyen a felhasználó megadhatná hétről hétre például a testtömeg változását és ez alapján hoznánk létre egy statisztikát számára.

Jelenleg az elfelejtett jelszó és a regisztrációs üzenet kikérése PHPMailerrel történik és a Mailtrap.io szolgáltatása kapja el a kiküldött e-maileket. A későbbiekben szeretnénk, hogy saját működő domain címünkről közvetlenül a felhasználóhoz jusson el az üzenet.

A későbbiekben szeretnénk, hogy a felhasználóknak legyen egy kihívás rendszer, mely arról szól például, hogy adott feladatok elvégzése után vagy akár huzamosabb alkalmazás használat után kitűzőket kapjanak, mely a profil beállításaiuknál lehetne megtekinthető. Ez egy presztízs rendszert hozna létre.

Jelenleg az oldalon szemléltetés céljából más képei vannak felhasználva. Szeretnénk, hogy a későbbiekben az oldalon saját magunk által készített képek legyen szemléltetésre használva.

2 Felhasználói dokumentáció

2.1 Alkalmazás ismertetése

A FitNet webalkalmazás egy online platform, amely lehetővé teszi a felhasználók számára, hogy egészséges életmódot folytassanak és támogassák egészségüket és fitnessüket. Az alkalmazás különböző funkciókat kínál, beleértve az edzésterv készítését, a közösségi támogatást és a motivációs tartalmakat. A felhasználók széles körben használhatják az alkalmazást az egészséges életmód kialakításához és fenntartásához. Lehetőségük van személyre szabott edzésterv készítésére az adott céljaiknak megfelelően, például fogyás, izomépítés vagy állóképesség növelése céljából. A közösségi funkcióknak köszönhetően a felhasználók inspirálhatják egymást, megoszthatják tapasztalataikat és motiválhatják egymást az elkötelezettség fenntartásában.

2.2 Rendszerkövetelmények

- **Hardverkövetelmények:**
 - Operációs rendszer: Az alkalmazás támogatja a legnépszerűbb operációs rendszereket, mint például Windows, macOS és Linux.
 - Processzor: Legalább egy közepes vagy magas teljesítményű processzor, amely képes futtatni a böngészőt és a webalkalmazást.
 - Memória: Ajánlott legalább 4 GB RAM az optimális teljesítmény érdekében.
- **Szoftverkövetelmények:**
 - Memória: Ajánlott legalább 4 GB RAM az optimális teljesítmény érdekében.
 - Internetkapcsolat: Stabil internetkapcsolat szükséges az alkalmazás használatához és a frissítések letöltéséhez.
- **Egyéb követelmények:**
 - Felhasználói fiók létrehozása: A felhasználóknak szükségük van egy regisztrált fiókra az alkalmazás használatához és a személyre szabott funkciókhoz való hozzáféréshez.
 - Bejelentkezés: A felhasználóknak be kell jelentkezniük a fiókjukba az alkalmazás használatához és az egyedi beállítások eléréséhez.

2.3 Szükséges beállítások

- **Fiók létrehozása vagy bejelentkezés:**
 - A index oldalra érve azonnal észrevehető a jobb felső sarokban található bejelentkezési gomb, amelynek segítségével könnyedén beléphetünk vagy regisztrálhatunk.
 - Ha a bejelentkezést választjuk, akkor lehetőségünk nyílik személyre szabott edzéstervet használni, valamint kommunikálni más felhasználókkal a platformon belül.
 - Amennyiben regisztrálunk, akkor az szükséges adatokat kell megadjuk, mint például a felhasználónév, e-mail cím és jelszó. Miután regisztráltunk és beléptünk, egy egyszeri megjelenő személyes beállítások ablak jelenik meg, amely segítségével testre szabhatjuk az edzéstervünket a későbbiekben.
- **Általános beállítások:**
 - A felhasználók teljeskörűen testre szabhatják személyes adataikat, mint például az e-mail címüket, telefonszámukat és nevüket. Emellett lehetőségük van megadni az életkorukat, testsúlyukat és magasságukat, valamint kiválaszthatják a kitűzött célt, amelyet az edzéssel el szeretnének érni. Továbbá előre is beállíthatják, hogy hány hónapra szeretnének előre tervezni az edzéseiket.

2.4 Alkalmazás használata

Az oldalunk főoldala a landing.php, ami a FitNet alkalmazás főoldala. Felül egy hárommenüponos menü található: "Home", "About us" és "Exercises". Ha lefelé görgetsz, eljutsz a "Rólunk" szekcióhoz, majd a lábléchez, ahol újra elérhetővé válnak a "Home", "About" és a "Team" oldalra mutató link.

FitNet

Fitness is

Start working out with us right away!

[Learn more](#)



Why is it so great?

Welcome to our FitNet website! Here, you'll find everything you need to kickstart your fitness journey and achieve your health goals. From personalized workout plans to expert nutrition advice, we've got you covered. Browse through a variety of exercise routines tailored to your fitness level and preferences, whether you're into cardio, strength training. Our platform also offers insightful articles, tips, and tutorials to help you stay motivated and informed every step of the way. Join our vibrant community of fitness enthusiasts, track your progress, and start transforming your lifestyle today with FitNet!

100%

User friendly

6

users

Over 100

exercises to choose
from



[Home](#) [About](#) [Team](#)

©2024 FitNet | All Rights Reserved

A lábléc Home menüpontját választva az oldal tetejére dobja, az About kiválasztása esetén az oldal about us részéhez navigálok a felhasználó. Ha a Team oldalt választjuk akkor az oldal alkotókról tudhatunk meg többet a team oldalon.

