**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: Wiezl 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: |  | A vizsgaremek átvettem: |
| Vác, 2024. |  | Vác, 2024. |
| Konzulens |  | A szakképzést folytató intézmény felelőse |

**Tartalomjegyzék**

[**Hallgatói nyilatkozat** 3](#_Toc165357371)

[**Konzultációs lap** 4](#_Toc165357372)

[**Tartalomjegyzék** 5](#_Toc165357373)

[**Témaválasztás** 8](#_Toc165357374)

[**1** **Fejlesztői dokumentáció** 9](#_Toc165357375)

[1.1 Alkalmazás ismertetése 9](#_Toc165357376)

[1.2 Felhasznált programok, technológiák 9](#_Toc165357377)

[1.2.1 PHP 9](#_Toc165357378)

[1.2.2 JavaScript 9](#_Toc165357379)

[1.2.3 MySQL és Apache 9](#_Toc165357380)

[1.2.4 Ajax 10](#_Toc165357381)

[1.2.5 REST API 10](#_Toc165357382)

[1.3 Fejlesztői környezet 10](#_Toc165357383)

[1.4 Adatszerkezet 11](#_Toc165357384)

[1.4.1 Adatbázis alapadatai 11](#_Toc165357385)

[1.4.2 Kapcsolatok 12](#_Toc165357386)

[1.4.3 Táblák 13](#_Toc165357387)

[1.5 Használati esetmodell 22](#_Toc165357388)

[1.5.1 Felhasználói szerepek és funkciók 22](#_Toc165357389)

[1.6 Modulok, menüpontok ismertetése 23](#_Toc165357390)

[1.6.1 db.php 23](#_Toc165357391)

[1.6.2 landing.php 24](#_Toc165357392)

[1.6.3 user\_register.php 26](#_Toc165357393)

[1.6.4 user\_login.php 29](#_Toc165357394)

[1.6.5 introducing.php 32](#_Toc165357395)

[1.6.6 submit.php 32](#_Toc165357396)

[1.6.7 musclegroup.php 36](#_Toc165357397)

[1.6.8 menu.php 38](#_Toc165357398)

[1.6.9 dashboard.php 39](#_Toc165357399)

[1.6.10 settings.php 42](#_Toc165357400)

[1.6.11 inbox.php 47](#_Toc165357401)

[1.6.12 forum.php 49](#_Toc165357402)

[1.6.13 admin.php 51](#_Toc165357403)

[1.6.14 statistic.php 53](#_Toc165357404)

[1.6.15 user.php 54](#_Toc165357405)

[1.6.16 filter\_users.php 57](#_Toc165357406)

[1.6.17 toggle\_ban.php 58](#_Toc165357407)

[1.6.18 banned.php 59](#_Toc165357408)

[1.6.19 save\_user.php 60](#_Toc165357409)

[1.6.20 announcements.php 61](#_Toc165357410)

[1.6.21 upload\_announcements.php 62](#_Toc165357411)

[1.6.22 delete\_announcements\_work.php 62](#_Toc165357412)

[1.6.23 exercise\_upload.php 63](#_Toc165357413)

[1.6.24 get\_categories\_count.php 64](#_Toc165357414)

[1.6.25 exercis\_upload\_work.php 65](#_Toc165357415)

[1.6.26 filter\_exercise.php 66](#_Toc165357416)

[1.6.27 edit\_exercise.php 66](#_Toc165357417)

[1.6.28 save\_user.php 68](#_Toc165357418)

[1.6.29 delete\_exercis.php 69](#_Toc165357419)

[1.6.30 delete\_exercise\_work.php 70](#_Toc165357420)

[1.6.31 topics\_and\_comments.php 70](#_Toc165357421)

[1.6.32 delete\_topic.php 70](#_Toc165357422)

[1.6.33 delete\_comment.php 71](#_Toc165357423)

[1.6.34 logout.php 71](#_Toc165357424)

[1.6.35 add\_comment.php 72](#_Toc165357425)

[1.6.36 avatar\_upload.php 73](#_Toc165357426)

[1.6.37 check\_access.php 74](#_Toc165357427)

[1.6.38 create\_topic.php 74](#_Toc165357428)

[1.6.39 delete\_topic.php 75](#_Toc165357429)

[1.6.40 delete\_comment.php 76](#_Toc165357430)

[1.6.41 fetch\_week\_data.php 76](#_Toc165357431)

[1.6.42 functions.php 77](#_Toc165357432)

[1.6.43 submit\_comment.php 78](#_Toc165357433)

[1.6.44 get\_completed\_dates.php 79](#_Toc165357434)

[1.6.45 get\_exercise\_details.php 80](#_Toc165357435)

[1.6.46 get\_exercise.php 81](#_Toc165357436)

[1.6.47 regenerate\_workout.php 82](#_Toc165357437)

[1.6.48 save\_completed\_workout.php 82](#_Toc165357438)

[1.6.49 search.php 83](#_Toc165357439)

[1.6.50 update\_workout\_consent.php 83](#_Toc165357440)

[1.7 Tesztelés, tesztesetek 85](#_Toc165357441)

[1.7.1 Regisztráció tesztelés 85](#_Toc165357442)

[1.7.2 Bejelentkezés tesztelés 86](#_Toc165357443)

[1.7.3 Felhasználói adatváltoztatás tesztelés 86](#_Toc165357444)

[1.7.4 Feladat feltöltés tesztelés 87](#_Toc165357445)

[1.7.5 Szemelyre szabott feladatsor generálás tesztelés 88](#_Toc165357446)

[1.7.6 Feladatsor lejátszásához szükséges modul tesztelés 88](#_Toc165357447)

[1.8 Továbbfejlesztési lehetőségek 89](#_Toc165357448)

[1.8.1 A weboldal továbbfejlesztése 89](#_Toc165357449)

[**2** **Felhasználói dokumentáció** 90](#_Toc165357450)

[2.1 Alkalmazás ismertetése 90](#_Toc165357451)

[2.2 Rendszerkövetelmények 90](#_Toc165357452)

[2.3 Szükséges beállítások 91](#_Toc165357453)

[2.4 Alkalmazás használata 91](#_Toc165357454)

[**3** **Irodalomjegyzék** 109](#_Toc165357455)

[**4** **Mellékletek** 110](#_Toc165357456)

# **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 fitnesz 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
* Reszponzívitás 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

# **Fejlesztői dokumentáció**

## Alkalmazás ismertetése

A fitnesz webalkalmazás egy online platform, amely lehetővé teszi bármely felhasználó számára nem, életkor és testi állapotától független, hogy egészséges életmódot folytassanak és támogassák egészségüket és fitneszüket. 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 képzik, szemléltetés céljából lettek felhasználva.

## Felhasznált programok, technológiák

### 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.

### 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.

### 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.

### 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.

### 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 félé; a szerverek kéréseket dolgoznak fel és a megfelelő választ küldik vissza.

## 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 preprocesszor, amely lehetővé tette a CSS kód strukturáltabbá és könnyebben karbantarthatóvá tételét.

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

## Adatszerkezet

### Adatbázis alapadatai

MySQL kliens verziója: 5.7.26

Sql szerver: mysql

Adatbázis neve: mesterremek

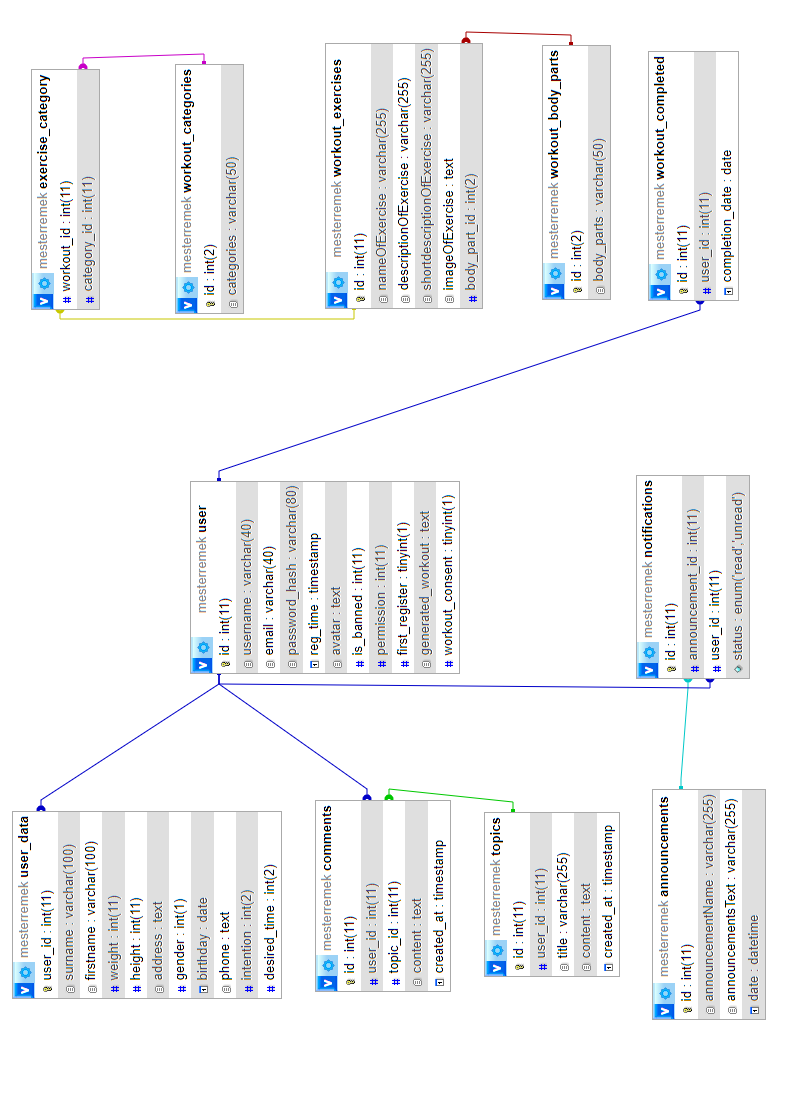
Tárolómotor: InnoDB

Alapértelmezett illesztés: utf8mb4\_general\_ci

Sql-parancs:



### Kapcsolatok



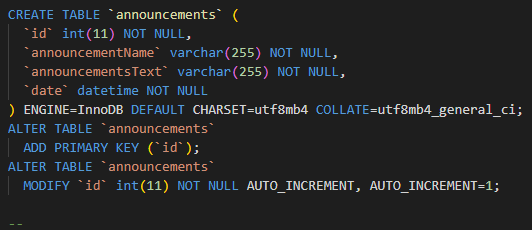
### Táblák

#### Az „announcements” tábla

A rendszerüzeneteket és annak kiküldésének ideját 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:

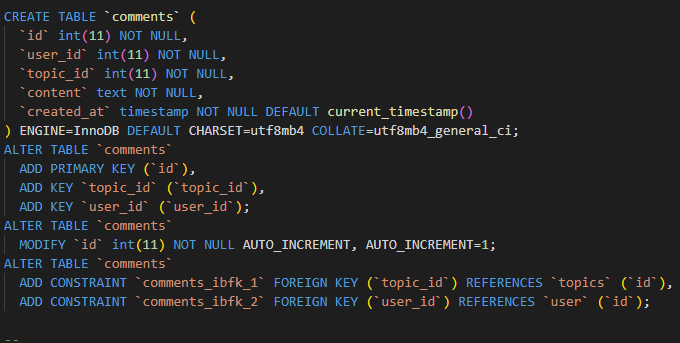


#### A „comments” tábla

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) | Felhasznló azonosítója | FK |
| topic\_id | int(11) | Topic azonosítója | FK |
| content | text | A komment tartalmát tárolja |  |
| created\_at | timestamp | Komment elkészítésének idejét tárolja |  |

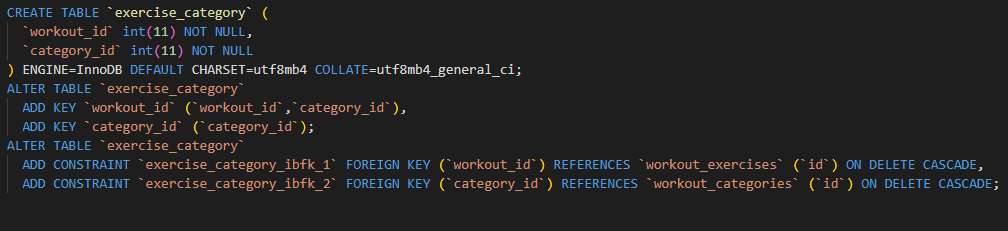
SQL-parancs:



