**Szabó Richárd és Híves Sebastian**

Edzés alkalmazás



Tartalom

[Bevezető 4](#_Toc132099189)

[1. Felhasznált technológiák 6](#_Toc132099190)

[1.1. HTML 6](#_Toc132099191)

[1.2. CSS 6](#_Toc132099192)

[1.3. JavaScript 6](#_Toc132099193)

[1.4. MySQL 6](#_Toc132099194)

[1.5. PHP 7](#_Toc132099195)

[2. Felhasznált szoftverek 7](#_Toc132099196)

[2.1. Visual Studio Code 7](#_Toc132099197)

[2.2. XAMPP 7](#_Toc132099198)

[2.3. GitHub 8](#_Toc132099199)

[3. Felhasználói dokumentáció 9](#_Toc132099200)

[3.1. Főoldal 9](#_Toc132099201)

[3.1.1. Navigációs menü 9](#_Toc132099202)

[3.1.2. Slideshow Carousel 10](#_Toc132099203)

[3.1.3. Edzésterv és étrendösszeállítás leírása 10](#_Toc132099204)

[3.1.4. Számláló (counter) 10](#_Toc132099205)

[3.1.5. Lábléc (footer) 10](#_Toc132099206)

[3.2. Regisztrációs oldal 10](#_Toc132099207)

[3.3. Bejelentkezés 11](#_Toc132099208)

[3.4. Kezdőlap a bejelentkezettek számára 11](#_Toc132099209)

[3.5. Saját profil 12](#_Toc132099210)

[3.6. Edzéstervek 12](#_Toc132099211)

[3.7. Chat 14](#_Toc132099212)

[3.8. Kliensek kezelése (edzők számára) 14](#_Toc132099213)

[3.9. Edzők kezelése (kliensek számára) 15](#_Toc132099214)

[4. Fejlesztői dokumentáció 16](#_Toc132099215)

[4.1. Adatbázis 16](#_Toc132099216)

[4.1.1. Szükséges információk 16](#_Toc132099217)

[4.1.2. Információk egyedekre osztása 16](#_Toc132099218)

[4.1.3. Információs elemek oszlopokká alakítása 17](#_Toc132099219)

[4.1.4. Kapcsolatok 18](#_Toc132099220)

[4.1.5. Táblák 19](#_Toc132099221)

[4.1.6. Adatbázis diagram 23](#_Toc132099222)

[4.2. Adatbázis kapcsolat 23](#_Toc132099223)

[4.3. Főoldal 24](#_Toc132099224)

[4.3.1. Navigációs menü 24](#_Toc132099225)

[4.3.2. Slideshow Carousel 25](#_Toc132099226)

[4.3.3. Edzésterv és étrendösszeállítás leírása 27](#_Toc132099227)

[4.3.4. Számláló(counter) 29](#_Toc132099228)

[4.3.5. Lábléc(footer) 30](#_Toc132099229)

[4.4. Chat 31](#_Toc132099230)

[4.4.1. Chat üzenetek kiírása 36](#_Toc132099231)

[4.4.2. Chat felület JavaScript része 37](#_Toc132099232)

[4.4 Kezdőlap 39](#_Toc132099233)

[4.4.1 Keresőmező 40](#_Toc132099234)

[4.4.2 Lista 41](#_Toc132099235)

[4.4.3 Extra funkció kliens típusú felhasználóknak 41](#_Toc132099236)

[4.4.4 tevTeljTorles() függvény 43](#_Toc132099237)

[4.5 sajatProfil.php – Saját profil adatai 43](#_Toc132099238)

[4.5.1 Lekérdezés 43](#_Toc132099239)

[4.5.2 Kimenet és profilkép összeállítása 44](#_Toc132099240)

[4.6 felhLista.php – (Felhasználó lista a kezdőlapon) 46](#_Toc132099241)

[4.7 tevekenysegek.php - Tevékenységek lekérése 48](#_Toc132099242)

[4.7.1 Lapozó 48](#_Toc132099243)

[4.7.2 Shorter függvény 49](#_Toc132099244)

[4.7.3 Tevékenységek lekérdezése és kimenet összeállítása 49](#_Toc132099245)

[4.8 tevModositas.php – Tevékenység módosítása/szerkesztése 51](#_Toc132099246)

[4.9 tevTorlese.php – Tevékenység törlése 51](#_Toc132099247)

[4.10 tevRogzitese.php – Tevékenység rögzítése 51](#_Toc132099248)

[4.11 edzesterv.php (Edzéstervek oldal) 52](#_Toc132099249)

[4.11.1 Edző típusú profil esetén 52](#_Toc132099250)

[4.11.2 Kliens típusú profil esetén 55](#_Toc132099251)

[4.12 Shorter függvény 57](#_Toc132099252)

[5. Tesztelés 58](#_Toc132099253)

[Összefoglalás 59](#_Toc132099254)

[Források 60](#_Toc132099255)

[Ábrajegyzék 61](#_Toc132099256)

# Bevezető

Csapatunk két főből áll, Híves Sebastian és Szabó Richárd. A témaválasztásunk abból a szempontból először nehéznek tűnt, hogy nagyon különböző az érdeklődési körünk. Egyikünk felvetette egy hangszerbolt webshopjának elkészítését, de amikor láttuk, hogy az osztályban már két csapat is webshop mellett döntött, ezt elvetettük, mivel egy egyedi munkát akartunk létrehozni. Ezért keresgéltünk az interneten, megnéztük a különböző ajánlásokat, és találtunk egy edzésprogrammal kapcsolatos ajánlást. Ez mindkettőnknek elnyerte a tetszését, mert ezzel meg is találtuk a közös érdeklődési pontot, hiszen a sport és az egészséges életmód mindkettőnket érdekel, és nagyon fontosnak tartjuk.

El is terveztünk gondolatban egy edző központot, de rájöttünk, ha ezt a gyakorlatban meg szeretnénk valósítani, az olyan mértékű anyagi befektetéssel járna, ami egy tőke nélküli kezdő vállalkozónak igen nehezen lenne kivitelezhető. Arra is gondoltunk, hogy sokaknak nincs is ideje személyesen ellátogatni sport központokba nyitvatartási időben étkezési és edzési tanácsot kérni, így jött az az ötletünk, hogy online sport centrumot hozzunk létre. Így a leendő ügyfelek bármikor kapcsolatba léphetnek az edzőikkel az online felületen, és tanácsot kérhetnek és kaphatnak, akár késő este munka után is, amikor épp idejük engedi a rohanó világunkban.

A weboldalunkon regisztrálhatnak edzők és vendégek egyaránt. A regisztráció során megadhatják adataikat, és feltölthetik profilfotójukat. Az edzők meghívhatnak vendégeket tanácsadásra, melyet a vendégek elfogadhatnak, vagy elutasíthatnak. Ugyanígy a vendégek is jelentkezhetnek egy edzőhöz tanácsadásra, amelyet az edző elfogadhat vagy visszautasíthat. Az edzők és vendégeik egy chat felületen beszélgethetnek egymással, ami privát beszélgetés, így más edzők és vendégek ezt nem látják. Az edzők bejegyzéseket hozhatnak létre, ahol az edzéstervet és étrend tanácsokat tudják megosztani személyre szabottan a vendégeikkel.

Dolgozatunk első részében a munka során felhasznált technológiákat és eszközöket mutatjuk be röviden, majd ezt követően a felhasználói dokumentációban részletesen ismertetjük az alkalmazás működését. A fejlesztői dokumentációban bemutatjuk az adatbázisunkat, és a tervezésének lépéseit.

A frontend és backend fejlesztéséből, azokat az elemeket emeltük ki, amelyek megvalósítására büszkék vagyunk. Akadtak nehézségek is, ezekről is írunk, valamint arról, hogyan oldottuk meg a felmerülő problémákat. Végezetül pedig értékeljük a közös munkánkat, amelyet nehéz lenne élesen kettéválasztani, mert a frontend megjelenésének munkálataiból mindketten kivettük a részünket. Talán a backend oldalt tudnánk jobban kettéválasztani. A chat megvalósítása Sebastian munkája, az oldalon a regisztráció, a beléptetés, az ehhez kapcsolódó tartalmak koordinálása pedig Richárd munkája.

# Felhasznált technológiák

## HTML

A HTML (HyperText Markup Language) egy leíró nyelv, amelyet weboldalak készítéséhez fejlesztettek ki. A W3C (World Wide Web Consortium) támogatásával vált internetes szabvánnyá. 1990 óta ezt a nyelvet használják a weboldalak elkészítéséhez. A HTML kód nyelv, nem programozási. És ez azt jelenti, hogy akár jegyzettömbben is tudnánk weboldalt írni a HTML segítségével. A HTML legfrissebb verziója a HTML 5, amely már 2004 óta használatban van. A számos rendelkezésre álló HTML jelölők sorából, körülbelül 20 félét használtunk fel a munkánk során.