[Home](#) [About us](#) [Exercises](#)

Fajta Jordán


Hello, my name is Jordan Fajta, a frontend developer focused on crafting captivating user experiences and polished interfaces. With expertise in user interface design, HTML/CSS/JavaScript development, responsiveness, interactivity, testing, and design implementation, I bring designs to life with clean, efficient code. I thrive on effective team collaboration and actively curate content to enrich projects. My goal is to exceed expectations and create memorable digital experiences.


Frontend

Tailwind css
Html
JavaScript
Scss

Contact

+36202422224
jordanfajta@gmail.com





Mártai Balázs

Greetings! My name is Balázs Márta, a backend developer dedicated to building robust applications and optimizing performance. My expertise lies in translating complex requirements into efficient application logic, managing databases, implementing secure user authentication, and optimizing performance. With a focus on reliability and collaboration, I strive to deliver backend solutions that elevate digital experiences.

Backend Database Contact

Php

MySql

+36301234567
bmartai05@gmail

93

Ha a vízszintes menüben a "Exercises" opciót választod, átirányít a musclegroup oldalra, ahol anatómiai test segítségével válogathatsz az izomcsoportok között. Kiválasztva egyet, kilistázzuk az izomcsoport gyakorlatait animált GIF-képekkel, leírásokkal.

[Home](#) [About us](#) [Exercises](#)