#### Az „exercise\_category” tábla

Segédtábla 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 | inr(11) | Kategória azonosítója | FK |

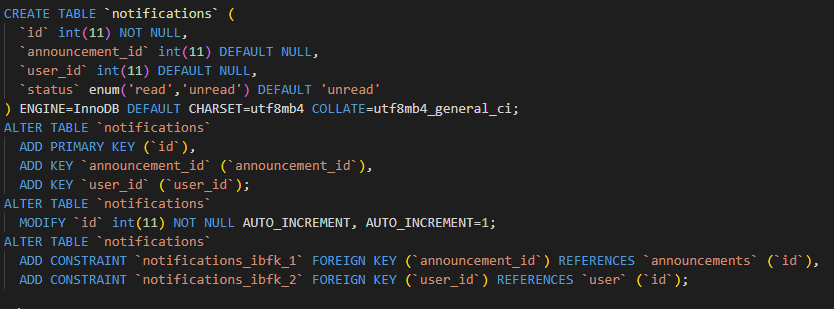
SQL-parancs:

#### A „notifications” tábla

A tábla a kiküldött rendszerüzentek 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:

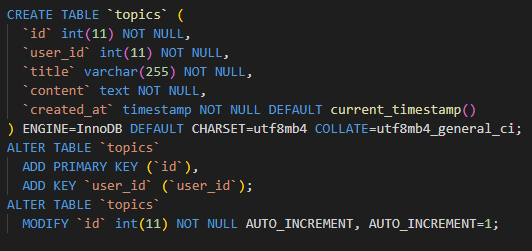


#### A „topics” tábla

A felhasználok 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 topi címét tárolja |  |
| content | text | A topic tartalmát tárolja |  |
| created\_at | timestamp | A topic létrehozásának az időpontját tárolja |  |

SQL-parancs:

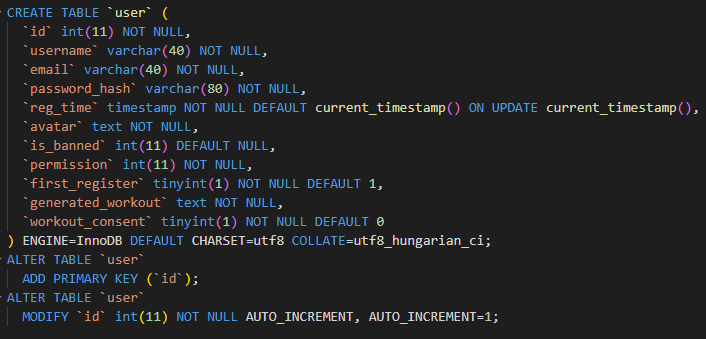


#### A „user” tábla

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 |  |
| passwor\_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:

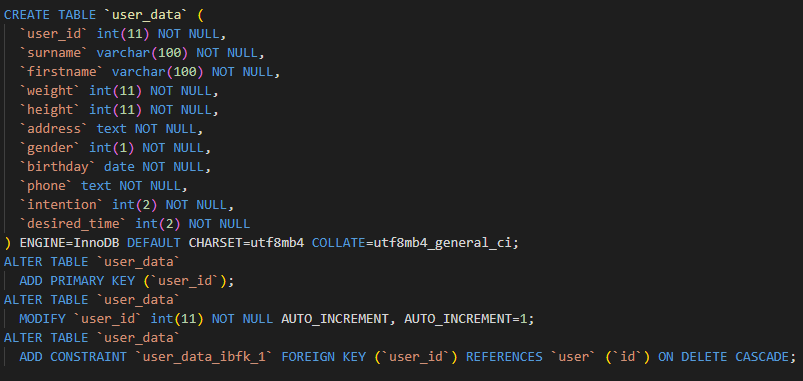


#### A „user\_data” tábla

A felhasználó személyes adatait tárolja.

|  |  |  |  |
| --- | --- | --- | --- |
| Mező neve | Típusa | Leírás | FK/PK |
| user\_id | int(11) | Egyedi belső azonosító | PK |
| surname | varchar(100) | A felhasználó vezetéknevét tárolja |  |
| firstname | varchar(100) | A felhasználó keresztnevét 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:

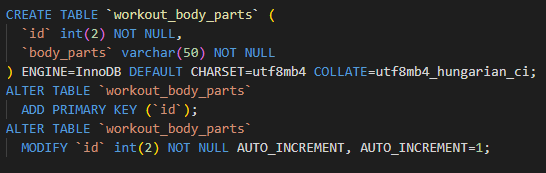


#### A „workout\_body\_parts” tábla

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:

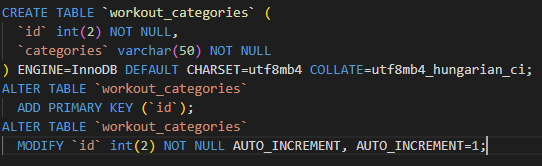


#### A „workout\_categories” tábla

Az edzés kategoriákat tárolja.

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

SQL-parancs:

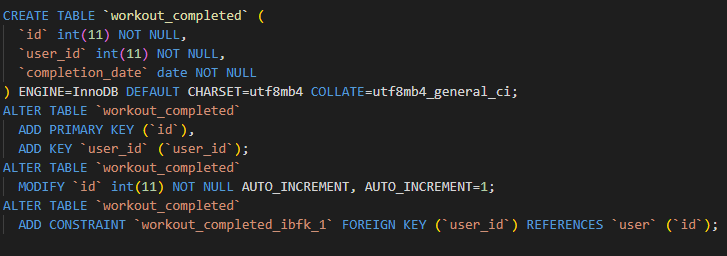


#### A „workout\_completed” tábla

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 elvégzet edzés dátumát tárolja |  |

SQL-parancs:

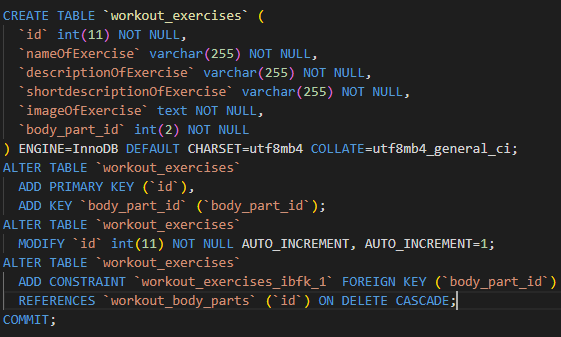


#### A „workout\_exercises” tábla

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:



## 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.

### Felhasználói szerepek és funkciók

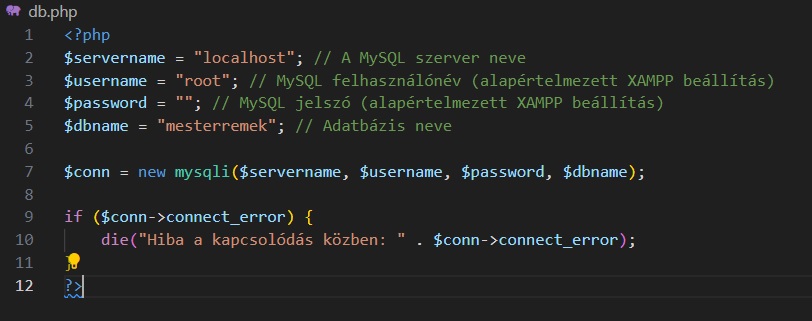
* Látogató
  + Regisztrációra való hivatkozás és regisztrációba lépés.
  + Böngészés az alkalmazás nyilvános tartalma között.
* Felhasználó
  + Bejelentkezés az alkalmazásba.
  + Edzésterv készítése és követése.
  + Közösségi funkciók használata, például kommentelés, posztolás stb.
* Adminisztrátor
  + 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

## Modulok, menüpontok ismertetése

### db.php

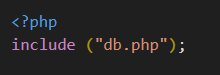
**Bevezetés**

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.

****

**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:



Ezután az adatbáziskapcsolat az $conn változón keresztül érhető el.

**Konfiguráció**

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

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

### 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.

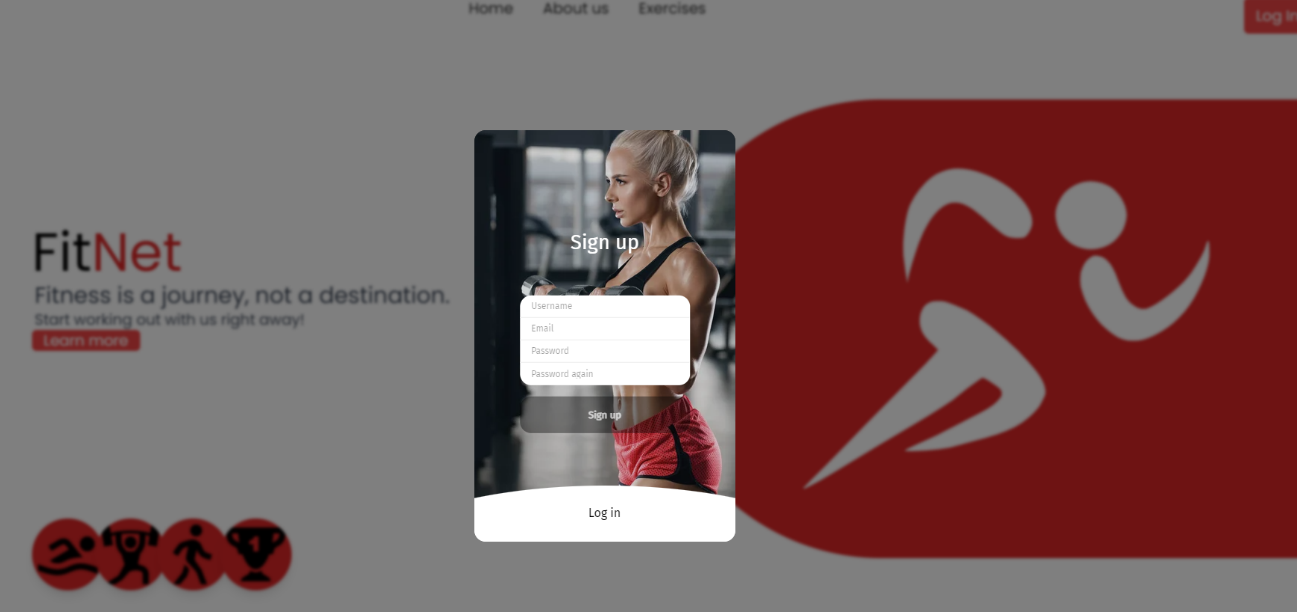


* **Home:** Ez a főoldal, amely visszavezet az oldal tetejére.
* **About us:** Ez az "About us" szakaszhoz visz, ahol információkat találhatunk az oldalról.
* **Exercises:** Ez a menüpont a musclegroup.php oldalra irányít át, ahol tudunk böngészni különböző izomcsoportok gyakorlatai között.

**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:**

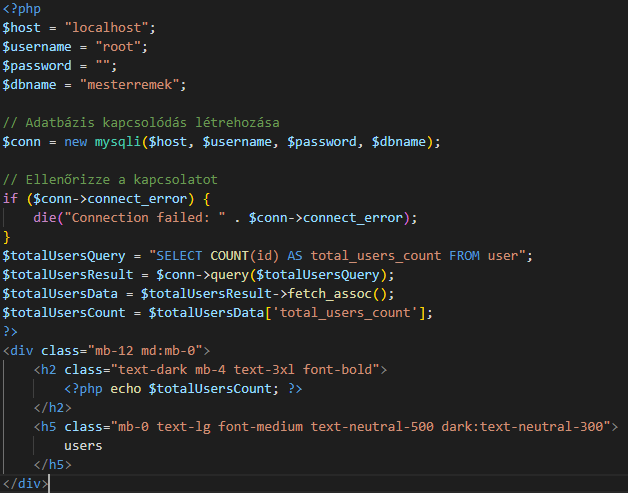
Az oldal elején megtalálható a session kezelése a PHP session\_start() függvénnyel. Ez arra szolgál, hogy a felhasználói munkamenetet kezelje az oldalon. A $\_SESSION tömböt használják az adatok tárolására és hozzáférésére a munkamenet során. Az if állítással ellenőrzik, hogy a felhasználó be van-e jelentkezve ($\_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.