## CSS

A CSS (Cascading Style Sheets) határozza meg a különböző elemek megjelenési tulajdonságait. A CSS segítségével tudjuk a weboldalakat formázni, pl. betűtípusokat, színeket állíthatunk be, de a legfontosabb az, hogy segítségével reszponzívvá tehetjük a weboldalunkat a média query-k használatával, vagyis mobil nézetre is optimalizálhatjuk az oldalainkat.

Egy stíluslapot több oldalhoz is hozzá tudunk kapcsolni. Ha megnyitunk egy stíluslapot a böngészőben, akkor az a böngésző gyorsítótárába tárolja el a stíluslapot. Így ha újra megnyitjuk a weboldalt, nem kell várakoznunk, hogy a CSS fájl betöltődjön, mert a gyorsítótár a böngészőben már eltárolta azt.

A munkánk során 4 stíluslapot készítettünk, ahol a weboldalak elrendezését főként flexbox-al oldottuk meg.

## JavaScript

A JavaScript egy népszerű programozási nyelv, amit a weboldalak dinamikus működtetéséhez használunk. A JavaScript lehetővé teszi a weboldalak számára, hogy interaktívvá váljanak, és képesek legyenek a felhasználó cselekvéseire megfelelően reagálva változtatni az oldal tartalmát, annak újratöltése nélkül. A JavaScripttel rendkívül sok dolog kivitelezhető Pl.: webes alkalmazások fejlesztése, webes játékok létrehozása, valamint adatok feldolgozása, ellenőrzése és elemzése.

## MySQL

A MySQL egy szerver-oldali adatbázis-kezelő rendszer, amely lehetővé teszi az adatok tárolását, lekérdezését és kezelését. A teljesen nyílt forráskódú LAMP (Linux-Apache-MySQL-PHP) összeállítás részeként költséghatékony, és könnyen beállítható megoldást ad a dinamikus webhelyek szolgáltatására. A MySQL már 1995-től elérhető. Nyílt forráskódú, és az adatok kezelésére is számos lehetőségeket biztosít. A legfrissebb verzió már több mint 1 éve, 2021-ben került közzétételre. Az adatbázisokhoz használhatunk különböző parancssori eszközöket, vagy grafikus felületű eszközöket, mint Pl.: a phpMyAdmin, amely PHP nyelven lett írva. A MySQL támogatja a különböző adatbázis-kezelő rendszereket, így lehetővé teszi az adatok importálását, és exportálását.

## PHP

A PHP már a szerveroldali programozási nyelv. A PHP segítségével tudjuk összekötni az adatbázist a weboldallal. Képes akár nagyméretű adatbázis alkalmazások kezelésére is, melynek segítésével egy statikus weboldalt dinamikus webes alkalmazássá alakíthatunk. Segítségével beolvashatunk az adatbázisból adatokat, módosíthatjuk azokat, új adatokat vihetünk fel és törölhetünk. Ezek az ún. CRUD (Create Read Update Delete) műveletek.  Továbbá kontrolálhatjuk vele a munkameneteket és felhasználók kezelését, hogy pl. egy az oldalra belépett felhasználó milyen adatokat érjen el, milyen jogosultságokat kapjon.

# Felhasznált szoftverek

## Visual Studio Code

A Visual Studio Code egy ingyenes kódszerkesztő, melyet a Microsoft fejleszt Windows, Linux és MacOS rendszerekre. Számos programozási nyelvet támogat (Pl.: HTML, CSS, JavaScript, PHP, Python stb.) és mára eléggé népszerűnek bizonyul a fejlesztők körében. Támogatja a hibakeresőket és képes az intelligens kódkiegészítésre az IntelliSense segítségével.

Rendkívül megkönnyíti és meggyorsítja a programozás folyamatát a hozzáadható moduloknak köszönhetően, amelyekből telepítettük az Emmet-et, Live Server-t, Open PHP-t, PHP Inthelephense-t, Auto Rename Tag-et.

## XAMPP

Mivel helyi gépen dolgoztunk így szükségünk volt egy webkiszolgálóra, az XAMPP-ot választottuk, amely tartalmazza az Apache webszervert, MySQL adatbázis-kezelőt és PHP értelmezőt. A XAMPP egy platformfüggetlen webszerver-szoftvercsomag, ami a webes alkalmazások készítésében, tesztelésében és futtatásában nyújt segítséget. Előnye, hogy egyetlen csomagban tartalmazza az összes eszközt, amely szükséges a webes alkalmazások készítéséhez, ezzel rengeteg telepítést és időt spórolhat meg.

## GitHub

A GitHub Inc. egy egyesült államokbeli nemzetközi vállalat, amely a Git segítségével szoftverfejlesztési verziókövetés-szolgáltatást nyújt. 2018-ban a Microsoft leányvállalata lett. Saját funkcióin felül a Git elosztott verziókövetését és forráskódkezelését (SCM) teszi elérhetővé. Hozzáférés-kezelést és számos együttműködési funkciót nyújt, mint például bugkövetés, szolgáltatáslekérés, feladatkezelés, valamint ún. wikiket minden projekthez.

A mi projektünknek Richárd hozott létre a GitHub-on egy repository-t, amelyhez hozzá adta Sebastiant. Így mindketten hozzá férünk, és ahogyan dolgoztunk a programunkon rendszeresen ún. commit-ot írtunk, amely mentette a változásokat a Github-ra is, mivel összekötöttük a fejlesztői környezettel. Így tudtuk egymás munkáját is nyomon követni, akár otthonról is.

# Felhasználói dokumentáció

A webes alkalmazásunk 9 különböző weblapból áll, melyeket a következő pontokban mutatunk be részletesen.

## Főoldal

A főoldalra belépve megtalálunk minden fontos adatot, ami a webalkalmazással kapcsolatos.

#### ábra Főoldal

### Navigációs menü

A weblap tetején található a navigációs menü, ami a weboldal címet tartalmazza (ez esetben a ShineGym&Fit). A navigációs menü jobb oldalán pedig a Főoldalra visszavezető link, valamint a Bejelentkezés és a Regisztráció fül.

### Slideshow/Carousel

A weboldalon található egy Slideshow/Carousel, ami automatikusan váltogat három képet, a képek két oldalán két nyíl is látható, amellyel lapozni is tudunk a fejléc képek között. Itt foglal helyet középen lent a weboldalra látogatók üdvözlésére szolgáló leírás.

### Edzésterv és étrendösszeállítás leírása

Lejjebb görgetve a webalkalmazás ajánlása látható, ahol szintén felajánljuk a felhasználónak, a regisztráció lehetőségét. Továbbá ismertetjük, hogy miért érdemes az alkalmazásra regisztrálni, és mik azok a szolgáltatások, amit a vendégeinknek biztosítunk. Alatta az étrendösszeállítás leírása foglal helyet, ami arról ad tájékoztatást, hogy az edzésterv mellett, legmegfelelőbb étrendet is megkaphatják az alkalmazásban a regisztráció után, melyet az általuk választott személyi edző készít el részükre.

### Számláló (counter)

A weboldal főoldalának alsó szekciójában elhelyeztünk egy számlálót is, ami az eddigi regisztrációkat, pozitív visszajelzéseket és a Facebook követőket mutatja. Mivel az oldalunk még nem szerepelt éles környezetben és ennél fogva követőink sincsenek, így csak random számokat jelenítettünk meg, a dizájn kedvéért. Viszont, ha a jövőben ténylegesen lesznek az oldalon látogatók, és követők a közösségi oldalakon, akkor ezek számát adatbázisba beolvashatjuk, és onnan frissítve megjeleníthetjük.

### Lábléc (footer)

A weboldal alján a lábléc található, ami a kapcsolati adatokat tartalmazza, melyeken keresztül a felhasználó, akár személyesen is felveheti a kapcsolatot az edzőkkel. Itt láthatóak a legnépszerűbb közösségi média logók is, amelyekre nem állítottunk be horgonyt, mivel nem hoztunk létre ezeken saját oldalakat. Szintén itt helyeztük el a szerzői jogi felhasználói feltételek is.

## Regisztrációs oldal

Az oldalra látogató új vendégek a Regisztráció menüpontra kattintva tudnak regisztrálni a weboldalra. A regisztrációra kattintás után a kliens ki tudja választani, hogy edzőként vagy kliensként szeretne-e regisztrálni. A profilképet nem kötelező megadni, de a felhasználó kedve szerint feltölthet egyet ha szeretne. A többi mező megadása kötelező, amit a csillag is jelez. Például a Vezetéknév, Keresztnév, E-mail, Jelszó, amit meg is kell erősíteni, és a Telefonszám, ami csak edzői profil regisztrációja esetén kötelező. Található egy bemutatkozó mező is. Az edzőnek kötelező megadnia a végzettségét, szakmai tapasztalatát. A kliensnek nem kötelező, de megadhatja az elérendő célt, ha sérülése van, akár valamilyen betegsége, azt ide felviheti. Ha sikeresen beregisztráltunk, már jelentkezhetünk is be.