Muscle Group Selector

ARMS

Biceps
Deltoids
Forearms
Triceps

BACK

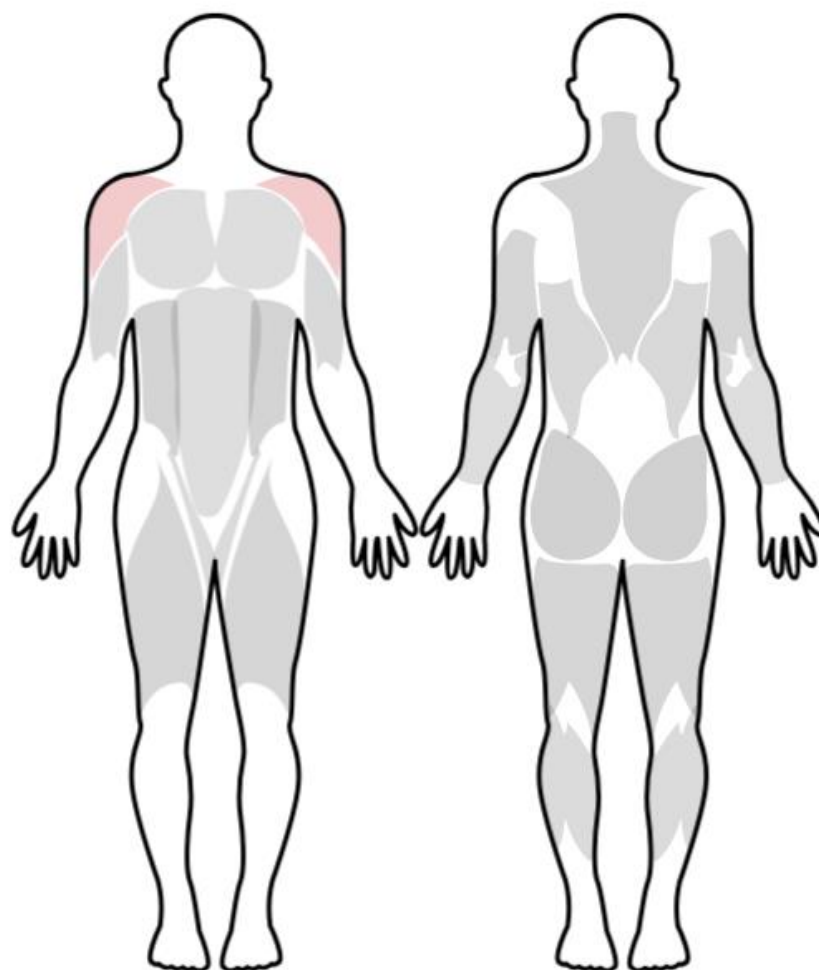
Trapezius
Lats

CORE

Abs
Obliques
Pectorals

LEGS

Adductors
Calves
Hamstrings
Glutes
Quads



Quads exercises



Leg extension

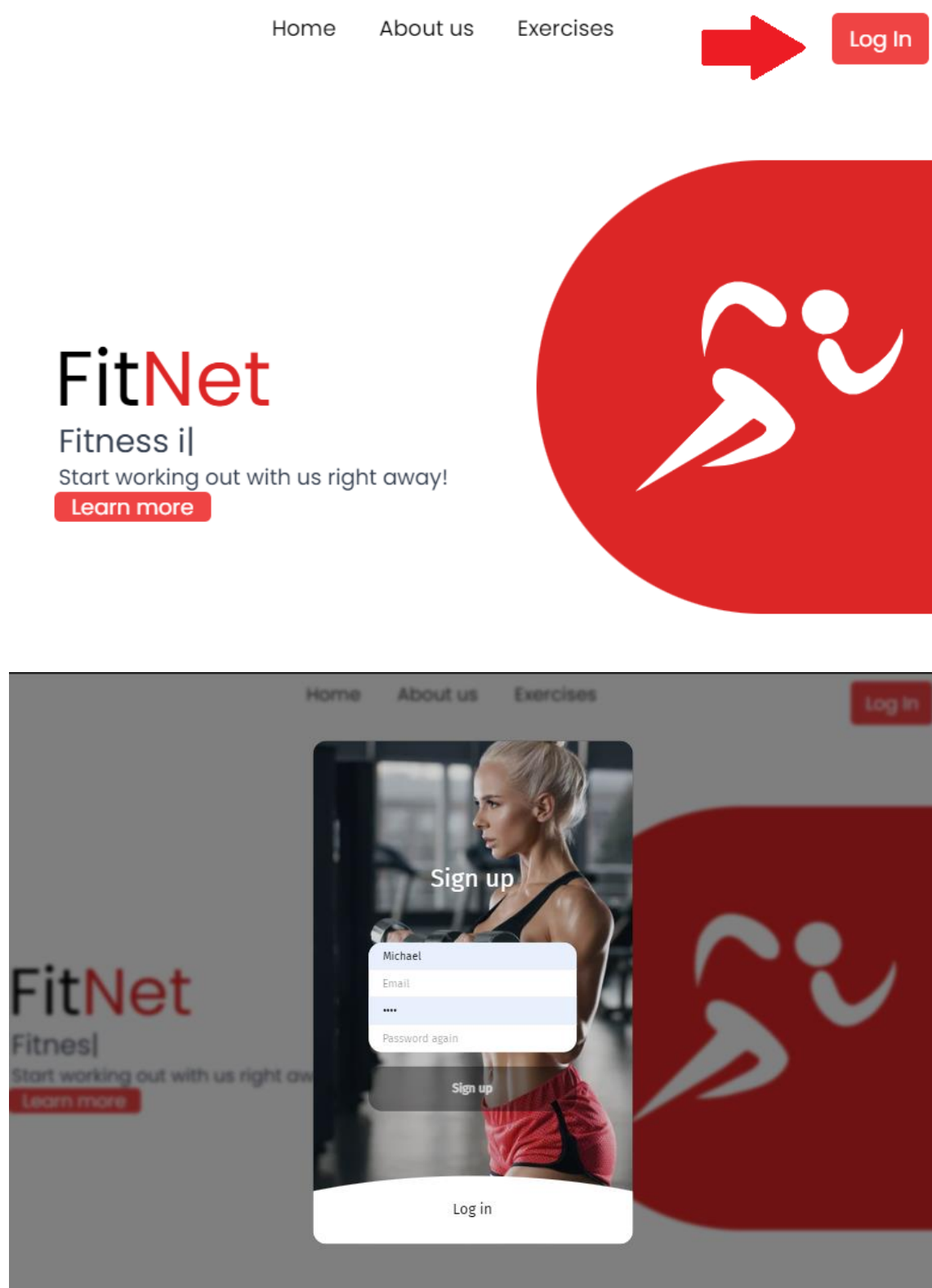
Leg extensions are a popular isolation exercise that primarily target the quadriceps muscles, which are located at the front of the thigh. Leg extension exercises typically involve the use of a leg extension machine found in most gyms.



Barbell split squat

The Barbell Split Squat is a dynamic lower body exercise that targets multiple muscle groups simultaneously, including the quadriceps, hamstrings, glutes, and calves. It involves placing a barbell across the shoulders while standing in a split stance, with one foot positioned forward and the other foot positioned back on an elevated surface. As you lower your body into a squat position, the forward knee bends while the back knee descends towards the ground. This exercise improves lower body strength, stability, and balance, making it a valuable addition to any strength training routine.

A főoldalon a jobb felső sarokban található egy bejelentkezés gomb. Kattintásra egy ablak jelenik meg, ahol bejelentkezhetsz, vagy regisztrálhatsz.



Az új regisztrációhoz egy egyszerű űrlapot kell kitöltened, majd átirányítunk a dashboard oldalra, ahol elérheted az edzésprogramokat. Itt láthatod a heteket (pl. week1), amire kattintva az edzésnapok megjelennek. Az edzésnapok közötti pihenőnapok külön jelölve vannak (rest day), melyekre nem tudsz kattintani.

Welcome to the introduction section!

For the best experience, we require you to upload your information about yourself.

Next

Page 1

Surname
Sergio

First name
Ramos

Weight
90 kg

Height
193 cm

Previous

Next

Page 2

Address
Madrid

Phone
+36701234567

Set your birthday:
1986. 03. 30.

Previous

Next

Page 3

Choose your gender:
☒ Man
☐ Women
☐ Etc.