****

**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.

### user\_register.php

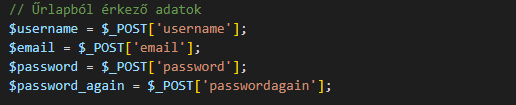
A user\_register.php a regisztrációs folyamatot valósítja meg a webalkalmazásunkban.

**Adatbázis kapcsolodás**

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

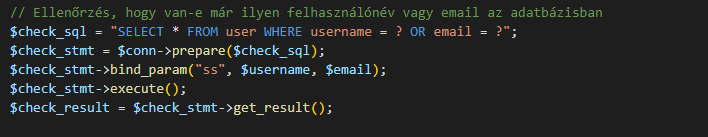
****

**Űrlapból érkező adatok**

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

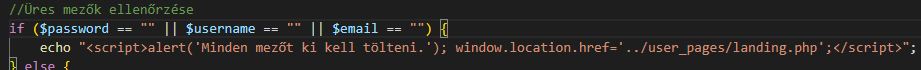
**Adatbázis ellenőrzése**

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

****

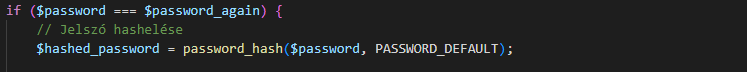
**Üres mezők ellenőrzése**

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

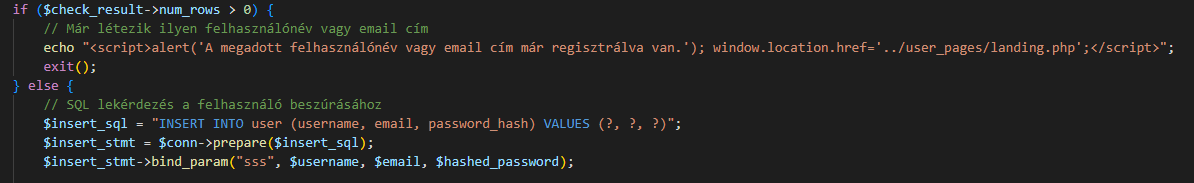


**Jelszó ellenőrzése és titkosítása (hashelése)**

Ellenőrzi, hogy a két jelszómezőben megadott jelszavak megegyeznek-e. Ha minden ellenőrzés sikeres volt, akkor a jelszót hasheli a PHP **password\_hash** függvény segítségével.



**Felhasználó beszúrása az adatbázisba**

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

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

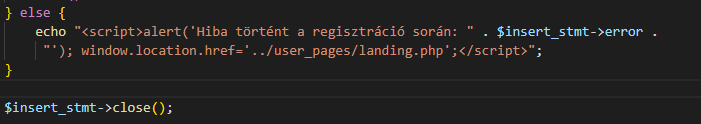
**Munkamenet indítása vagy folytatása**

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



**Hibakezelés**

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



**Adatbázis kapcsolat bezárása**

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



### 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.

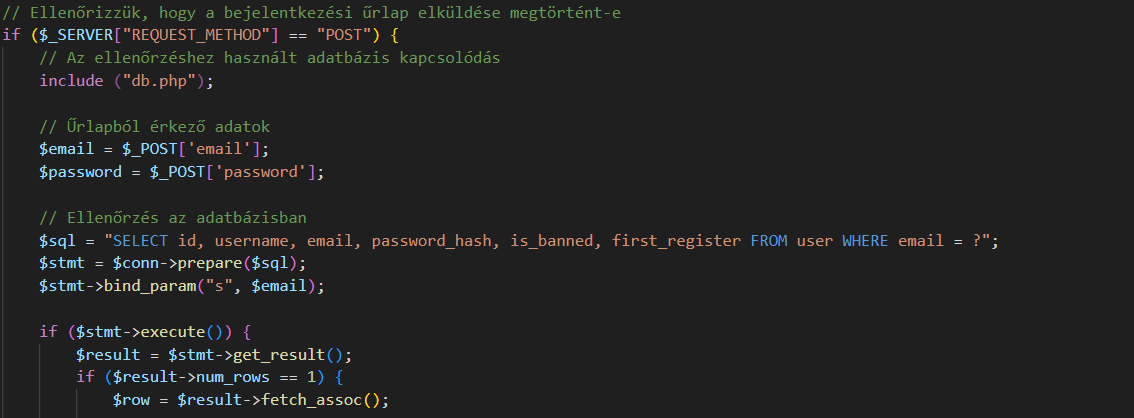
**Session kezelése**

A bejelentkeztetési állapotot a PHP session segítségével tárolja.

****

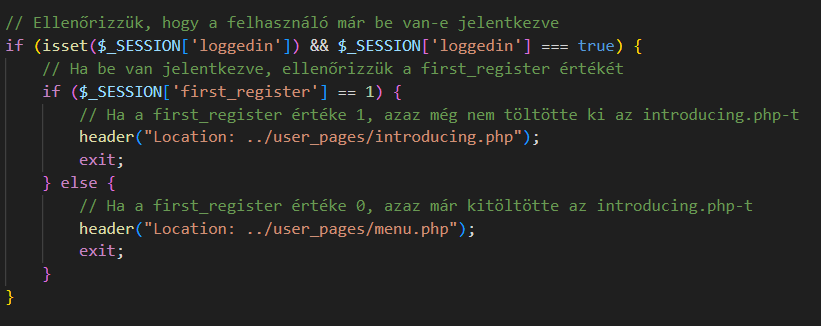
**Bejelentkezési ellenőrzése**

Az űrlapról érkező adatokat ellenőrzi az adatbázisban tárolt adatokkal.

****

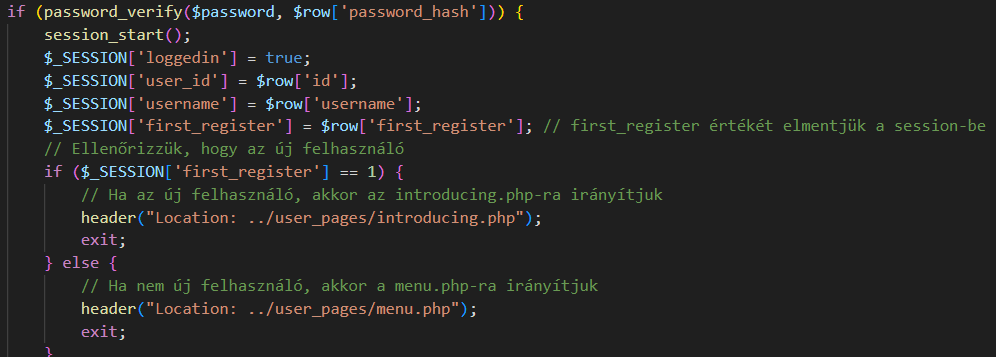
**Bejelentkezési státusz ellenőrzése, visszairányítás**

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.

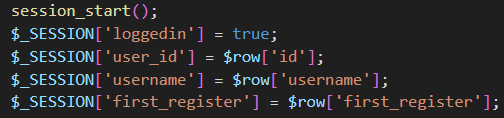


**Bejelentkezés és visszairányítás**

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

****

**Session kezelés**

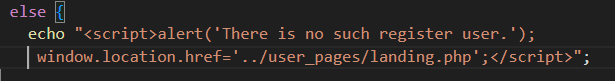
****Ha a bejelentkezés sikeres, létrehozza a session-t, és elmenti a szükséges adatokat, például a felhasználó azonosítóját és a bejelentkezési státuszt.

**Hibakezelés**

* **Hibás jelszó**

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

* **Nincs ilyen felhasználó**

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

* **Bejelentkezési hibák**

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

### 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.

### 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**

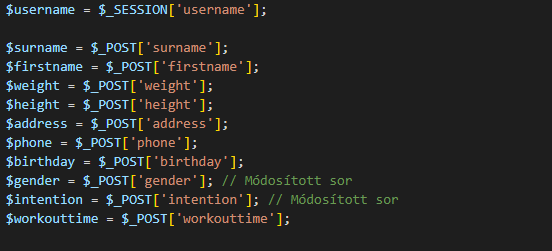
Az **include("db.php")** sorral a **db.php** fájl tartalmát betöltjük, amely tartalmazza az adatbáziskapcsolatot.

**Munkamenet indítása vagy folytatása**

**session\_start()** függvény segítségével elkezdjük vagy folytatjuk a munkamenetet.

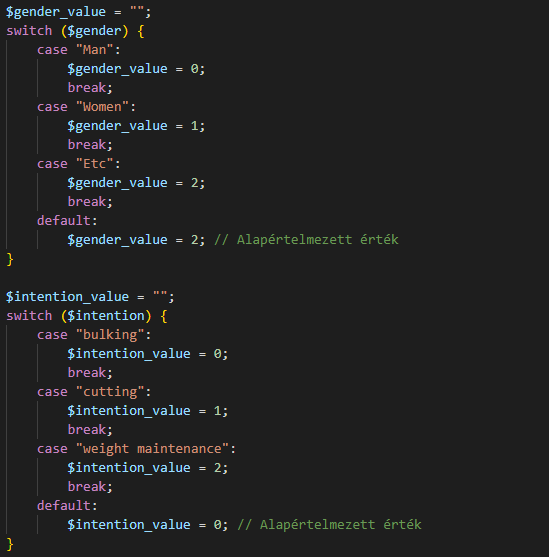
**Felhasználónév és POST kérésekből érkező adatok lekérése**

Az aktuális adatok lekérdezése a munkamenetből és tárolása változókban.

****

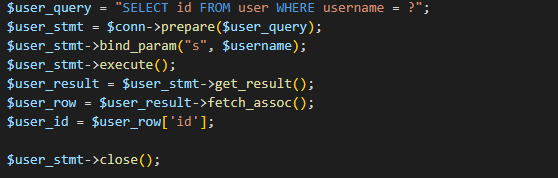
**Gender és Intention értékek beállítása**

A **$gender** és **$intention** értékek alapján egyedi értékek beállítása a **$gender\_value** és **$intention\_value** változókban.

****

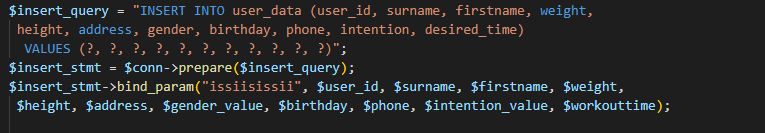
**Felhasználó azonosítása a user táblából**

Az adatbázisból lekérdezzük az aktuális felhasználó azonosítóját a **$user\_id** változóba.

****

**Felhasználó adatainak beszúrása a user\_data táblába**

Az űrlapról érkező adatok beszúrása az adatbázisba.

****

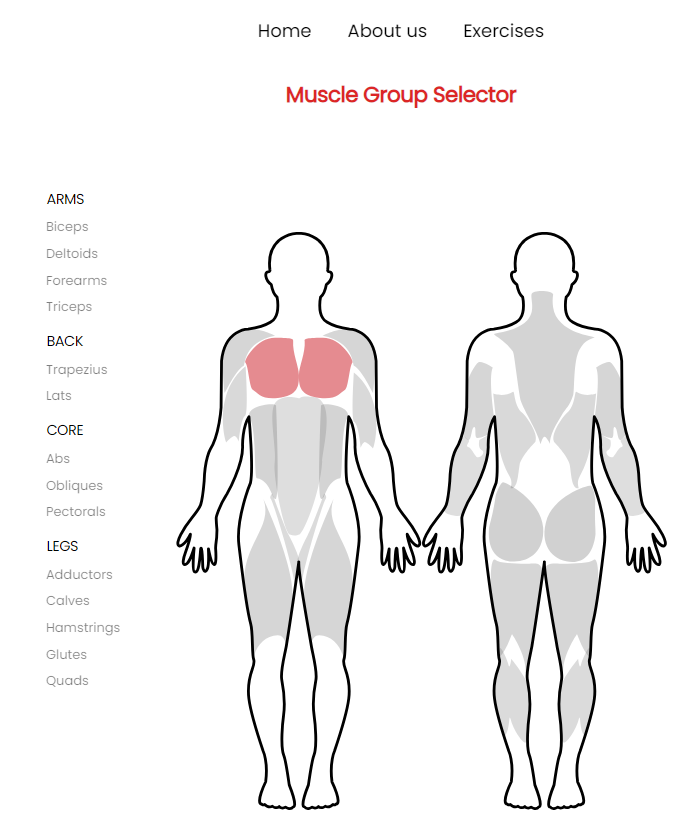
**Sikeres beszúrás esetén**

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.