#### ábra Regisztrációs felület

## Bejelentkezés

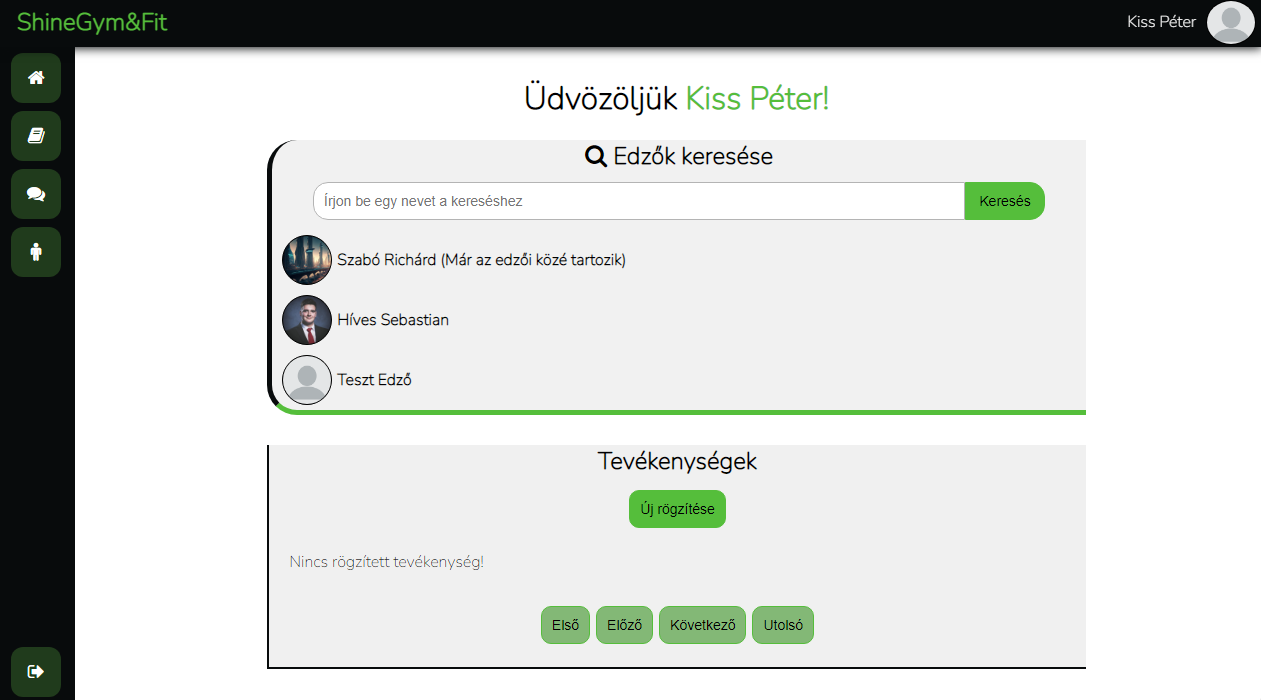


#### ábra Bejelentkezés felülete

A bejelentkezés menüpontot el tudjuk érni a navigációs menüből is, vagy a sikeres regisztráció után a lap tetején felugró üzenetből. A bejelentkezésnél a felhasználónak csak az e-mail címét és a jelszavát kell megadnia.

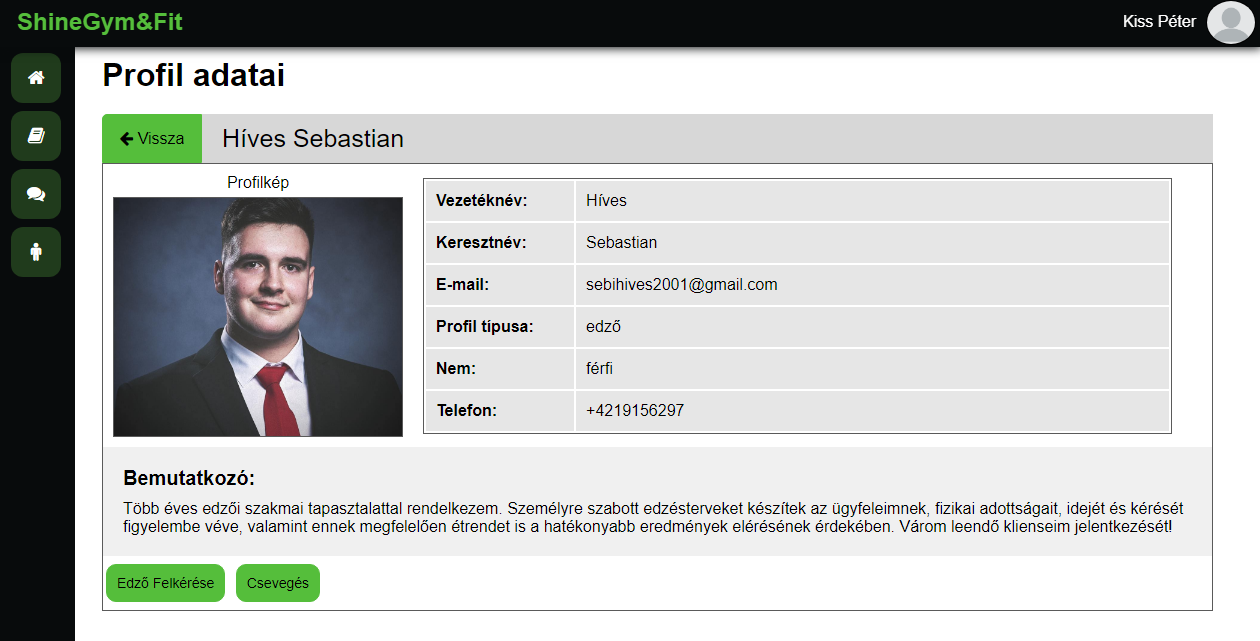
## Kezdőlap (Bejelentkezés után)

Bejelentkezés után ez a felület fogad minket. Ha kliens profillal vagyunk bejelentkezve az üdvözlő üzenet alatt megjelenik egy keresősáv és egy görgethető lista, amelyen az összes regisztrált edző szerepel. A keresőmezővel pedig név alapján kereshetünk a listában. Ha edző profillal vagyunk belépve akkor pedig az összes regisztrált kliens jelenik meg a listában.



#### ábra Kezdőlap

Ha rákattintunk egy személyre a listában, akkor megtekinthetjük a választott személy adatlapját. Itt minden szükséges információt megtalálunk róla pl.: e-mail cím, telefonszám vagy a bemutatkozó szöveg.



#### ábra Felhasználó adatlapja

A felkérés gomb segítségével felkérhetjük a kiválasztott klienst vagy edzőt. A Csevegés gombra kattintva privát üzenetben beszélhetünk a választott személlyel.

## Saját profil

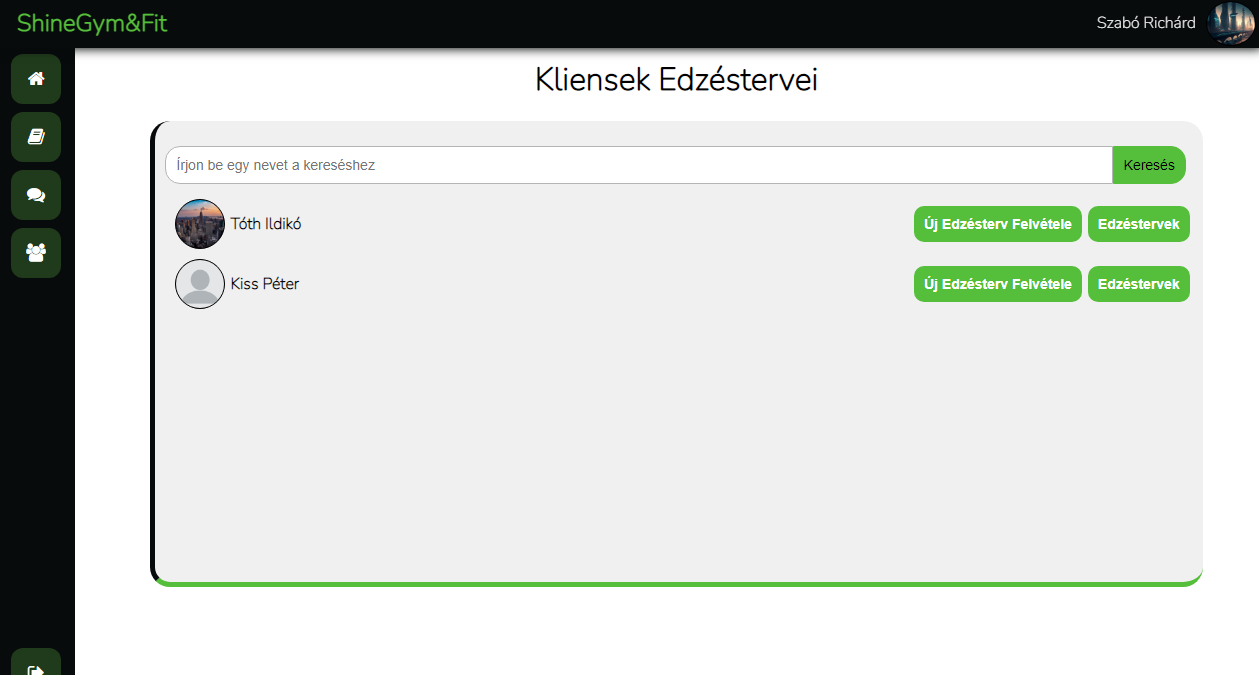
A felhasználók a felső navigációs sávon szereplő profiljukra kattintva tudják az adataikat megtekinteni, mint a feltöltött fotó, név, e-mail cím, profil típusa, nem és a telefonszám. Itt lehetőség van ezen adatok szerkesztésére is a jobb felső sarokban lévő gombra kattintva.