Set your intention:
Weight Maintenance

Previous

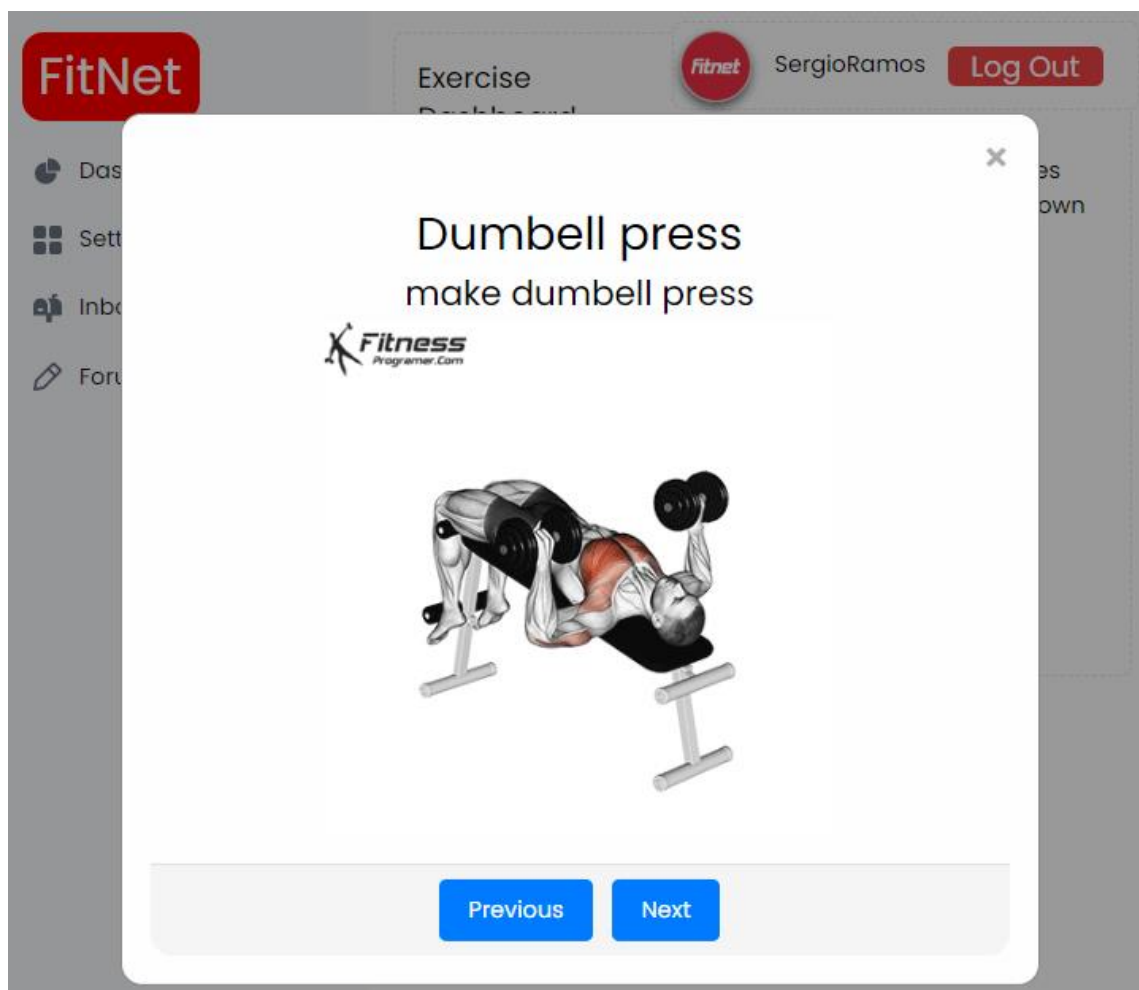
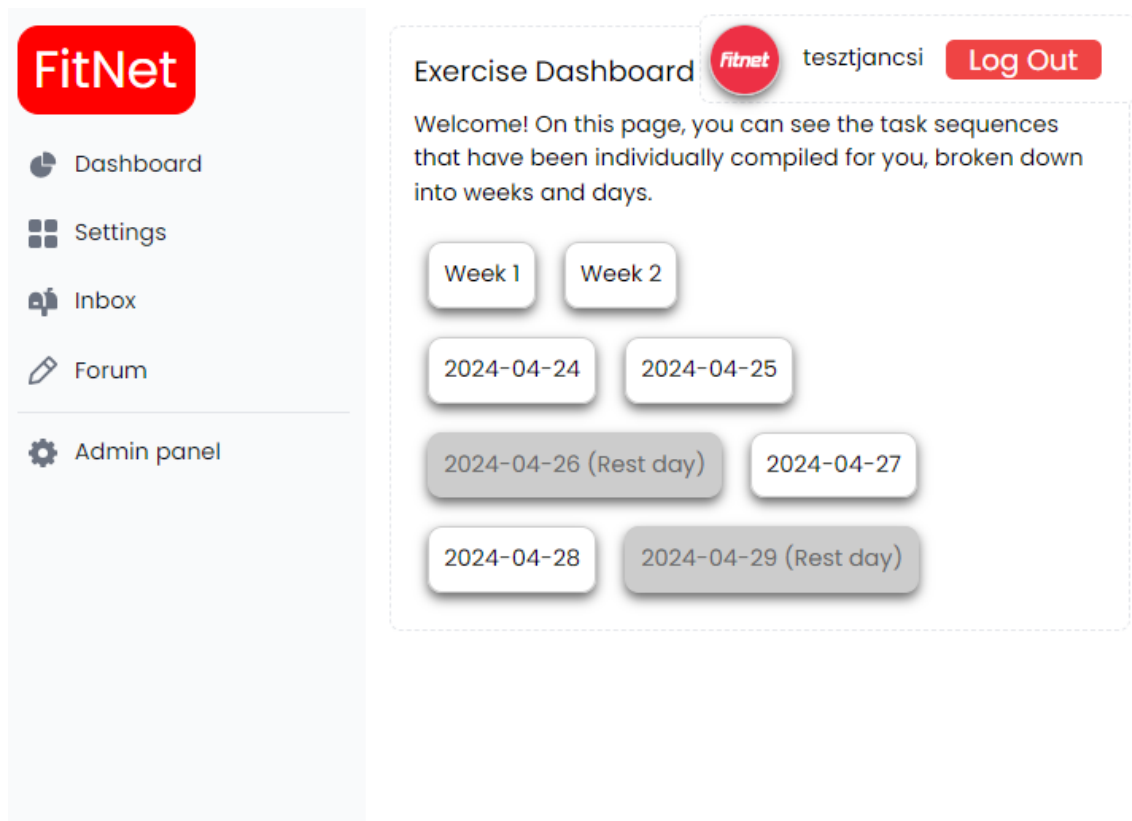
Next