****

### musclegroup.php

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



**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.

### 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**

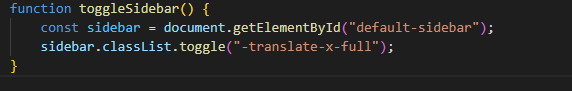
* **db.php**: Fájl az adatbáziskapcsolat létrehozásához.
* **check\_access.php**: Fájl a felhasználói bejelentkezési állapot és hozzáférési engedélyek ellenőrzésére.
* **dashboard.php**, **exercises.php**, **inbox.php**, **users.php**, **upgrade.php**, **settings.php**, **contact.php**, **forum.php**: 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:**

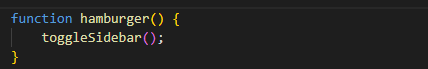
1. **Felhasználói Hitelesítés:**
   * Ellenőrzi, hogy a felhasználó be van-e jelentkezve (**$\_SESSION['loggedin']**) és átirányít a bejelentkezési oldalra, ha nem átirányit a bejelentkezési oldalra.
   * Felhasználói információkat kér le, beleértve a felhasználónevet, a profilképet és az olvasatlan értesítések számát.
2. **Oldalsáv Navigáció:**
   * Összecsukható oldalsáv navigációs menü a különböző alkalmazásrészletek eléréséhez.
   * A szakaszok közé tartozik a Dashboard, Settings, Inbox, Forum és Admin panel (ha a felhasználónak van adminisztrátori jogosultsága).
3. **Dinamikus Oldalbetöltés:**
   * Dinamikusan betölti a tartalmat a kiválasztott oldal alapján (**$\_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.



1. **hamburger() függvény működése:**
   * Ez a függvény egyszerűen meghívja a **toggleSidebar()** függvényt, hogy váltson az oldalsáv megjelenítése és elrejtése között.



### 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 (Nyílgombok) 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** fájltól 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** fájltól 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 avalaibe” 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**

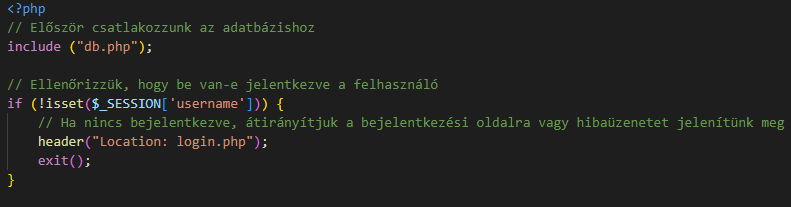
* **fetch\_week\_data.php**: Az edzésütemterv JSON adatának lekérése és feldolgozása.
* **get\_completed\_dates.php**: A befejezett dátumok lekérése az adatbázisból a vizuális jelzéshez.
* **get\_exercises.php**: Gyakorlat részleteinek lekérése a kiválasztott dátum alapján.
* **get\_exercise\_details.php**: 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.

### 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.



**Felhasználóid adatok lekérése**

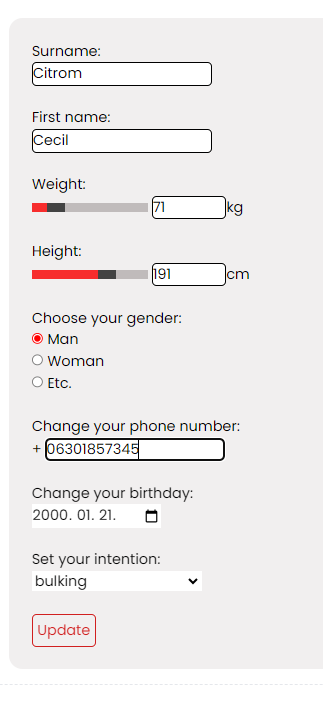
A bejelentkezett felhasználóhoz tartozó adatok lekérdezése az adatbázisból. Az adatokat változókba mentjük a későbbi felhasználás céljából.

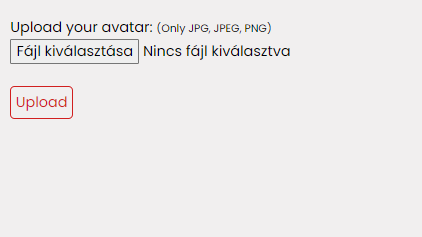
**Adatok frissítése**

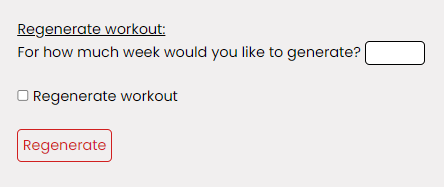
Ha a felhasználó az "Update" gombra kattintott, az újonnan megadott adatokat frissítjük az adatbázisban. Az adatok frissítésekor az űrlapból érkező POST adatokat használjuk.

**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.







**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.



### inbox.php

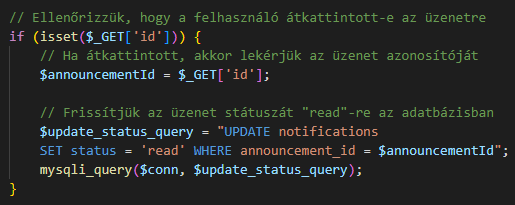
A **inbox.php** lehetővé teszi a felhasználók számára, hogy megtekintsék az értesítéseiket a FitNet alkalmazásból. Az oldalon megjelennek az értesítések, amelyeket az adatbázisból lekérdeznek és megjelenítenek.

**Adatbáziskapcsolat**

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

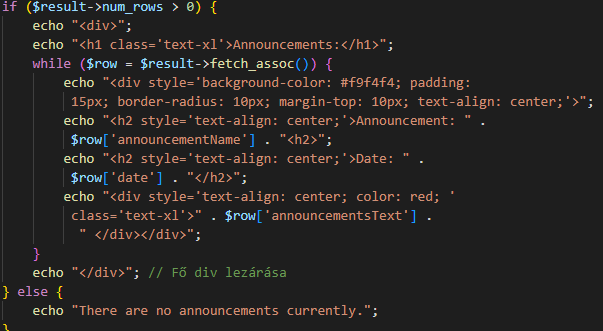
**Értesítések kezelése**

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

****

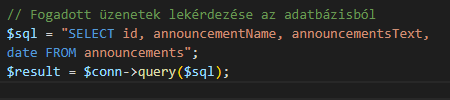
**HTML generálása**

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



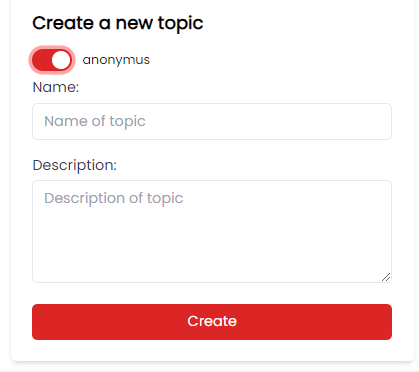
**Üzenetek lekérdezése adatbázisból**

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



### 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.

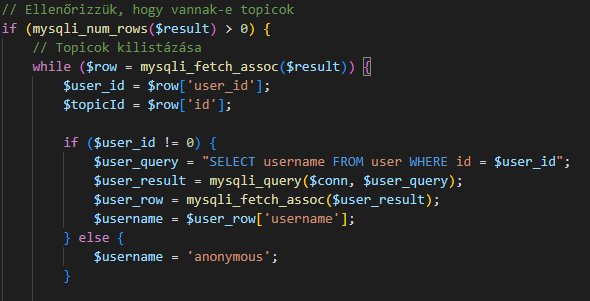


**Adatbázis kapcsolat**

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

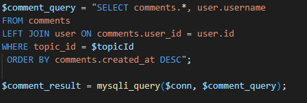
**Fórum témák listázása és megjelenítése**

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.



**Hozzászólások listázása és megjelenítése**

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



**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 elrejti 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, elrejti azokat, ha nem, akkor megjeleníti. A függvény továbbá megjeleníti vagy elrejti 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 válaszát 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.

### 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.

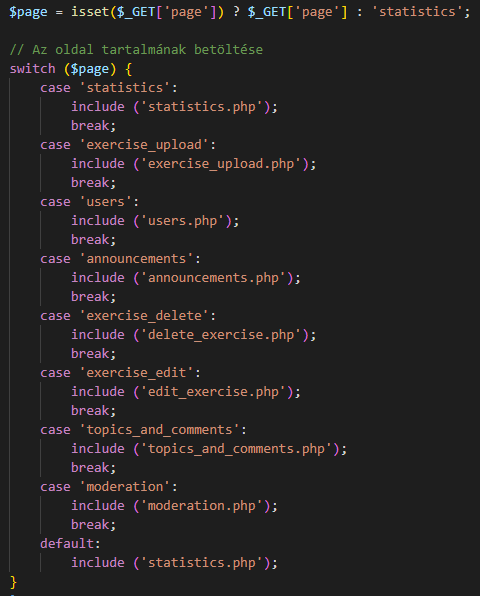
**Avatar kezelése**

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.



**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.



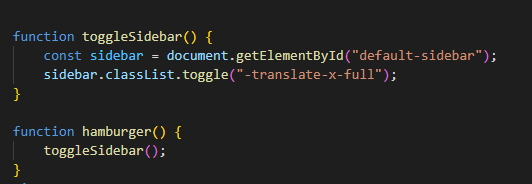
**Navigációs menü ismertetése**

* **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 elrejteni. A **hamburger()** függvény az oldalsávot megjeleníti vagy elrejti, amikor a hamburger ikonra kattintanak.

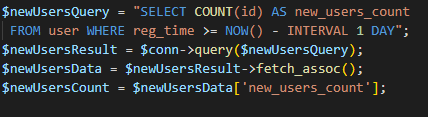


### statistic.php

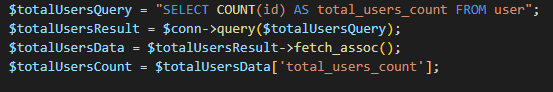
A **statistics.php** a FitNet webalkalmazás statisztikai adatainak megjelenítését végzi, beleértve az új felhasználók számát az elmúlt 24 órában, az összes felhasználó számát és a szerver uptime-ot.

**Funkciók**

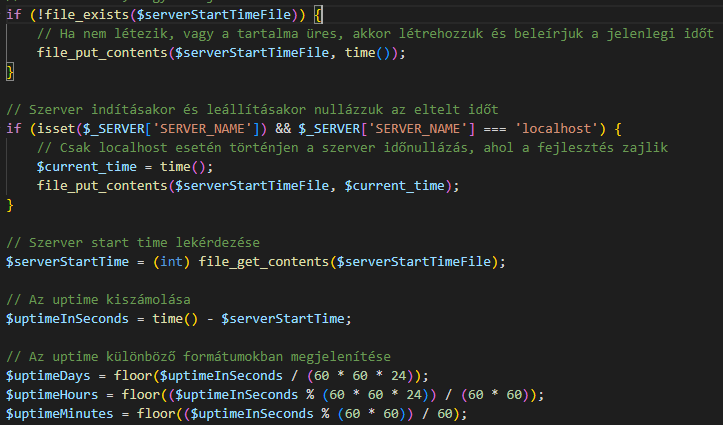
1. **Új Felhasználók Megjelenítése**: Lekérdezi az új felhasználók számát az elmúlt 24 órában a **user** táblából, majd megjeleníti ezt az adatot a felhasználókhoz tartozó kártyában.