#### ábra Saját profil

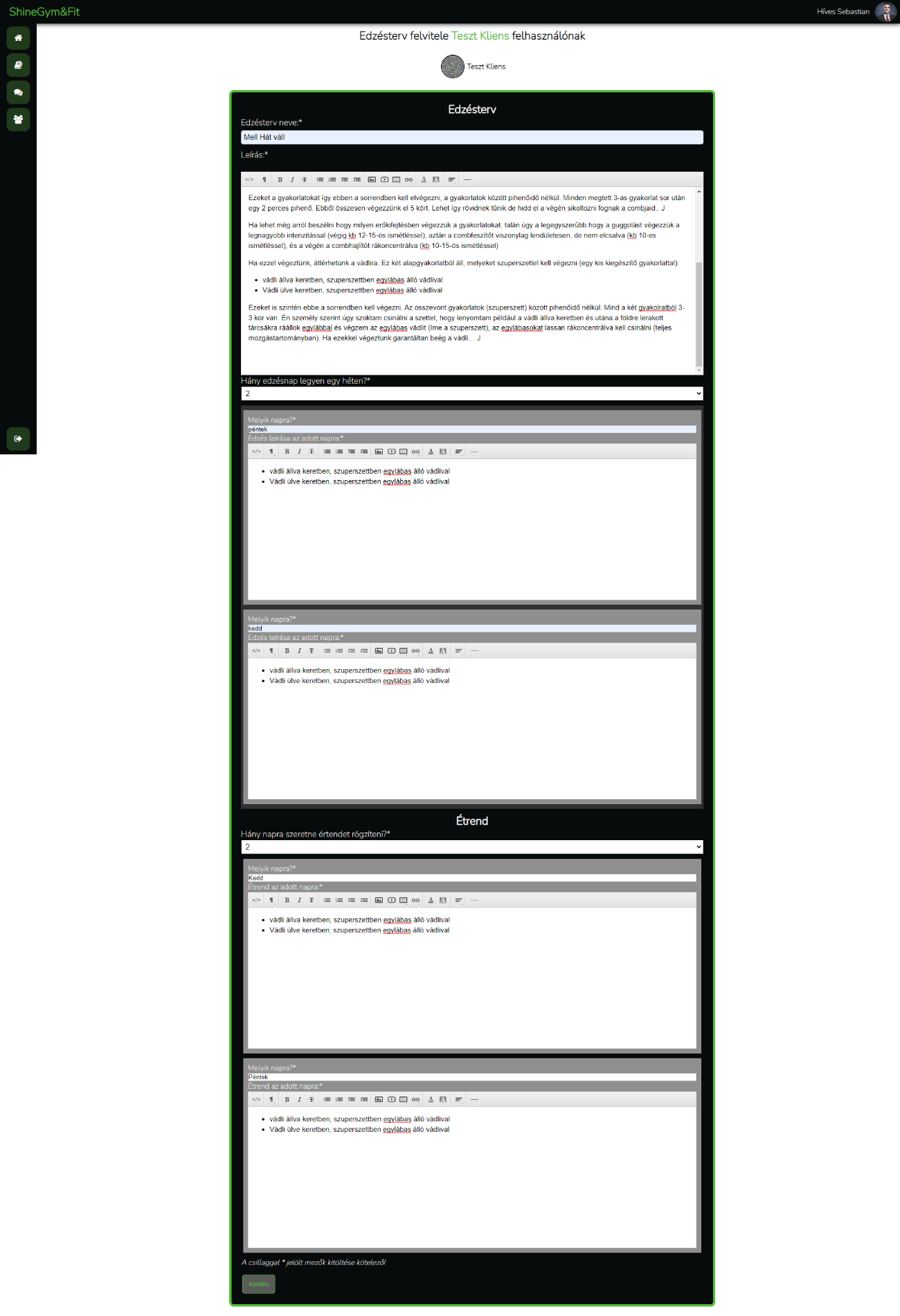
## Edzéstervek

Miután küldtünk vagy kaptunk egy felkérést és az el lett fogadva, az Edzéstervek menüpontban az edző fel tudja venni a kliens edzéstervét és étrendjét. Itt az edző ki tudja választani, hogy melyik klienségnek szeretne edzéstervet vagy étrendet rögzíteni, illetve megtekintheti a korábbi edzésterveit is, amiket ő írt korábban (tehát ha más edző írt edzéstervet vagy étrendet a kliensek, azt nem fogja tudni megtekinteni).



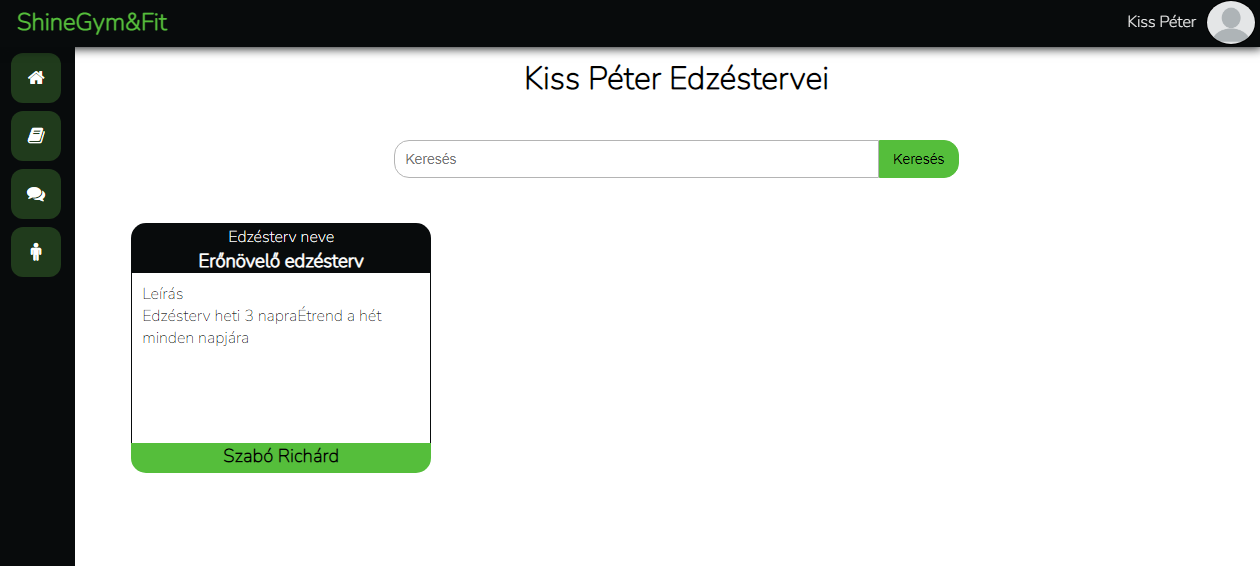
#### ábra Edzéstervek oldal (edző profil)

Az edző megadja, hogy mi legyen az edzésterv neve, leírása, melyik napokra szól az edzésterv, az étrendnél pedig szintén meg tudja adni, hogy melyik napra szóljon az étrend és mi legyen az étrend az adott napra. Amint az az alábbi képen is látható, olyan funkcióval is elláttuk a szövegbeviteli mezőt, hogy az edző szükség szerint ki tudjon emelni szövet, felsorolást tudjon létre hozni, hivatkozhat linkekre, és képet is tud szükség esetén a gyakorlatokról feltölteni.