Page 4

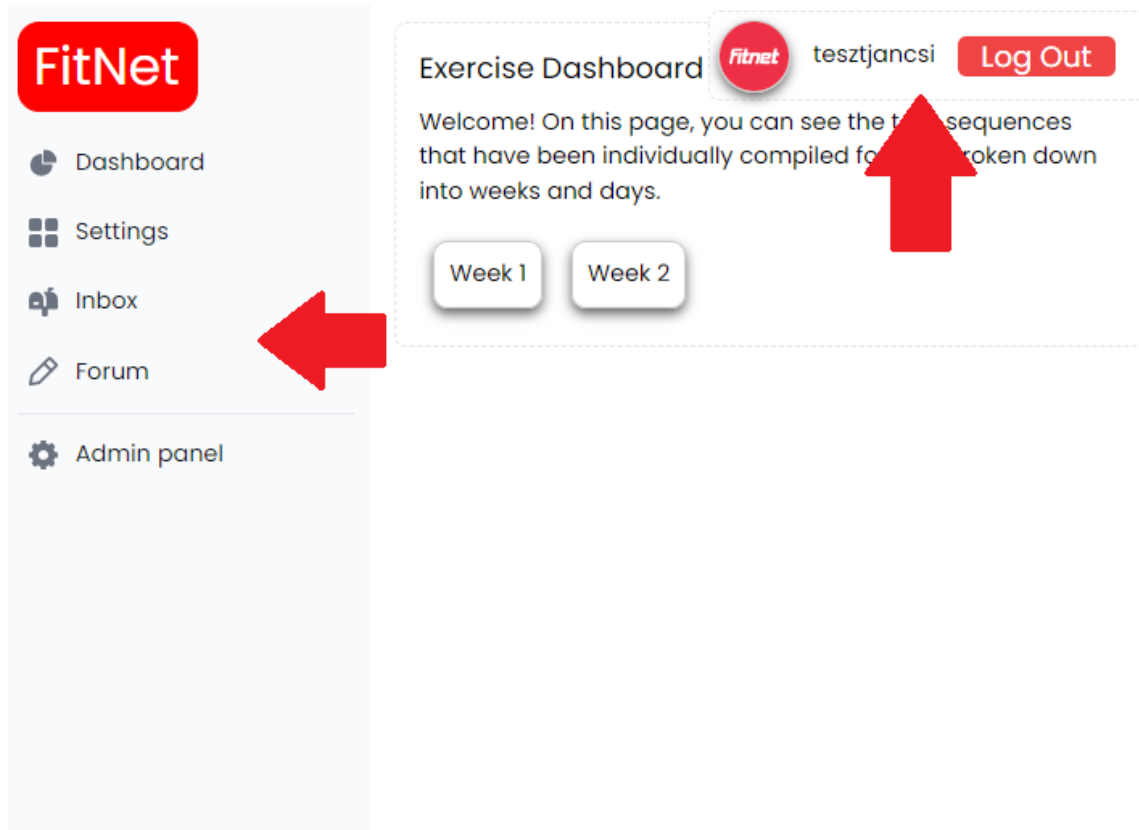
For how much week would you like to generate the workout plan?
4

Previous

Submit




A baloldalon található a horizontális menü, ahol megtalálod a "Dashboard", "Settings", "Inbox", "Forum" lehetőségeket, valamint az admin jogokkal rendelkezők számára az "Admin panelt". A jobb oldalon láthatod a feltöltött profilképed (ha van), a felhasználóneved és a kijelentkezés gombot, melyek csak akkor láthatóak, ha be vagy jelentkezve.




A "Settings" menüpont alatt szerkesztheted az adataidat, generálhatsz új edzésprogramot vagy feltöltheted a profilképedet, megváltoztathatod a jelszavad.

Surname:

First name:


Weight:
 kg

Height:
 cm

Choose your gender:

☒ Man
☐ Woman
☐ Etc.

Change your phone number:
+

Change your birthday:
2000. 01. 21. 

Set your intention:
 ▼

Upload your avatar: (Only JPG, JPEG, PNG)

Fájl kiválasztása

Nincs fájl kiválasztva

Upload

Regenerate workout:

For how much week would you like to generate?