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



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



### user.php

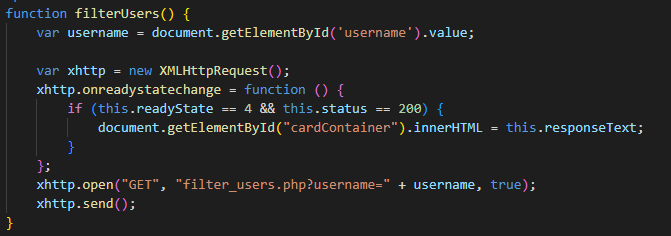
A **user.php** felhasználói beállítások felületet nyújt, ahol az adminisztrátorok megtekinthetik, szűrhetik, szerkeszthetik és kezelhetik a felhasználói fiókokat a FitNet rendszerben. Ez az oldal lehetővé teszi az adminisztrátorok számára olyan tevékenységek végrehajtását, mint például a felhasználók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

**Fájlstruktúra**

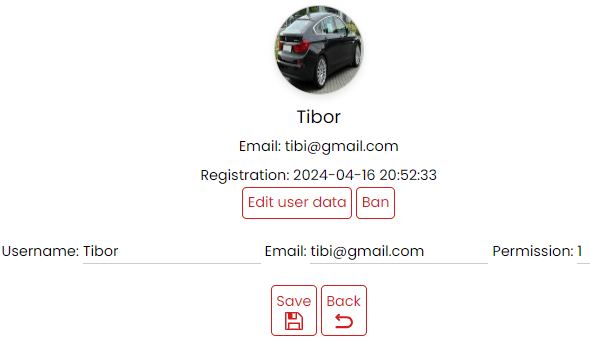
* **filter\_users.php**: Szerveroldali szkript, amely az AJAX kéréseket kezeli a felhasználók szűrésére felhasználónév alapján.
* **toggle\_ban.php**: Szerveroldali szkript, amely az AJAX kéréseket kezeli a felhasználók tiltásának/feloldásának kapcsolásához.
* **save\_user.php**: Szerveroldali szkript, 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.



1. **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).



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



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

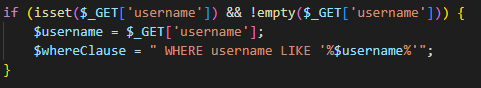


### filter\_users.php

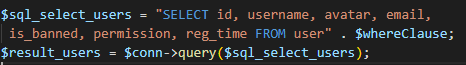
A **filter\_users.php** lehetővé teszi a felhasználók keresését és szűrését a rendszerben tárolt adatok alapján.

**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.



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



1. **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.
2. **Adatbázis kapcsolat bezárása**: Az adatbáziskapcsolat bezárása a művelet végén.

### 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.



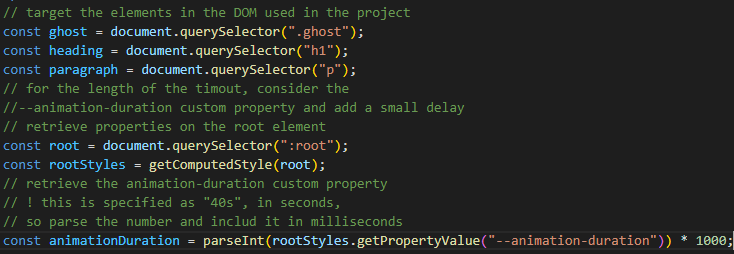
### banned.php

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.



### 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 ”sesssion\_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.



### announcements.php

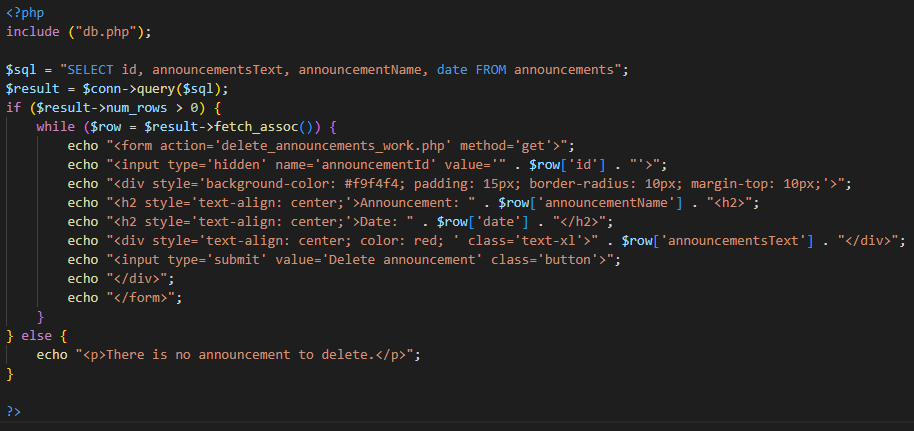
A FitNet **announcements.php** oldala egy platformként szolgál közlemények küldésére és törlésére. Az adminoknak lehetőséget biztosít új közlemények létrehozására és meglévő közlemények listájának megtekintésére, valamint azok törlésére.

**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.



### 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 változókba menti, mint például **announcementName**, **announcementInput** és **announcementDate**.
* Az SQL lekérdezéssel az adatokat beszúrja az **announcements** táblába.
* Ha sikeres volt a beszúrás, a felhasználókhoz 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.

### 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.

### exercise\_upload.php

Az **exercise\_upload.php** fájl egy HTML űrlapot tartalmaz, amely lehetővé teszi felhasználók számára, hogy gyakorlatokat töltsenek fel az adatbázisba.



**Javascript függvények**

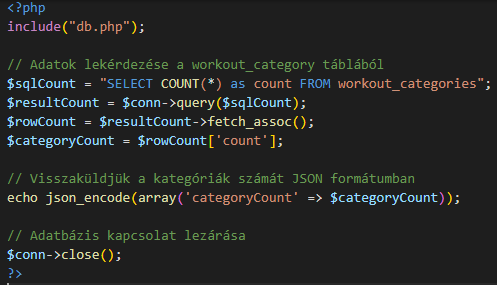
1. **addSelect(className)**:
   * Ez a függvény felelős az új kategória vagy testrész mező hozzáadásáért az űrlaphoz.
   * Paraméterként megkapja a mezők osztálynevét, amelyekhez hozzá kell adni az új mezőt.
   * AJAX hívást használ a **get\_categories\_count.php** fájlhoz, hogy meghatározza, hány további 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.

### 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.



### 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.

### filter\_exercise.php

A **filter\_exercise.php** felelős az edzésgyakorlatok szűréséért és megjelenítéséért az adatbázisból.

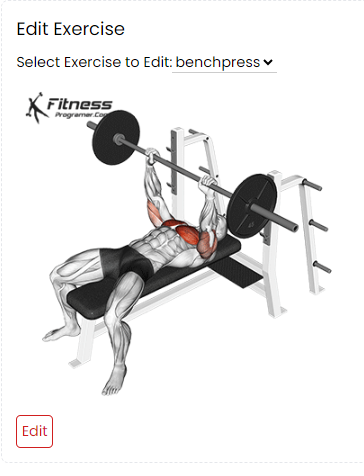
**Funkciók**

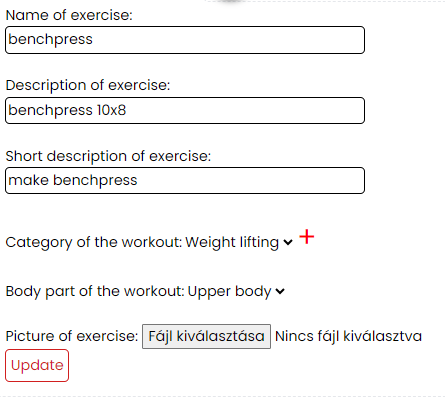
1. **Adatbázis kapcsolat**: Az első részben a db.php csatlakozik az adatbázishoz.
2. **Keresési feltételek ellenőrzése**: Ellenőrzi, hogy van-e keresési paraméter a GET kérésben (pl. **exercisename**), és ha van, létrehozza a megfelelő WHERE feltételt az SQL lekérdezéshez.
3. **Edzésgyakorlatok lekérdezése**: 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();).**

### 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.





### save\_user.php

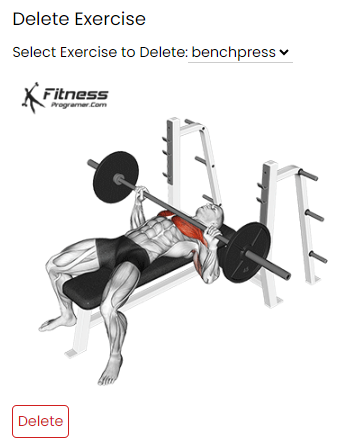
A **save\_user.php** felelős az adatbázisban tárolt felhasználói adatok frissítéséért az adott felhasználó módosítása esetén.

**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 változókban.
4. **Adatbázis kapcsolat megnyitása**: A **db.php** fájl beillesztésével létrehozzuk 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 változókban, 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.

### 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 kiválasztják és töröljék azokat.



### 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**

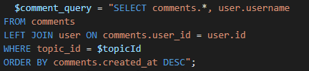
1. **Adatbázisból való törlés**: Az oldal eltávolítja a megadott **id**-vel azonosított gyakorlatot az adatbázisból.
2. **Kép fizikai törlése**: Amennyiben a törölni kívánt gyakorlathoz tartozik kép, az oldal törli azt a fájlszerverről.
3. **Értesítés**: A felhasználót értesíti a művelet eredményéről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

### 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.



### 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.

### delete\_comment.php

A **delete\_comment.php** felelős a kommentek törléséért az adatbázisból.

**Fájl strukturája és működése**

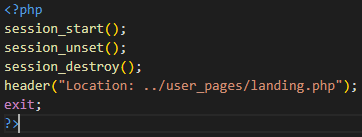
* A fájl elején a **db.php** fájl beilleszti az adatbázis kapcsolatot.
* Az oldal ellenőrzi, hogy a **comment\_id** érték át lett-e adva a GET kérésben.
* Ha a **comment\_id** be lett adva, akkor a következő lépések történnek:
  + - A **comment\_id** értékét kinyerik a GET kérésből.
    - Végrehajtják 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.

### 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 oldalátirányítás segítségével visszatérnek a bejelentkező oldalra, ami a **landing.php** oldal. Az **exit** függvénnyel azonnal leállítják a fájl végrehajtását, így nem folytatódik további kódvégrehajtás.



### 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.

### 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 kapcsolatot 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.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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. **$workouts = array();**: Ebben a tömbben tároljuk majd a generált edzéstervet hetek és napok szerint.
4. A **for** ciklus segítségével generáljuk meg a feladatsorokat a megadott időtartamra. Az edzések időtartama minden egyes hétnek az elején kezdődik újra.
5. A belső **for** ciklus segítségével minden héten generálunk 7 napos edzéstervet. Az első héten az első nap kihagyása, valamint minden harmadik nap pihenőnap beiktatása történik.
6. A **date** függvénnyel kiszámítjuk az aktuális nap dátumát a héten belül.
7. SQL lekérdezéseket hajtunk végre az edzésterv összeállításához. Az **upperBodyQuery** és **lowerBodyQuery** változókban tároljuk azokat a lekérdezéseket, melyek az upper body és lower body gyakorlatokat kérdezik le a felhasználó szándéka (**$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.

### submit\_comment.php

A **submit\_comment.php** PHP fájl felelős az új komment felvételéért az adatbázisba egy adott témahoz.

**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.

### 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 kérési módszer**: Ha a kérés nem GET metódussal érkezett, akkor 405-ös hibakódot küldünk vissza, és kilépünk a programból a "Method Not Allowed" üzenettel.

### 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.

### 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.

### 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.

### 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.

### 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 kiíratása.
6. Az adatbázis kapcsolat bezárása a kód végén, hogy ne foglalja a kapcsolatot feleslegesen.

### 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.

## Tesztelés, tesztesetek

A tesztelés szintjei a következőek:

* 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.

### Regisztráció tesztelés

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

A tesztesetek a következőek:

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

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

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

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

1. **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.

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

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

1. **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.

### Bejelentkezés tesztelés

A bejelentkezésnél használjuk az előzőleg beregisztrált felhasználói adatokat.

A tesztesetek a következőek:

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

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

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

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

1. **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.

1. **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.

1. **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.

1. **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.

### Felhasználói adatváltoztatás tesztelés

A tesztesetek a következőek:

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.

1. **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.

1. **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.

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

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

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

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

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

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

1. **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.

### Feladat feltöltés tesztelés

Töltsünk fel az edzésterv előállításához szükséges feladatot.

A tesztesetek a következőek:

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.

1. **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.

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

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

1. **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.

1. **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.

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

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

### Szemelyre szabott feladatsor generálás tesztelés

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

A tesztesetek a következőek:

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.

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

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

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

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

### Feladatsor lejátszásához szükséges modul tesztelés

A tesztesetek a következőek:

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.

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

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

1. **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.

1. **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.

## Továbbfejlesztési lehetőségek

### 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 beállításaiknál 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.

# **Felhasználói dokumentáció**

## Alkalmazás ismertetése

A FitNet webalkalmazás egy online platform, amely lehetővé teszi a felhasználók számára, hogy egészséges életmódot folytassanak és támogassák egészségüket és fitneszüket. 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.