#### ábra Edzésterv felvitele

Kliens profil esetén itt az adott kliens edzői által felvitt edzéstervek láthatóak, amelyekre rákattintva tudja megtekinteni részletesebb formában az adott edzéstervet.



#### Edzéstervek oldal (kliens profil)

## Chat

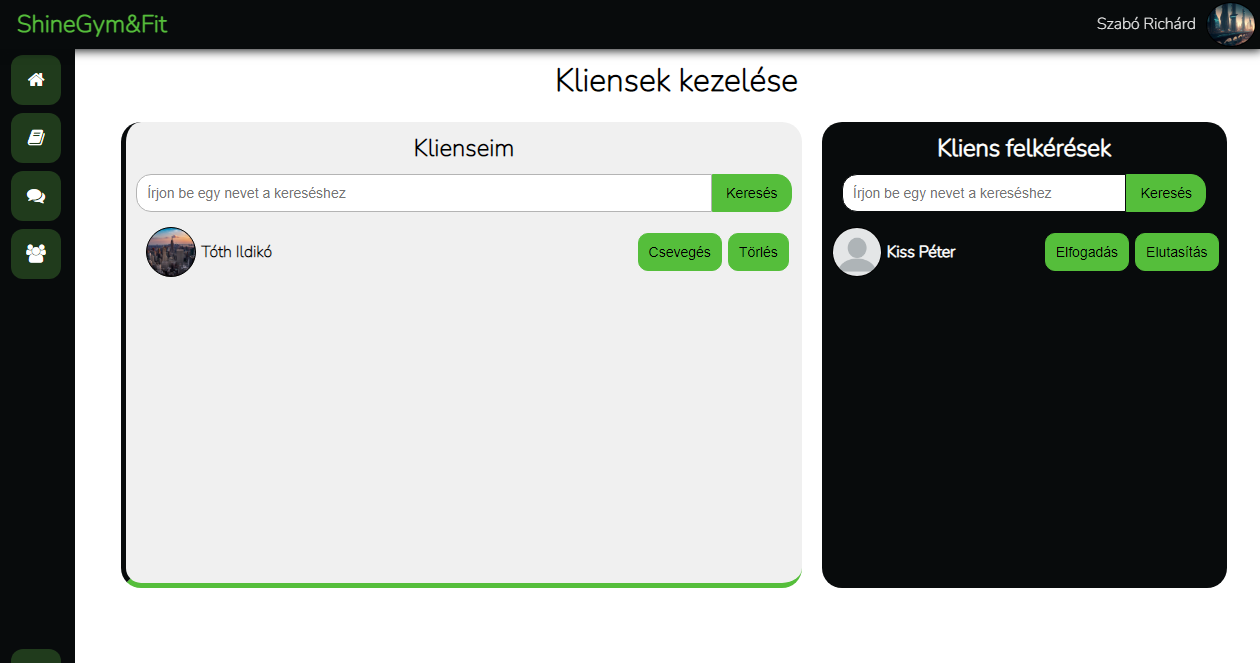
A Chat menüpontot megnyitva kliensek és edzők tudnak egymással privát üzenetben kommunikálni. Bal oldalon választható ki, hogy kivel szeretnénk csevegni, ha kiválasztottuk és rákattintottunk a jobb oldalon meg is jelenik az adott felhasználóhoz tartozó chat felület. Betöltődnek a korábban küldött üzenetek (ha van), megjelenik az üzenet írásához használható mező és a küldés gomb.



#### ábra Chat

## Kliensek kezelése (edzők számára)

Ha már valaki felkért bennünket azt a jobb oldali listában fogadhatjuk, illetve utasíthatjuk el a felkérését. Bal oldalon a már meglévő klienseket kezelhessük itt van lehetőség a kapcsolat felbontására is a törlés gombra kattintva. A kapcsolat törlésénél törlődnek az ahhoz tartozó edzéstervek és étrendek is.



#### ábra Kliensek kezelése

## Edzők kezelése (kliensek számára)

Az edzők kezelése pedig a kliensek számára szolgál, mivel edző edzőt nem kezelhet. Ennek értelmében pedig a felület ugyan az, csak edzőkre specifikálva. Kliens típusú felhasználónak is lehetősége van törölni az edzőit, ha már nem szeretne több edzéstervet/étrendet kapni az illetőtől, vagy egyszerűen nem volt megelégedve azzal.



#### ábra Edzők kezelése

# Fejlesztői dokumentáció

## Adatbázis

Elsőkörben az adatbázisunk működését és tervezési lépéseit mutatjuk be részletesen.

### Szükséges információk

Első körben összeszedtük, hogy milyen adatokat és hol szeretnénk majd tárolni az adatbázisunkban.

**Felhasználók adatai**: Mindenképpen szükséges eltárolnunk a felhasználók alapvető adatait, amelyek az alkalmazás működéséhez szükségesek (ilyenek lehetnek például: vezetéknév, keresztnév, nem, e-mail cím, regisztrációkor megadott jelszó, profilkép, telefonszám). Ezen kívül szükséges még egy bemutatkozó szöveg és a profil típusának eltárolása, mivel azt tervezzük, hogy 2 profil típus (edző vagy kliens) közül választhatnak majd a felhasználók.

**Üzenetek adatai**: Mivel egy edző profillal belépve lehetőségünk van privát üzenetben csevegni az összes regisztrált klienssel, és egy kliens profillal belépve lehetőségünk van privát üzenetben csevegni az összes regisztrált edzővel - el kell tárolnunk, hogy ki küldte az üzenetet, kinek küldte és magát az üzenet tartalmát (üzenetet).

**Edző-Kliens kapcsolatok**: Minden regisztrált felhasználónak lehetősége van az ő profiljával ellentétes típusú profillal edző-kliens kapcsolatot kezdeményezni (tehát edző csak klienssel és kliens csak edzővel veheti fel a kapcsolatot). Ez alapján el kell majd tárolnunk, hogy ki kezdeményezte a kapcsolatot, kivel, a felkérés (kapcsolat kezdeményezése) dátumát, elfogadta-e a kiválasztott személy és amint elfogadta eltároljuk még a kapcsolat kezdetének dátumát.

**Edzésterv adatai**: Az előbb említett kapcsolatok lényege, hogy az edző edzéstervet és étrendet küldhessen a kliensének. Tehát egy terv esetében eltároljuk majd az edző által megadott terv nevét és annak leírását. Az terv edzés részénél eltároljuk majd, hogy melyik napra szól az adott edzés és az adott edzés leírását (teendőket, feladatokat). Az edző részhez hasonlóan az étrendben is eltároljuk, hogy melyik napra szól és mi az adott napra az étrend.

### Információk egyedekre osztása

**Felhasználók egyed:** Úgy döntöttünk, hogy a felhasználók adatait 1 táblában tároljuk el, mert szinte teljesen egyforma tulajdonságokat tárolunk el edző és kliens profilról is. Tehát a felhasználók táblában fog szerepelni a vezetéknév, keresztnév, nem, e-mail cím, jelszó, profil típusa, profilkép, bemutatkozó szöveg és a telefonszám.

**Üzenetek**: Az üzenetek táblában fog szerepelni, hogy ki küldte az üzenetet, kinek küldte és magát az üzenet tartalmát (üzenetet).

**Edző-Kliens kapcsolatok:** Ez a tábla tartalmazza, hogy ki kezdeményezte a kapcsolatot, kivel, a felkérés dátumát, elfogadta-e a kiválasztott személy és a kapcsolat kezdetének dátumát.

**Terv egyed:** Itt tárjuk az edző által megadott terv nevét és annak leírását.

**Edzés egyed:** Ez a tábla tartalmazza az edzéstervet, itt tároljuk az edzés leírását (feladatok) és hogy melyik napra szól az adott edzés.

**Étrend egyed:** Ez a tábla az étrendet tartalmazza, tehát hogy melyik napra szól az étrend és az étrend leírását az adott napra.

### Információs elemek oszlopokká alakítása

|  |
| --- |
| Felhasználók |
| felhasználó\_id |
| vezetéknév |
| keresztnév |
| nem |
| e-mail |
| jelszó |
| profil típus |
| profilkép |
| bemutatkozó |
| telefon |

|  |
| --- |
| Üzenetek |
| üzenet\_id |
| kimenő\_id |
| bejövő\_id |
| üzenet |

|  |
| --- |
| Tevékenységek |
| tev\_id |
| felhasználó\_id |
| dátum |
| leírás |

|  |
| --- |
| EdzőKlienskapcs |
| kapcs\_id |
| kuldő\_id (kapcsolat kezdeményezőjének azonosítója [felkérő/küldő]) |
| fogadó\_id (annak azonosítója akivel kezdeményezni szeretné a kapcsolatot [fogadó]) |
| elfogadva (elfogadva[true] vagy elutasítva[false]) |
| felkérés dátuma |
| kapcsolat kezdetének dátuma |

|  |
| --- |
| Terv |
| edzésterv\_id |
| neve |
| leírása |
| kapcs\_id |

|  |
| --- |
| Edzéstervek |
| edzés\_id |
| nap |
| edzésterv (edzés leírása az adott napra) |
| edzésterv\_id |

|  |
| --- |
| Étrendek |
| étrend\_id |
| nap |
| étrend leírása az adott napra |
| edzésterv\_id |

### Kapcsolatok

Üzenetek N:1 Felhasználók

Egy felhasználóhoz több üzenet is tartozhat és egy üzenet pedig 2 felhasználóhoz tartozik (egy küldőhöz és egy fogadóhoz).