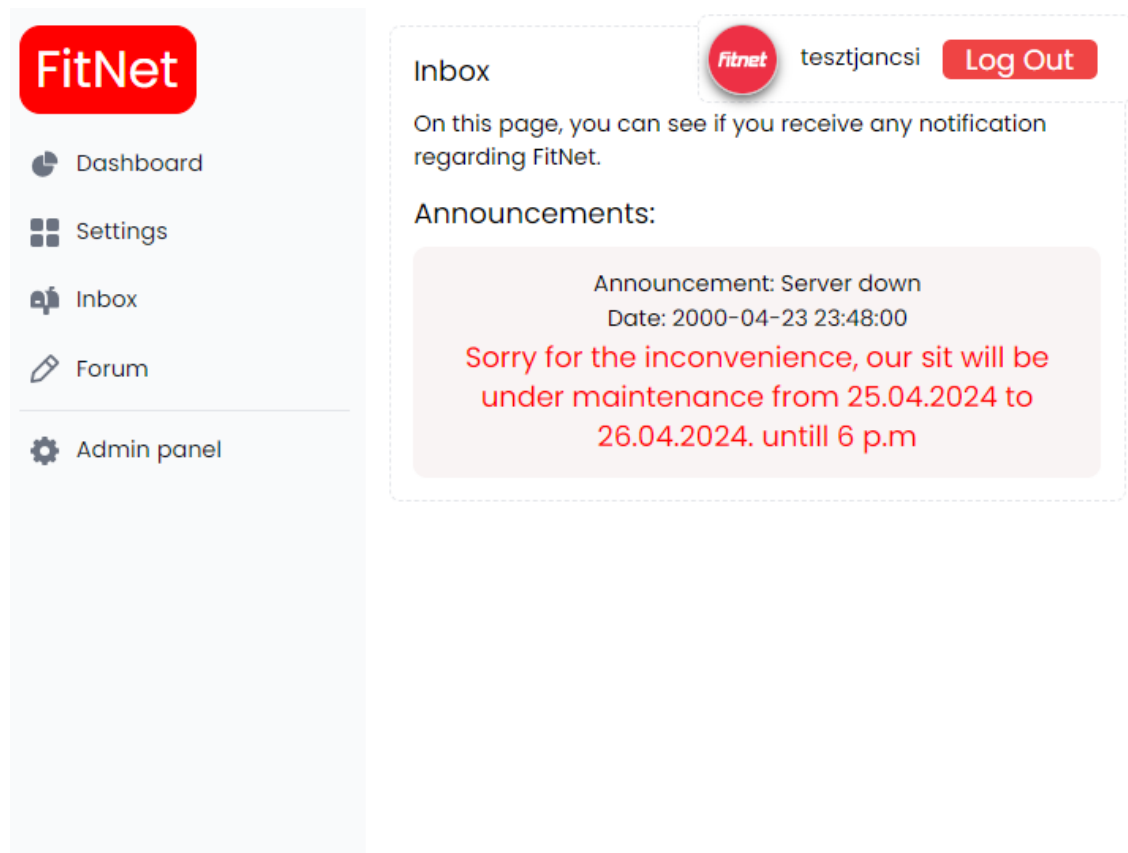
☐ Regenerate workout

[Regenerate](#)

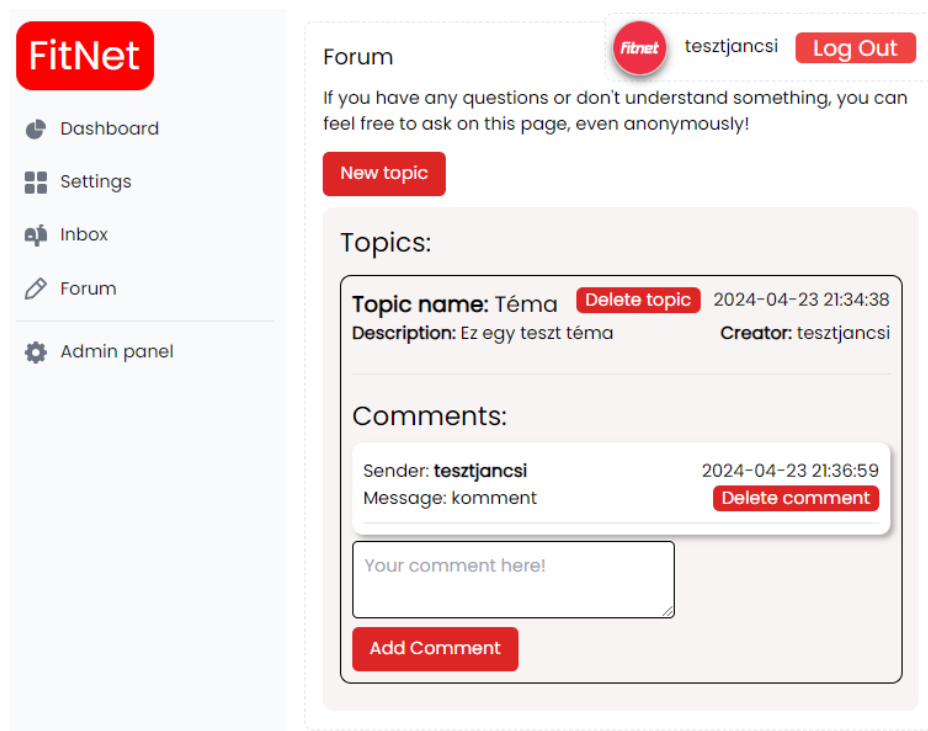
Get new password:

☐ I need a new password

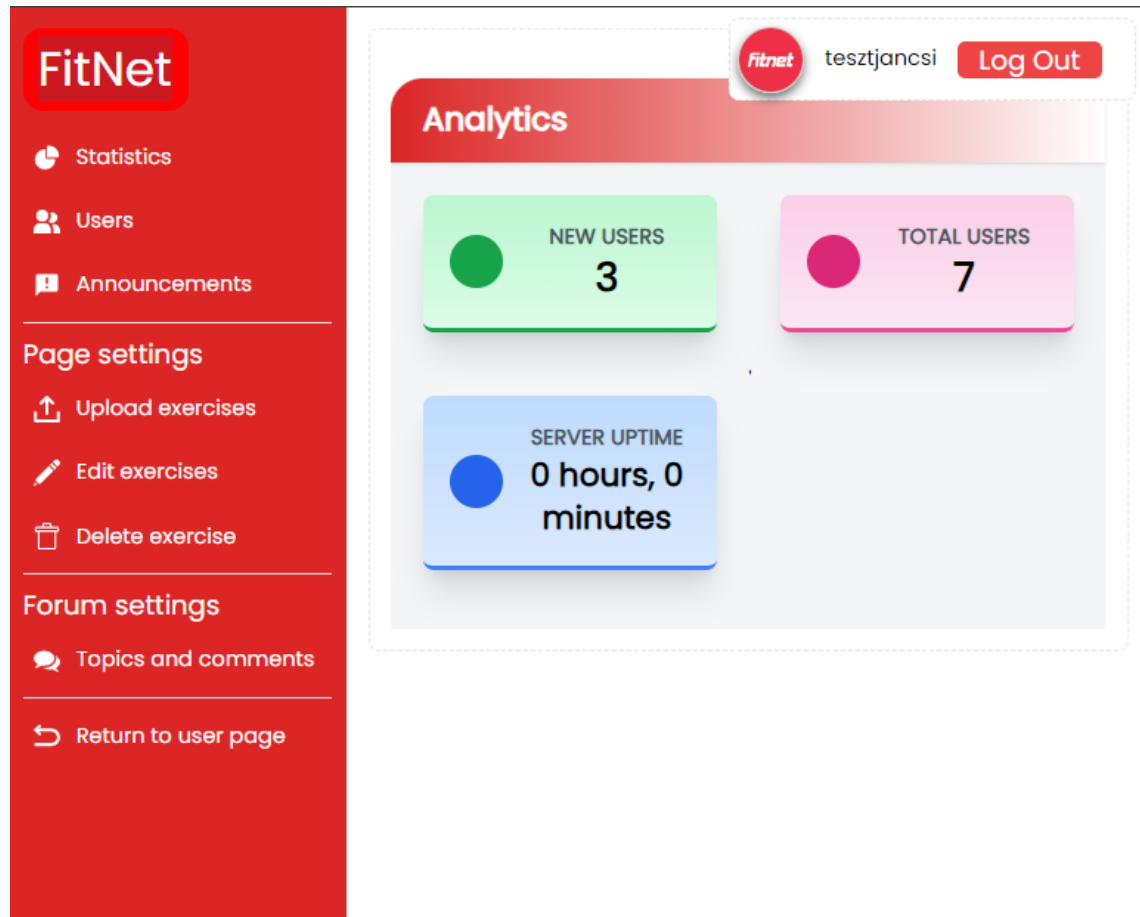
Az "Inbox" menüpontra kattintva megtekintheted az adminisztrátorok által küldött rendszerüzeneteket, például az oldal karbantartásáról.



A "Forum" menüpontra kattintva témákat tudsz létrehozni vagy hozzászólni meglévő témákhoz. A téma kiválasztása után lehetőség van törölni a saját témát vagy kommentjeidet.



Ha adminisztrátori jogokkal rendelkezel, az "Admin panel" menüpontra keresztül elérheted a Statistics, Users, Announcements, Upload exercises, Edit exercises, Delete exercise, Topic and comments, és Return to user page menüpontokkal rendelkező admin panelt. Az admin panel elsődleges oldal a Statistics, ahol megtekinthetők az olyan statisztikai adatok mint: összes felhasználó száma, új felhasználók, server uptime.



Ha a "Users" menüpontra kattintasz, kilistázódnak a felhasználók, ahol keresővel tudsz rájuk keresni név szerint. Az oldal lapozási lehetőséggel kilistázza a felhasználókat, és a "Ban" vagy "Edit user data" gombokkal módosíthatod vagy törölheted a felhasználó adatait.

FitNet

Statistics

Users

Announcements

Page settings

Upload exercises

Edit exercises

Delete exercise

Forum settings

Topics and comments

Return to user page

User settings

FitNet

tesztjancsi

Log Out

Search by Username: Enter username

Search

Fitnet


asd

Email: asd@gmail.com

Registration: 2024-04-16 20:44:43

Edit user data

Ban



Tibor

Email: tibi@gmail.com

Registration: 2024-04-16 20:52:33

103

Az "Announcements" menüpont alatt tudod küldeni vagy törölni a rendszerüzeneteket.

FitNet

Statistics

Users

Announcements

Page settings

Upload exercises

Edit exercises

Delete exercise

Forum settings

Topics and comments

Return to user page

Send announcement

Fitnet

tesztjancsi

Log Out

Name of announcement

Please input the announcement text

éééé. hh. nn. --:--:--

Send announcement

Delete announcement

Announcement: Server down

Date: 2000-04-23 23:48:00


Sorry for the inconvenience, our sit will be under maintenance from 25.04.2024 to 26.04.2024. untill 6 p.m


Delete announcement


Az "Page settings" részen keresztül pedig feltölthetsz új gyakorlatokat az Upload exercises menüpontnál, szerkesztheted az Edit exercises-re kattintva vagy törölheted a meglévőket a Delete exercise menüponton keresztül.