## Rendszerkövetelmények

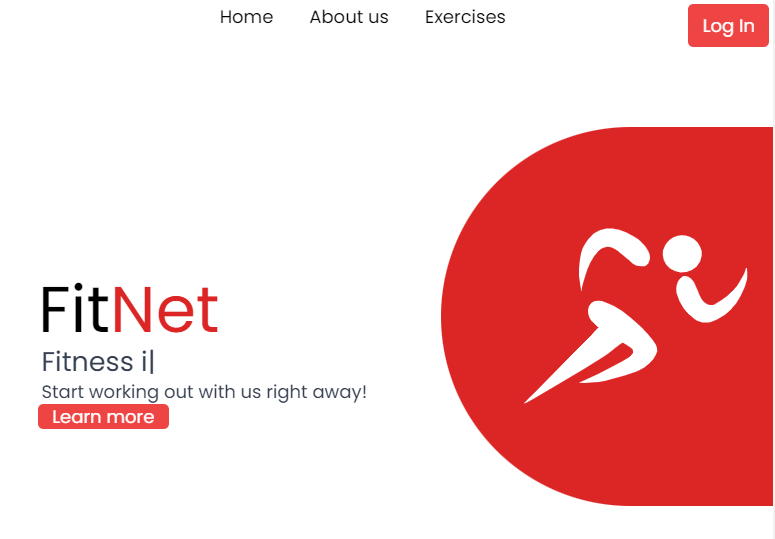
* Hardverkövetelmények:
  + Operációs rendszer: Az alkalmazás támogatja a legnépszerűbb operációs rendszereket, mint például Windows, macOS és Linux.
  + Processzor: Legalább egy közepes vagy magas teljesítményű processzor, amely képes futtatni a böngészőt és a webalkalmazást.
  + Memória: Ajánlott legalább 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.

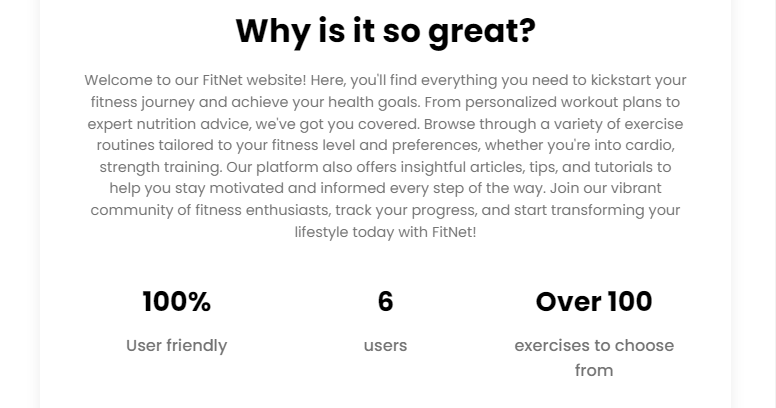
## Szükséges beállítások

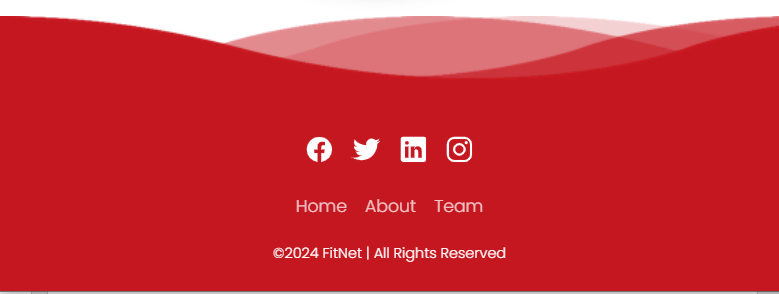
* Fiók létrehozása vagy bejelentkezés:
  + A index oldalra érve azonnal észrevehető a jobb felső sarokban található bejelentkezési gomb, amelynek segítségével könnyedén beléphetünk vagy regisztrálhatunk.
    - Ha a bejelentkezést választjuk, akkor lehetőségünk nyílik személyre szabott edzéstervet használni, 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.

## Alkalmazás használata

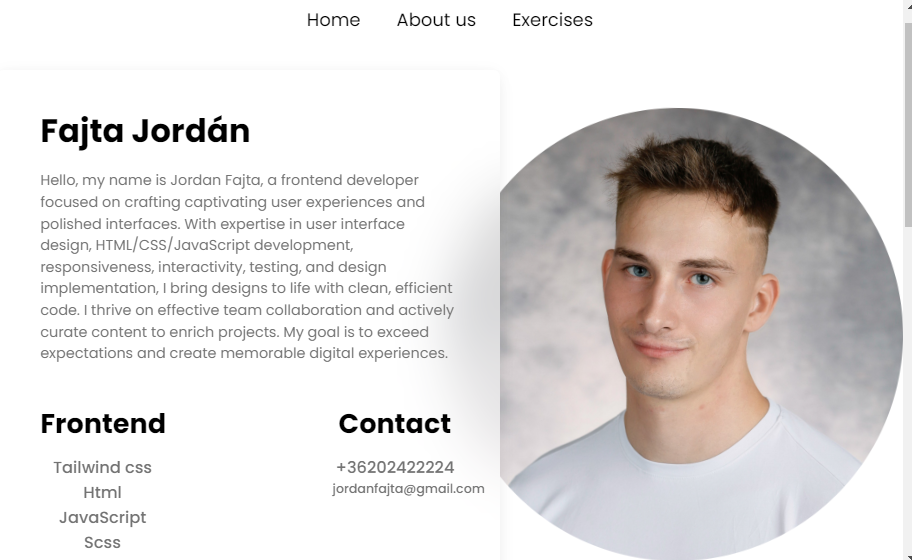
Az oldalunk főoldala a landing.php, ami a FitNet alkalmazás főoldala. Felül egy hárommenüpontos menü található: "Home", "About us" és "Exercises". Ha lefelé görgetsz, eljutsz a "Rólunk" szekcióhoz, majd a lábléchez, ahol újra elérhetővé válnak a "Home", "About" és a "Team" oldalra mutató link.

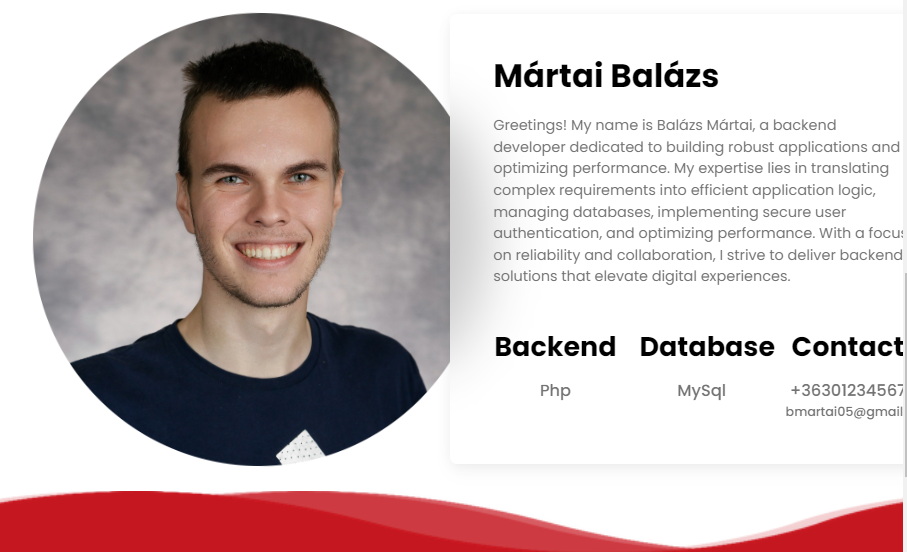




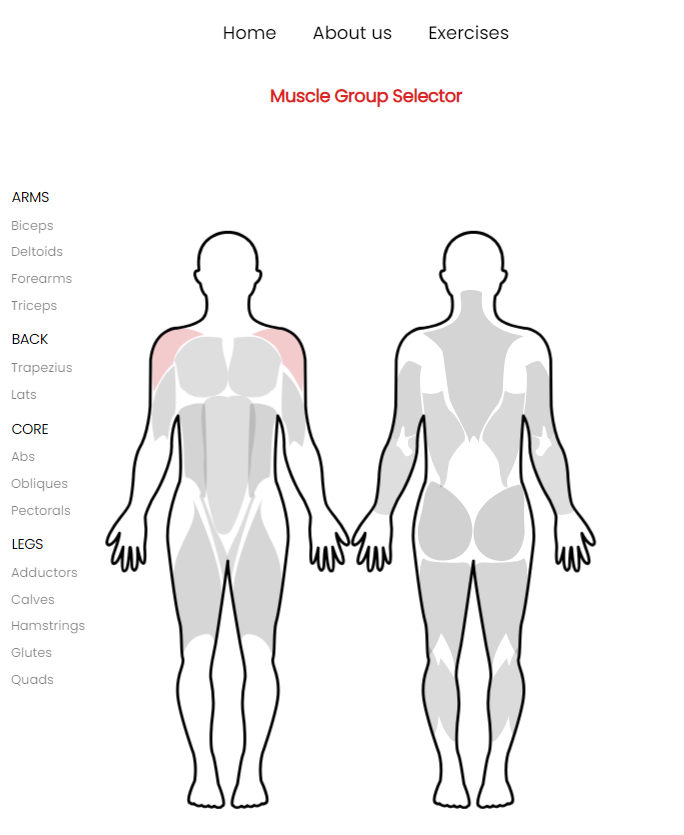


A lábléc Home menüpontját választva az oldal tetejére dobja, az About kiválasztása esetén az oldal about us részéhez navigálodik a felhasználó. Ha a Team oldalt választjuk akkor az oldal alkotókról tudhatunk meg többet a team oldalon.



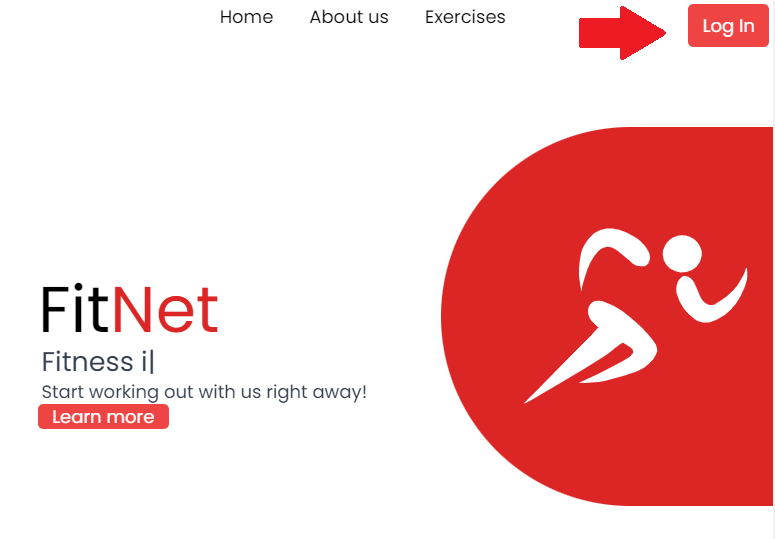


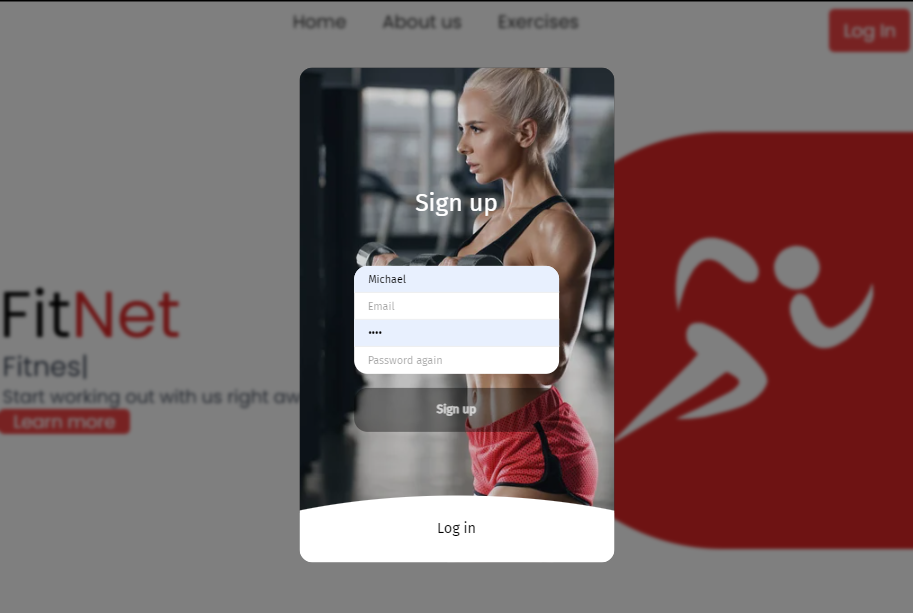
Ha a vízszintes menüben a "Exercises" opciót választod, átirányít a musclegroup oldalra, ahol anatómiai test segítségével válogathatsz az izomcsoportok között. Kiválasztva egyet, kilistázzuk az izomcsoport gyakorlatait animált GIF-képekkel, leírásokkal.