Tevékenységek N:1 Felhasználók

Egy felhasználóhoz több rögzített tevékenység, de egy adott tevékenység csak 1 felhasználóhoz tartozhat.

Felhasználók 1:N EdzőKliensKapcs

Egy felhasználó több Edző-Kliens kapcsolatba tartozhat, viszont egy Edző-Kliens kapcsolatban csak egyszer szerepelhet egy adott felhasználó (vagy küldőként, vagy fogadóként).

EdzőKliensKapcs 1:N Terv

Egy Edző-Kliens kapcsolathoz több terv, de egy terv csak egy adott kapcsolathoz tartozhat.

Terv 1:N Edzéstervek

Egy tervhez több edzésterv, de egy edzésterv csak egy adott tervhez tartozhat.

Terv 1:N Étrendek

Egy tervhez több étrend, de egy étrend csak egy adott tervhez tartozhat.

### Táblák

|  |  |  |
| --- | --- | --- |
| felhasznalok | | |
| **mező neve** | **típusa** | **leírás** |
| felhasznalo\_id | int(11) | felhasználó azonosítója |
| vnev | varchar(50) | vezetéknév |
| knev | varchar(50) | keresztnév |
| email | varchar(255) | e-mail cím |
| jelszo | varchar(255) | jelszó |
| profil\_tipus | varchar(6) | profil típusa (edző/kliens) |
| kep | varchar(255) | kép neve, kiterjesztése |
| nem | tinyint(1) | nem (true esetén férfi, false esetén nő) |
| bemutatkozo | text | bemutatkozó szöveg (HTML kód a szöveg formázása miatt) |
| telefon | varchar(16) | telefonszám (külföldi és belföldi) |



#### ábra Felhasználók tábla

|  |  |  |
| --- | --- | --- |
| uzenetek | | |
| **mező neve** | **típusa** | **leírás** |
| uzenet\_id | int(11) | üzenet azonosítója |
| kimeno\_id | int(11) | kimenő azonosító (felhasználó kimenő üzenetei) |
| bejovo\_id | int(11) | bejövő azonosító (felhasználó bejövő üzenetei) |
| uzenet | text | üzenet tartalma |



#### ábra Üzenetek tábla

|  |  |  |
| --- | --- | --- |
| tevekenysegek | | |
| **mező neve** | **típusa** | **leírás** |
| tev\_id | int(11) | tevékenység azonosítója |
| felhasznalo\_id | int(11) | felhasználó azonosítója |
| datum | date | dátum (mikor rögzítették a tevékenységet) |
| leiras | text | tevékenységek leírása (HTML kód a szöveg formázása miatt) |



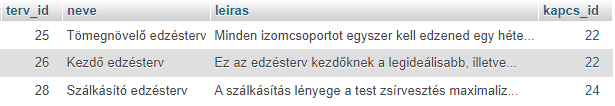
#### ábra Tevékenységek tábla

|  |  |  |
| --- | --- | --- |
| edzoklienskapcs | | |
| **mező neve** | **típusa** | **leírás** |
| kapcs\_id | int(11) | kapcsolat azonosítója |
| kuldo\_az | int(11) | küldő azonosítója (kérelem küldője) |
| fogado\_az | int(11) | fogadó azonosítója (kérelem fogadója, felkért) |
| elfogadva | tinyint(1) | kérelem állapota  elfogadva (true)  függőben (false) |
| felkeres\_datuma | datetime | felkérés küldésének dátuma |
| kapcs\_kezdete | datetime | kapcsolat kezdetének dátuma |



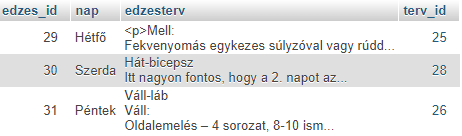
#### ábra Edző-Kliens kapcsolatok tábla

|  |  |  |
| --- | --- | --- |
| terv | | |
| **mező neve** | **típusa** | **leírás** |
| terv\_id | int(11) | terv azonosítója |
| neve | varchar(40) | neve (edző által kitalált) |
| leiras | text | leírás pl.: edző által leírt tanácsok, terv rövid bemutatója… stb.  (HTML kód a szöveg formázása miatt) |
| kapcs\_id | int(11) | kapcsolat azonosítója |



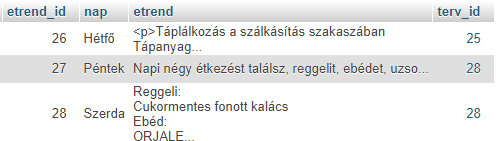
#### ábra Terv tábla

|  |  |  |
| --- | --- | --- |
| edzestervek | | |
| **mező neve** | **típusa** | **leírás** |
| edzes\_id | int(11) | edzésterv azonosítója |
| nap | varchar(10) | nap (a hét melyik napjára szól az edzésterv) |
| edzesterv | text | edzésterv leírása (HTML kód a szöveg formázása miatt) |
| terv\_id | int(11) | terv azonosítója |



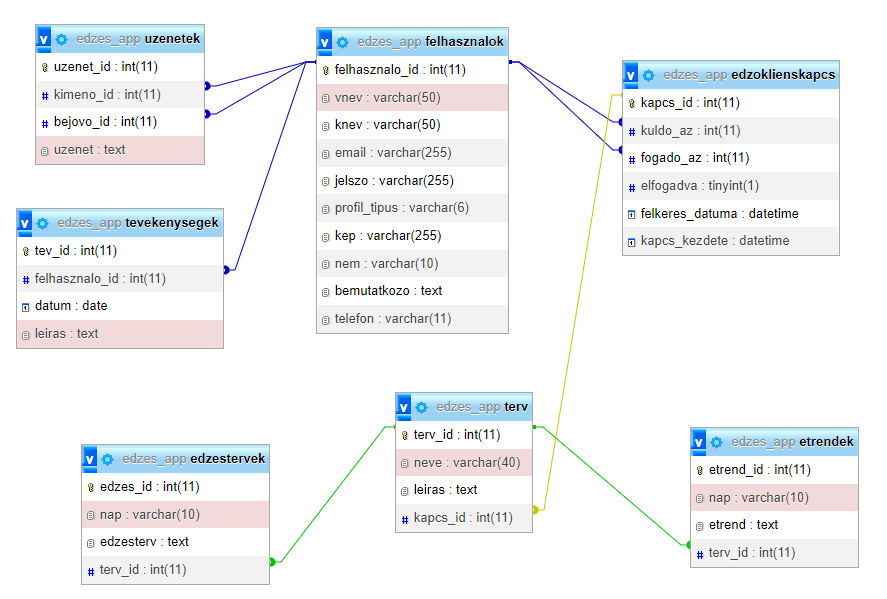
#### ábra Edzéstervek tábla

|  |  |  |
| --- | --- | --- |
| etrendek | | |
| **mező neve** | **típusa** | **leírás** |
| etrend\_id | int(11) | étrend azonosítója |
| nap | varchar(10) | nap (a hét melyik napjára szól az étrend) |
| etrend | text | étrend leírása (HTML kód a szöveg formázása miatt) |
| terv\_id | int(11) | terv azonosítója |



#### ábra Étrendek tábla

### Adatbázis diagram



#### ábra Adatbázis diagram

## Adatbázis kapcsolat

Külön fájlban (kapcsolat.php) hoztuk létre a kapcsolatot az adatbázis és a weboldal között, hogy ne kelljen minden egyes oldalon megírni, mert ezáltal redundánssá válik a kód és minden oldalon tesztelni kellene, hogy valóban létrejön-e a kapcsolat.

$dbconn = @mysqli\_connect("localhost", "root", "", "edzes\_app");

A @mysqli\_connect() (a @ szimbólum hibák figyelmen kívül hagyására szolgál) függvény segítségével létrehozzuk az adatbázis kapcsolatot, paraméterül adjuk az adatbázis szerver elérési útvonalát (localhost), felhasználónevet (root), jelszót (jelen esetben ez üres) és az adatbázis nevét, amihez kapcsolódni szeretnénk(edzes\_app).