Upload exercises


FitNet


 Statistics


 Users

 Announcements


Page settings


 Upload exercises

 Edit exercises


 Delete exercise

Forum settings

 Topics and comments

 Return to user page


Exercise upload

 **tesztjancsi** **Log Out**

Name of exercise:

Description of exercise:

Short description of exercise:

Category of the workout: Weight lifting ▼ 

Body part of the workout: Upper body ▼

Picture of exercise:


Fájl kiválasztása


Nincs fájl kiválasztva


Upload

Edit exercises


FitNet


 Statistics


 Users

 Announcements


Page settings


 Upload exercises

 Edit exercises


 Delete exercise

Forum settings

 Topics and comments

 Return to user page

Edit Exercise


 **Fitnet**


tesztjancsi

Log Out

Select Exercise to Edit

Dumbbell press ▾








Edit

Delete exercise


FitNet


 Statistics


 Users

 Announcements


Page settings


 Upload exercises

 Edit exercises


 Delete exercise

Forum settings

 Topics and comments

 Return to user page



Delete Exercise

 testjancsi

Log Out

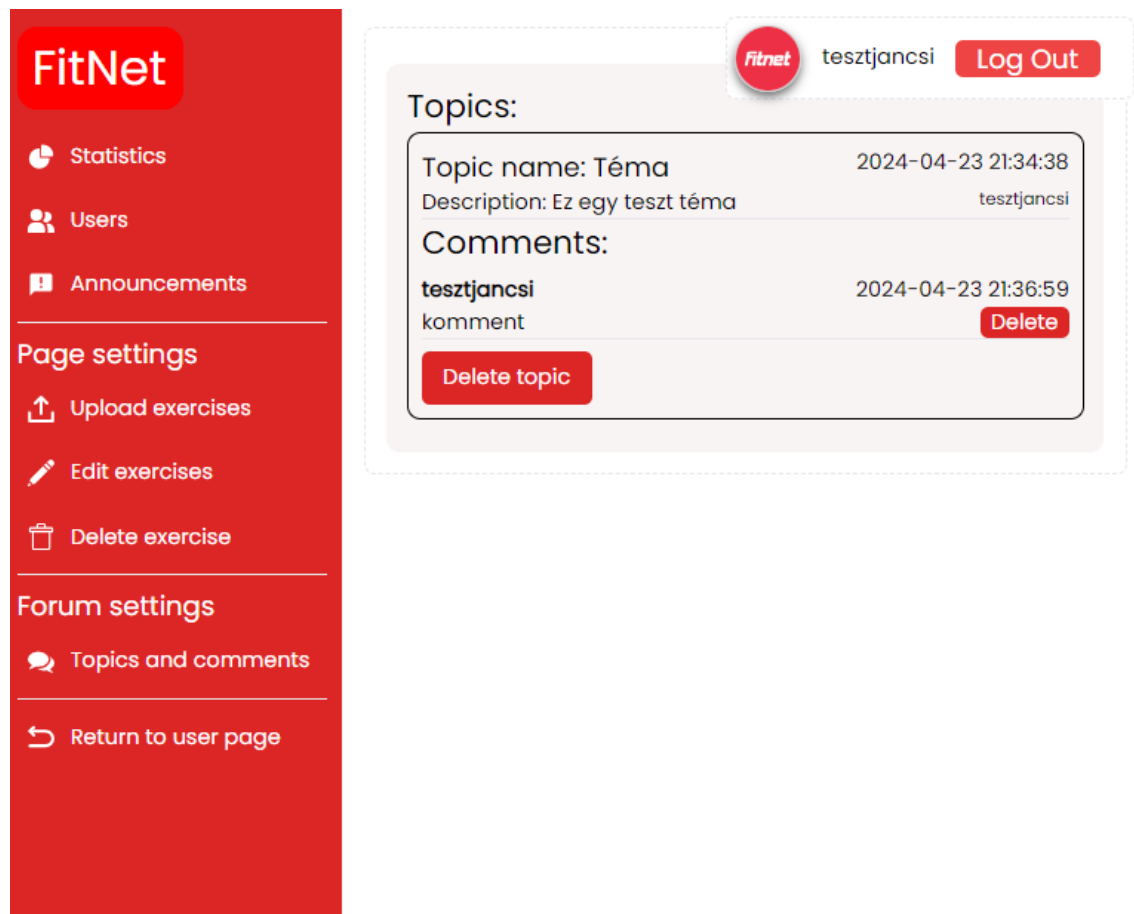
Select Exercise to Delete

Dumbbell press

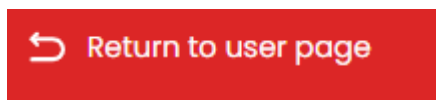


Delete

A "Topics and comments" részben találod a felhasználók által írt témákat és kommenteket, amelyeket szintén szerkeszthetsz vagy törölhetsz.



A "Vissza a felhasználói felülethez" lehetőséggel visszairányítunk a felhasználói felületre.



3 Irodalomjegyzék

AJAX irodalom – [https://hu.wikipedia.org/wiki/Ajax_\(programoz%C3%A1s\)](https://hu.wikipedia.org/wiki/Ajax_(programoz%C3%A1s))

REST api irodalom – <https://hu.wikipedia.org/wiki/REST>

Apache irodalom – https://hu.wikipedia.org/wiki/Apache_HTTP_Server

MySQL irodalom – <https://hu.wikipedia.org/wiki/MySQL>

JavaScript irodalom – <https://hu.wikipedia.org/wiki/JavaScript>

PHP irodalom – <https://hu.wikipedia.org/wiki/PHP>

PHP – <https://www.php.net/>

Stack Overflow - <https://stackoverflow.com/>

W3Schools – <https://w3schools.com/>

ChatGPT – <https://chat.openai.com/>

4 Mellékletek

GitHub: <https://github.com/Bazseee/Vizsgaremek>

Mappák:

dokumentacio – Tartalmazza a dokumentációt

adatbázis – Tartalmazza az feltöltött és üres adatbázis fájlait

web – Tartalmazza a weboldal forráskódjait, fájlait

server – Tartalmazza a Xampp telepítő fájlait

Mailtrap: <https://mailtrap.io/>