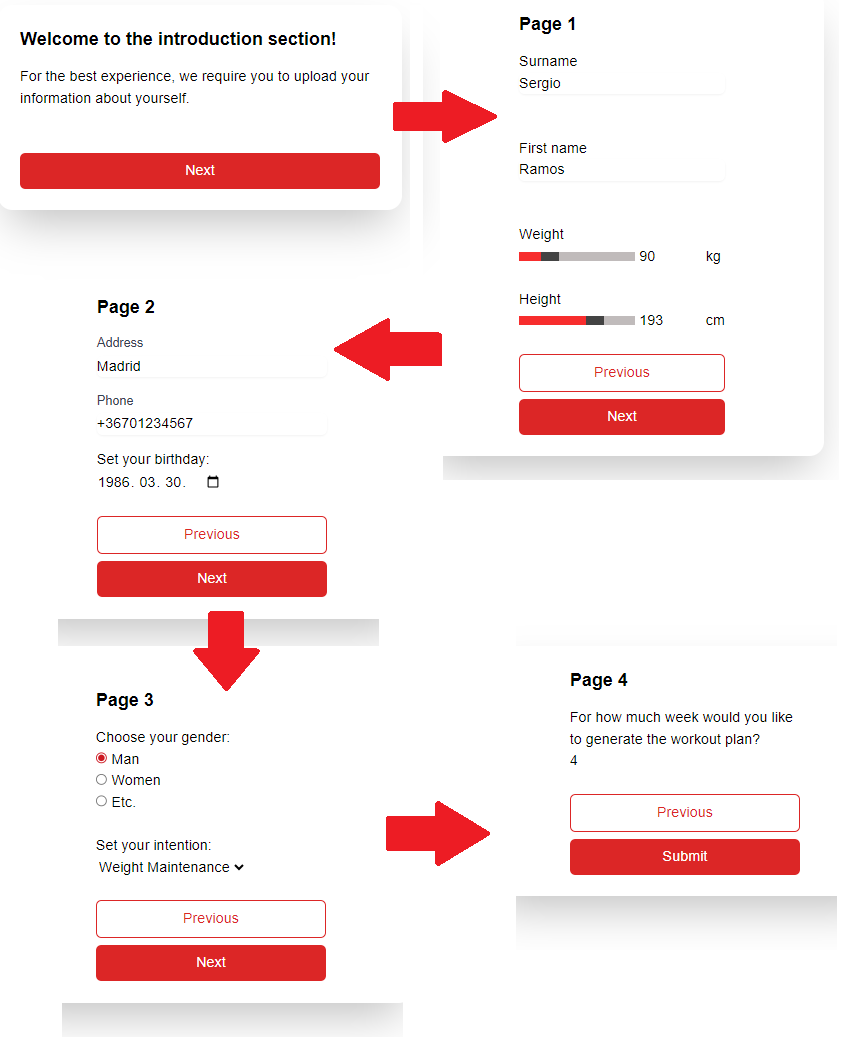


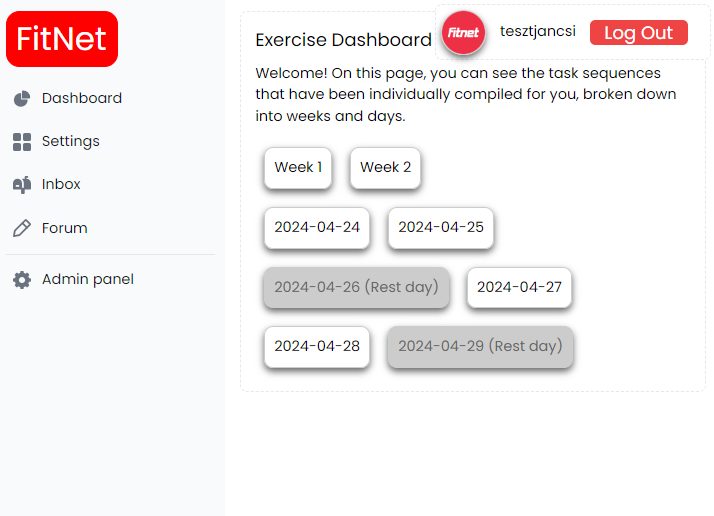
A főoldalon a jobb felső sarokban található egy bejelentkezés gomb. Kattintásra egy ablak jelenik meg, ahol bejelentkezhetsz, vagy regisztrálhatsz.

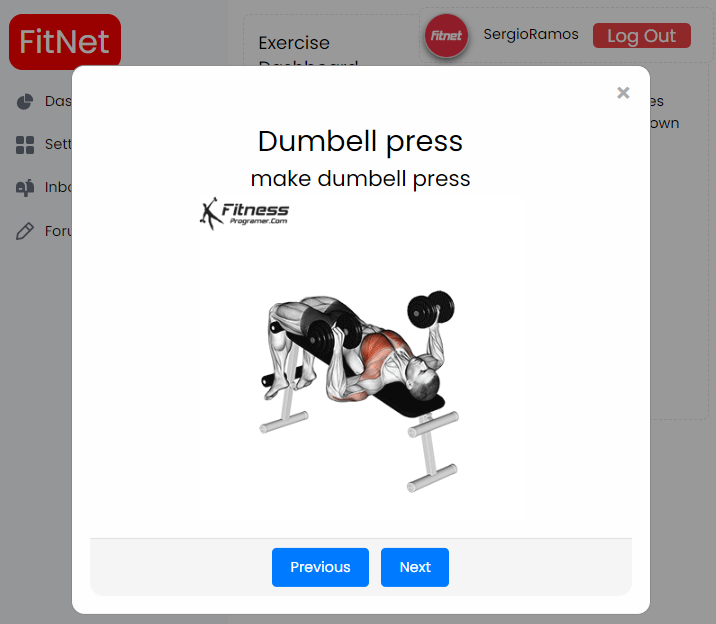




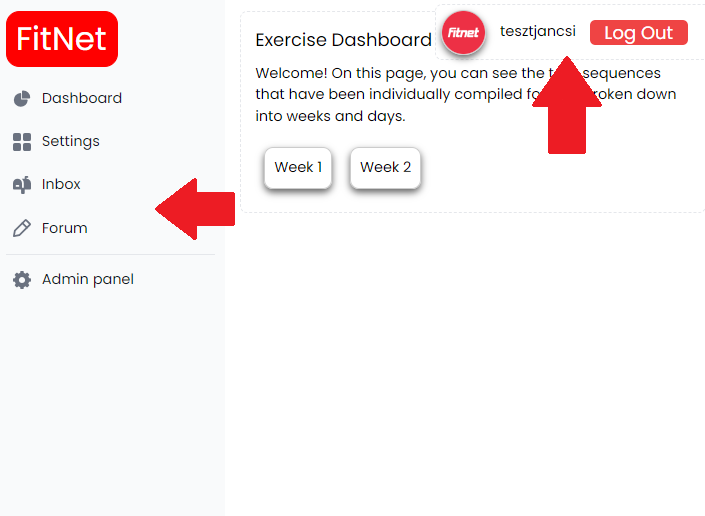
Az új regisztrációhoz egy egyszerű űrlapot kell kitöltened, majd átirányítunk a dashboard oldalra, ahol elérheted az edzésprogramokat. Itt láthatod a heteket (pl. week1), amire kattintva az edzésnapok megjelennek. Az edzésnapok közötti pihenőnapok külön jelölve vannak (rest day), melyekre nem tudsz kattintani.



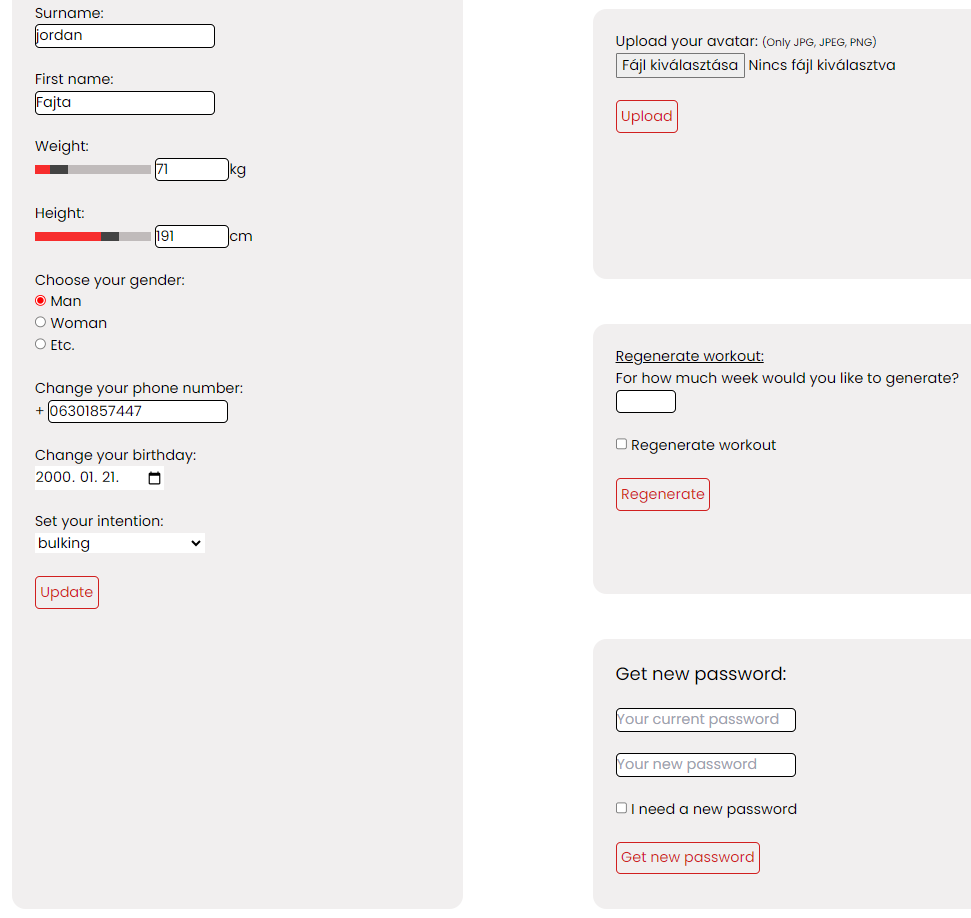




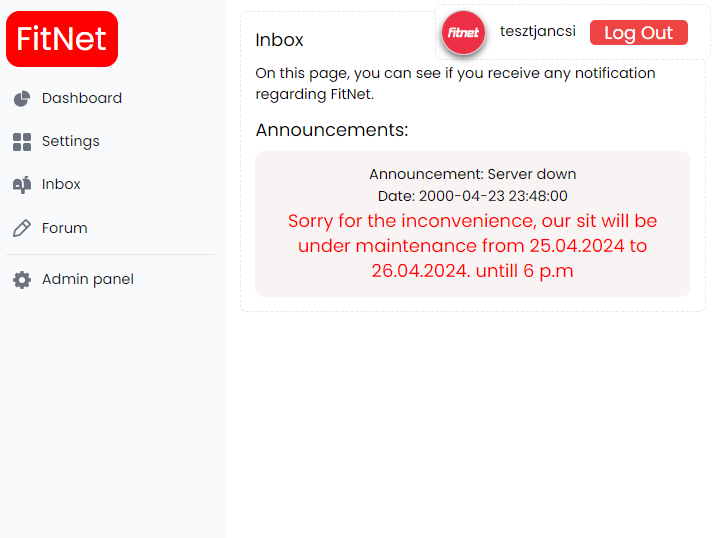
A baloldalon található a horizontális menü, ahol megtalálod a "Dashboard", "Settings", "Inbox", "Forum" lehetőségeket, valamint az admin jogokkal rendelkezők számára az "Admin panelt". A jobb oldalon láthatod a feltöltött profilképed (ha van), a felhasználóneved és a kijelentkezés gombot, melyek csak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