//ellenőrzés

/\*

if(!$dbconn){

die("hiba: ". mysqli\_connect\_error());

} else{

print "sikeres kapcsolat";

}

\*/

If/else utasítással ellenőrizzük, hogy sikeresen kapcsolódtunk e az adatbázishoz. Hiba esetén a program kilép és kiírja a hibaüzenetet, sikeres kapcsolódás esetén pedig kiírjuk, hogy sikeres kapcsolat. Miután leteszteltük, hogy sikeres a kapcsolat az ellenőrzés kódrészletét kommentbe tettük, hogy ne kapjunk minden egyes oldalon visszajelző üzenetet.

@mysqli\_query($dbconn, "SET NAMES utf8");

A @mysqli\_query() függvénnyel pedig beállítjuk karaketerkódolást.

## Főoldal

### Navigációs menü

A navigációs menün az elemek elrendezését flexbox-al oldottuk meg. A címre rákattintva egy <a> tagben lévő href linket helyeztünk el, ami a weboldal főoldalára mutat. Ha mobil vagy tablet nézetben tekintjük meg a weboldalt, akkor a menüpontok helyett egy hamburger ikon jelenik meg, amelyre rákattintva lenyílnak az elrejtett menüpontok, így reszponzívvá téve az oldalt. Ennek a működését Javascript-el készítettük el, ha kattintás történik a hamburger ikonra akkor az active osztály kerül hozzáadásra, melyre a CSS-ben meghatároztuk azokat a tulajdonságokat és értékeket, melynek köszönhetően egymás alatt jelennek meg a menüpontok. A jobb oldalon találhatóak a linkek a Főoldalra, bejelentkezésre és a regisztrációra. Ezt számozatlan listában helyeztük el, amely kattintásra elvezet a megfelelő oldalra.

A cím és az abban található link:

<nav class="menu">

<a class="mcim" href="index.html">ShineGym&Fit</a>

A hamburger ikon és a linkek:

<a href="#" class="toggle-button">

<span class="bar"></span>

<span class="bar"></span>

<span class="bar"></span>

</a>

<div class="links">

<ul>

<li><a href="index.html">Főoldal</a></li>

<li><a href="belepes.php">Bejelentkezés</a></li>

<li><a href="reg.html">Regisztráció</a></li>

</ul>

</div>

</nav>

### Carousel

A főoldal carouselje 4 divből áll, amiben helyet foglal a kép és az ahhoz tartozó szöveg. A kép divje a carousel-img osztálynévre hallgat, míg a szöveg osztálya a text osztálynévre. A köztük lévő slide fade osztályú div elválasztja a carouselben megtalálható képeket, egy áttünéses animációval.

A carousel alján gombok vannak, amik a dots osztálynévre hallgatnak, benne pedig span taggel a 3 gombra onclick eseménnyel rákötött function, amit a JavaScriptben írtunk meg. Egy <a> tagben megírtuk a gombokat, ami a képernyő két szélén található. Kattintásra egy onclick eseménnyel hozzáadunk vagy kivonunk egyet 1 változóból, ez a változó határozza meg, hogy melyik kép jelenik meg éppen.

A Carousel JavaScriptjében pedig létrehoztunk egy változót SlideIndex néven, aminek a kezdőértéke 0, ez határozza meg, hogy melyik kép jelenik éppen meg, léptetésnél ehhez adunk hozzá vagy ebből vonunk ki.

A ShowSlides függvényben megírtuk a lapozáshoz tartozó gombok működését, paraméterként n változót adtuk meg, ami az oldal számát jelentené. A slides változót kiszelektáljuk document.getElementsByClassname-el, amiben megadtuk. Ugyanezt megcsináltuk a dot osztálynévvel is, ennek a változónak a neve pedig a dots. Az első if azt dönti el, hogy ha n nagyobb, mint a képek hossza, akkor a slideIndex legyen egyenlő 1-el, azaz azon az oldalon marad, amit kiválasztottunk. Ha az n kisebb, mint egy, akkor a slideIndex legyen egyenlő a képek hosszával.

Ezek után látható két for ciklus, ahol az első a slideokon iterál végig, a másik pedig a gombokon. A második for ciklus a 3 kis gomb működéséért felel.

let slideIndex = 0;

AutoSlide();

function plusSlides(n) {

showSlides(slideIndex += n);

}

function currentSlide(n) {

showSlides(slideIndex = n);

}

function showSlides(n) {

let i;

let slides = document.getElementsByClassName("slide");

let dots = document.getElementsByClassName("dot");

if (n > slides.length) {

slideIndex = 1;

}

if (n < 1) {

slideIndex = slides.length;

}

for (i = 0; i < slides.length; i++) {

slides[i].style.display = "none";

}

for (i = 0; i < dots.length; i++) {

dots[i].className = dots[i].className.replace(" active", "");

}

slides[slideIndex-1].style.display = "block";

dots[slideIndex-1].className += " active";

}

A második része az AutoSlide-ra mutat, azaz automatikusan lapozza a képeket. Ez hasonló, sőt ugyanaz az előző részhez képest, csak annyi az egésznek a lényege, hogy setTimeout-al x másodpercenként lapozza a képeket. a for ciklus és az if szerkezet között láthatunk egy SlideIndex++-t, ami azt csinálja, hogy hozzáad egyet a képekhez, tehát lapozza azt. És ezt automatizáltuk a setTimeout metódussal, aminek paraméterként megadtuk az AutoSlide functiont, és hogy 6,5 másodpercenkét váltogassa a képeket.

function AutoSlide() {

let i;

let slides = document.getElementsByClassName("slide");

let dots = document.getElementsByClassName("dot");

for (i = 0; i < slides.length; i++) {

slides[i].style.display = "none";

}

slideIndex++;

if (slideIndex > slides.length) {

slideIndex = 1;

}

for (i = 0; i < dots.length; i++) {

dots[i].className = dots[i].className.replace(" active", "");

}

slides[slideIndex-1].style.display = "block";

dots[slideIndex-1].className += " active";

setTimeout(AutoSlide, 6500);

}

### Edzésterv és étrendösszeállítás leírása

A főoldalon található 2 wrapper osztályú div és egy section. Az első és második div konténer az oldalon lévő leírást tartalmazza. Ezeket, hogy a képernyő egyik szélétől a másik széléig legyen, 100%-os szélességgel állítottam be. A margókat és a paddinget 50 pixelesre állítottam, és, hogy automatikus legyen. A felső margóhoz 15 pixelt adtam hozzá, hogy elváljon a konténer tetejétől a h1-es címsor. Elhelyeztünk egy border osztálynévre hallgató divet is, ami a konténer alján található zöld sík. CSS-ben a wrapper osztályú elemekhez adtunk egy fehér betűszínt és fekete hátteret. A görgetősáv stílusát is megváltoztattuk a weboldal alap színeihez igazodva.

Egy link is látható az első containerben, ami a regisztrációra mutat. A containerek szövegeit, text-align: centerrel igazítottuk középre. A container1-en lévő listánál, hogy ne jelenjenek meg a szimbólumok előtte, egy list-style-type: none-t adtunk meg. A betűszíne zöld, és 25pixeles betűméret.

### Számláló(counter)

A számlálót azért hoztuk létre, hogy a vendég nyomon tudja követni az oldalra regisztráltak, a pozitív visszajelzések és a követők számát. Ezt CSS Grid-el valósítottuk meg, hogy egymás mellett legyenek elhelyezve az elemek. A számláló egy section tagben van, ami a counters osztályra hallgat. Ezen belül pedig egy container található. Az ikonokat social media ikonokkal helyeztük el. A szamlalo osztályhívónévvel ellátott div tagben, azon belül a <p> tagben egy 0 van kezdőértékként, hogy a számláló majd a nullától induljon. A data-target pedig azt takarja, hogy a számlálás megfelelően fusson le. A számláló működését is Javascriptben oldottuk meg.