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



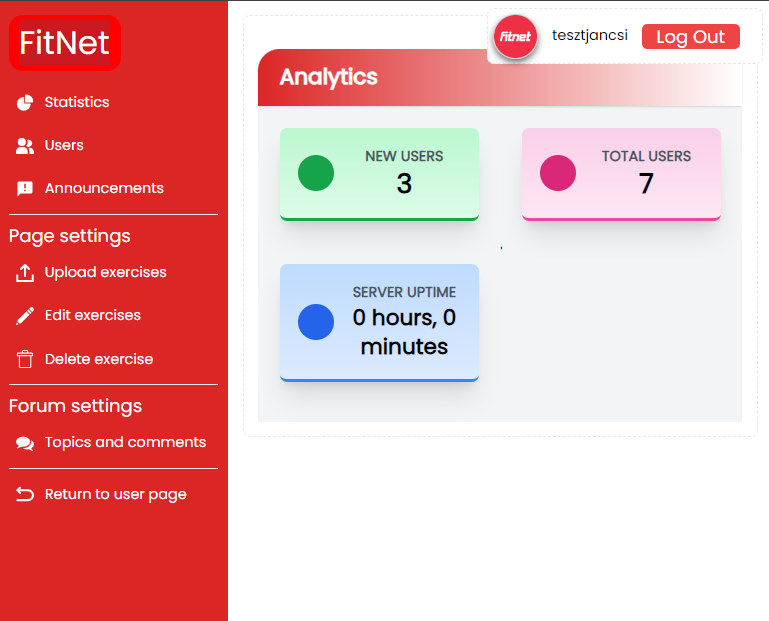
Az "Inbox" menüpontra kattintva megtekintheted az adminisztrátorok által küldött rendszerüzeneteket, például az oldal karbantartásáról.



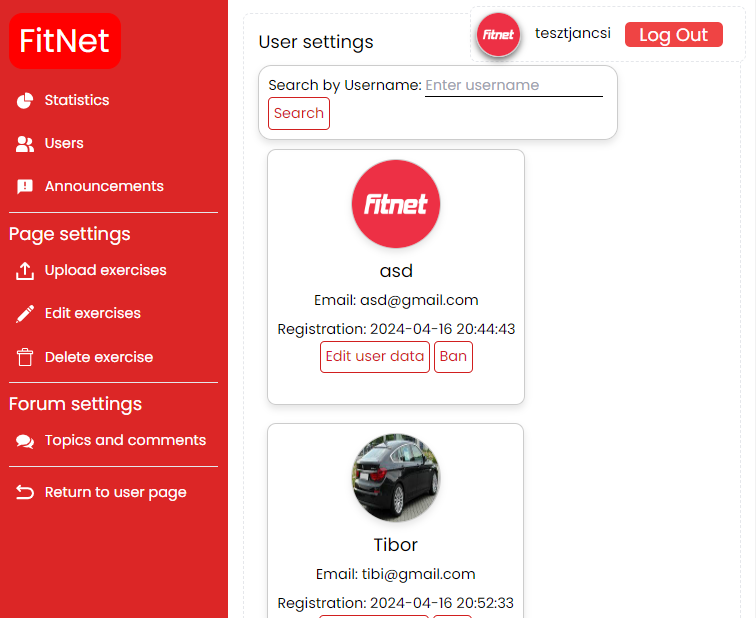
A "Forum" menüpontra kattintva témákat tudsz létrehozni vagy hozzászólni meglévő témákhoz. A téma kiválasztása után lehetőséged van törölni a saját témádat vagy kommentjeidet.



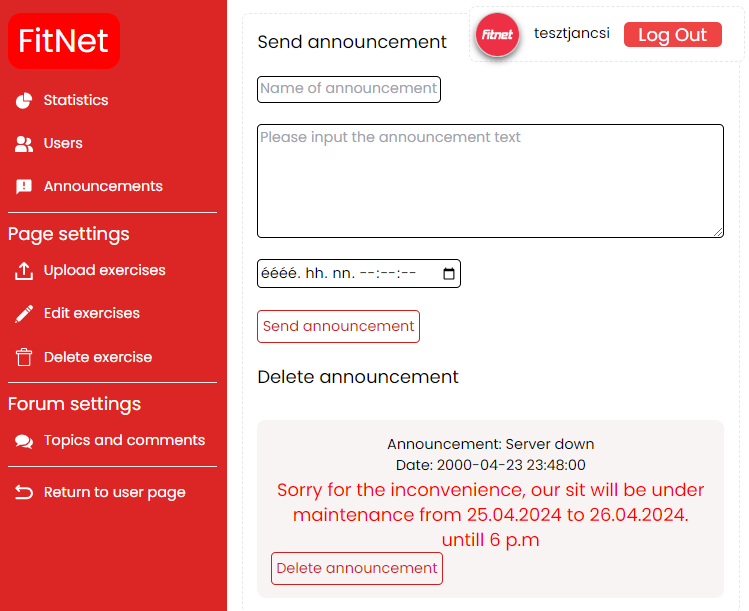
Ha adminisztrátori jogokkal rendelkezel, az "Admin panel" menüponton keresztül elérheted a Statistic, Users, Announcements, Upload exercises, Edit exercises, Delete exercise, Topic and comments, és Return to user page menüpontokkal rendelkező admin panelt. Az admin panel elsődleges oldal a Statistic, 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.

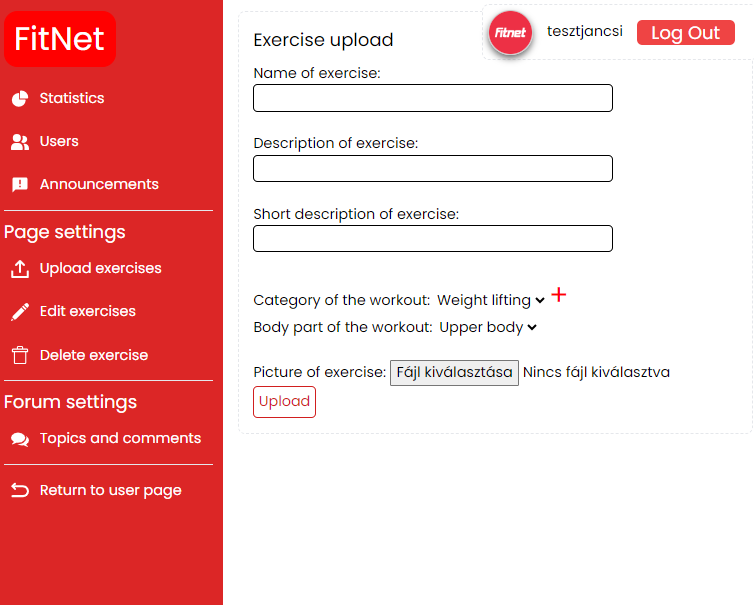


Az "Announcements" menüpont alatt tudod küldeni vagy törölni a rendszerüzeneteket.

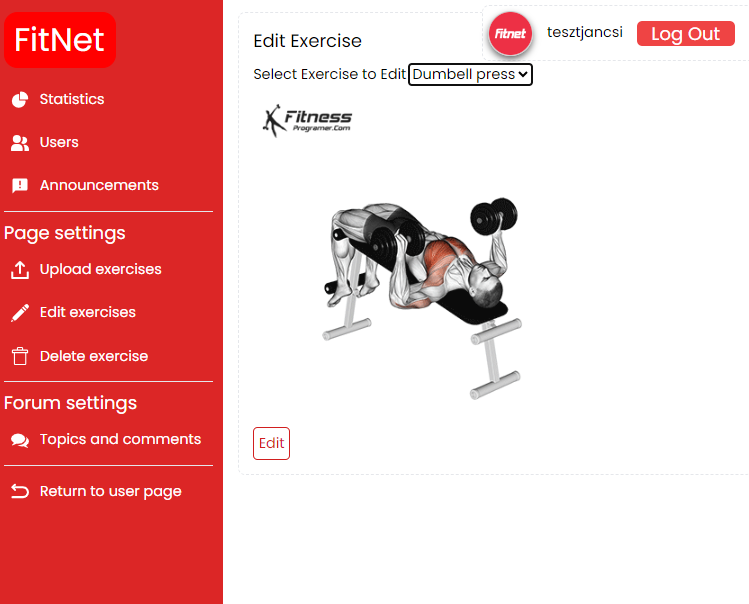


Az "Page settings" részen keresztül pedig feltölthetsz új gyakorlatokat az Upload exercises menüpontnál, szerkesztheted az Edit exercises-re kattintva vagy törölheted a meglévőket a Delete exercise menüponton keresztül.

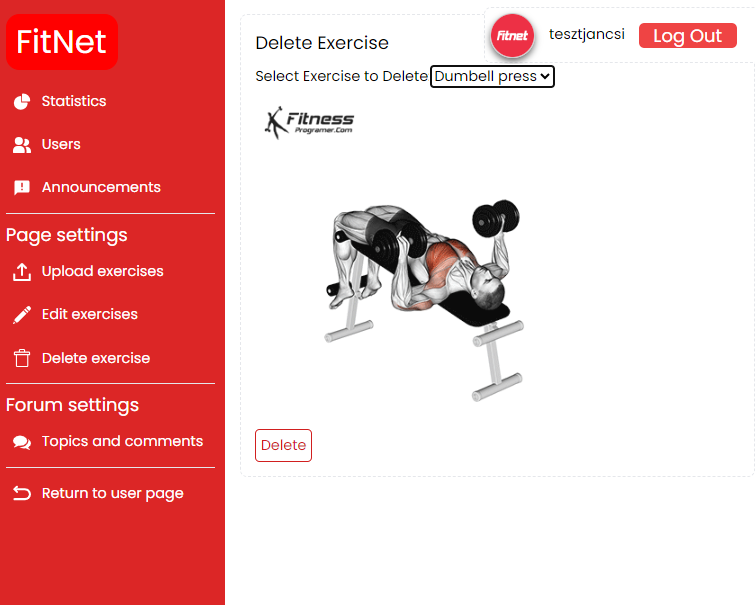
**Upload exercises**

****

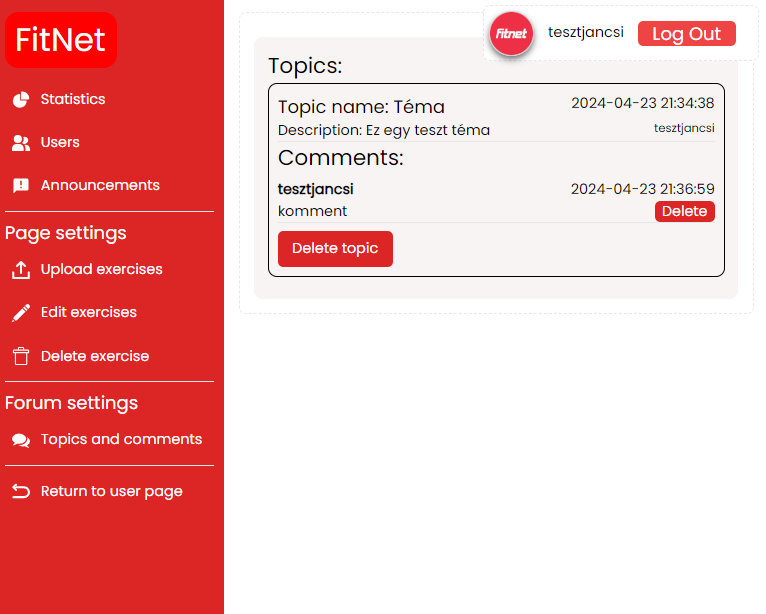
**Edit exercises**

****

**Delete exercise**

****

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.

****

# **Irodalomjegyzék**

AJAX irodalom – <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/>

# **Mellékletek**

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

Mappák:

dokumentacio – Tartalmazza a dokumentációt

adatbazis – Tartalmazza az feltöltött és üres adatbázis fájlait

web – Tartalmazza a weboldal forráskódjait, fájlait

server – Tartalmazza a Xampp telepítő fájlait

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