A counters változóban tároltuk el a kiválasztott osztályt, ami ezesetben a counter. A sebességet eltárultuk egy speed nevű változóban, aminek értékül 100-at adtunk. A counters változót egy forEach ciklussal futtatjuk le, ahol functionnel, aminek a paramétere a counter, létrehozunk egy új functiont updateCount néven. Itt indul el a számolás menete. Itt további új változókat hozunk létre, először a target változót. Itt a HTML-ben lévő data-target értékét választjuk ki attribútumként. A második változó a count változó, itt pedig a counter osztályba beiratjuk az értéket. Az inc változóban Math.floor metódus segítségével kiszámoljuk a számláló értékeit, és a számok, amiket kiszámolt, eredményként itt tárolja el if szerkezettel és ha a count kisebb, mint a target és az inc nagyobb, mint 0, akkor a counter innerTextjében, azaz a counter változóba írja be, és adja össze a count és az inc változót. Ezt setTimeouttal, paraméterként az updateCount-al, a számlálás sebességét pedig 8 másodpercre állítottuk. Egyébként pedig a végleges eredményt írja ki a HTML-ben.

let counters = document.querySelectorAll('.counter');

let speed = 100;

counters.forEach(counter => {

let updateCount = () => {

let target =+ counter.getAttribute('data-target');

let count =+ counter.innerText;

let inc = Math.floor((target - count) / speed);

if (count < target && inc > 0) {

counter.innerText = count + inc;

setTimeout(updateCount, 8);

}else{

count.innerText = target;

}

}

updateCount();

});

### Lábléc(footer)

A footerben található minden elérhetőség. Egy footer tagen belül vannak két osztállyal divek. Az egyik osztály a footer-container, a másik pedig listak. A lista osztályon belül található egy számozatlan lista, amiben el vannak helyezve Az ikonok és az elérhetőségek szövege. Ezt szintén flexbox segítségével oldottuk meg.

## Chat

Az első sorban ellenőrizzük, hogy van e bejelentkezett felhasználó. Ha nincs, akkor visszairányítsuk a felhasználót header() függvény segítségével a bejelentkezés oldalára. Ha viszont be van jelentkezve, require() függvénnyel importáljuk a kapcsolat.php fájlt, amely az adatbázishoz való kapcsolódáshoz szolgál. Miután kapcsolódtunk lekérjük a bejelentkezett profil adatait is a sajatProfil.php fájlból, ennek köszönhetően jelenik meg a felső navigációs sávban a nevünk és profilképünk.

session\_start();

if (!isset($\_SESSION['felh\_id'])) {

header("Location: ../belepes.php");

exit();

} else {

define('eleres', true);

require("kapcsolat.php");

//Saját profil adatainak lekérése

require("leker/sajatProfil.php");

### Bal oldali felhasználó lista

Ha van keresett kifejezés azt eltároljuk a $kifejezesChat nevű változóban.

$kifejezesChat = (isset($\_POST['kifejezesChat'])) ? $\_POST['kifejezesChat'] : "";

A $felhChat nevű változóban eltároljuk a keresendő profilok típusát. Ha a bejelentkezett profil típusa edző, akkor a klienst, egyébként pedig edzőt adunk értékül a változónak.

if ($\_SESSION['p\_tipus'] == "edző") {

$felhChat = "kliens";

}else{

$felhChat = "edző";

}

Végrehajtjuk a lekérdezést, amelyben lekérdezzük az összes olyan felhasználót ahol megegyezik a profil típusa a $felhChat nevű változó értékével és ahol szerepel a keresett kifejezés a vezetéknév és keresztnév összefűzésében.

$fosszesChat = mysqli\_query($dbconn, "SELECT \* FROM felhasznalok WHERE profil\_tipus = '{$felhChat}' AND CONCAT(vnev, ' ', knev) LIKE '%{$kifejezesChat}%'");

Majd while ciklus segítségével végig megyünk a lekérdezett eredményeken és összeállítsuk a felhasználók listáját a $chatLista változóba.

$chatLista = "";

while($felhasznalo = mysqli\_fetch\_assoc($fosszesChat)) {

$chatLista .= "<a href=\"?chat={$felhasznalo['felhasznalo\_id']}\">

<div class=\"prof\">

<div class=\"pkep pkep-meret\"><img src=\"../pics/profile/" .$felhasznalo['kep']. "\"></div>

<p>{$felhasznalo['vnev']} {$felhasznalo['knev']}</p>

</div>

</a>";

}

### Chat felület

Ha rákattintunk valakire a bal oldali listában az hozzáadja az adott felhasználó azonosítóját a webcímhez, amit $\_GET metódus segítségével kiolvasunk belőle. Ha létezik egy ilyen azonosító a webcímben, akkor lekérdezzük a választott felhasználó adatait.

if (isset($\_GET['chat'])) {

$sqlValP = mysqli\_query($dbconn, "SELECT vnev, knev, kep, profil\_tipus FROM felhasznalok WHERE felhasznalo\_id = {$\_GET['chat']}");

$ValP = mysqli\_fetch\_assoc($sqlValP);

$Vvnev = $ValP['vnev'];

$Vknev = $ValP['knev'];

$Vkep = $ValP['kep'];

$Vptipus = $ValP['profil\_tipus'];

Ellenőrizzük, hogy a választott profil típusa biztosan ellentétes a mienkkel, hiszen edző csak klienssel és kliens csak edzővel folytathat kommunikációt a chat felületen.

if($\_SESSION['p\_tipus'] == "edző" && $Vptipus == "kliens" || $\_SESSION['p\_tipus'] == "kliens" && $Vptipus == "edző"){

A $\_SESSION['chataz'] globális változónak értékül adjuk az webcímből kiolvasott azonosítót. Ezt a session változót használjuk majd az üzenetek lekérdezéséhez.

$\_SESSION['chataz'] = $\_GET['chat'];

$fogadoAz = $\_GET['chat'];

Ha rányomtunk az üzenetküldés gombra és az üzenet szövege nem üres, akkor beszúrjuk az üzenetek táblába a kimenő azonosítót (saját), bejövő azonosítót (a választott partner azonosítója) és magát az üzenetet.

if(isset($\_POST['ChatUzenet']) && $\_POST['szoveg'] != ""){

$uzenet = $\_POST['szoveg'];

$sqlBeszur = mysqli\_query($dbconn, "INSERT INTO uzenetek (kimeno\_id, bejovo\_id, uzenet) VALUES ('{$\_SESSION['felh\_id']}', '{$fogadoAz}', '{$uzenet}')");

}

Egyébként ha a profil típusa nem ellentétes a miénkkel akkor elvezessük a felhasználót egy hiba.html oldalra, hiszen edző edzővel illetve kliens-klienssel nem cseveghet.

} else{

header("Location: hiba.html");

}

}

### Chat üzenetek lekérése adatbázisból

Először is elindítjuk a munkamenetet a session\_start() függvénnyel majd létrehozzuk a kapcsolatot az adatbázissal.

session\_start();

require("../kapcsolat.php");

Ha létezik a chataz session globális változó, akkor a $fogadoAz változónak értékül adjuk az értékét.

Majd lekérdezzük az összes olyan üzenetet, ahol a kimenő azonosító megegyezik az előbb létrehozott $fogadoAz értékével és a bejövő azonosító megegyezik a bejelentkezett felhasználó azonosítójával, vagy fordítva. Lényegében tehát a 2 azonosítóhoz tartozó üzeneteket kérjük le (a közös üzeneteket).

if (isset($\_SESSION['chataz'])) {

$fogadoAz = $\_SESSION['chataz'];

$sqlKim = mysqli\_query($dbconn, "SELECT kimeno\_id, bejovo\_id, uzenet FROM uzenetek WHERE kimeno\_id = {$\_SESSION['felh\_id']} AND bejovo\_id = {$fogadoAz}

OR bejovo\_id = {$\_SESSION['felh\_id']} AND kimeno\_id = {$fogadoAz}");

Létrehoztunk egy $kiiras nevű változót, amelyben majd konkatenáljuk a kimenő és bejövő üzeneteket. Egy while ciklussal végig megyünk a lekérdezett eredményeken, ahol $sorKim változóval megadtunk egy asszociatív tömböt a mysqli\_fetch\_assoc() segítségével, paraméterként megadtuk neki a $sqlKim változót, amelyben lekérdeztük az adatokat az adatbázisból.

$kiiras = "";

while($sorKim = mysqli\_fetch\_assoc($sqlKim)){

Ha a kimenő azonosító megegyezik a $\_SESSION[’felh\_id’]-vel (a bejelentkezett felhasználó azonosítójával), akkor az egy kimenő üzenet lesz (tehát ezt mi küldtük). Ezt a jobb oldalra rendezzük fekete háttérrel. Egyébként, ha nem egyezik meg, akkor az egy bejövő üzenet lesz (tehát a csevegőpartner küldte), ez a bal oldalra rendezzük és fehér hátteret adunk neki.

if($sorKim['kimeno\_id'] == $\_SESSION['felh\_id']){

$kiiras .= "<div class=\"kimenoUz\">

<p>{$sorKim['uzenet']}</p>

</div>";

}

else{

$kiiras .= "<div class=\"bejovoUz\">

<p>{$sorKim['uzenet']}</p>

</div>";

}

}

Végül pedig print segítségével kiírjuk az üzeneteket.

print $kiiras;

}

### Chat felület JavaScript része

Elsőnek kiválasszuk és eltároljuk a szükséges DOM elemeket külön konstans változókban, amelyekre a továbbiakban hivatkoznunk kell majd. Ezután ezeket a változókat fel tudjuk használni a megfelelő elemek módosítására vagy eseményekhez való kapcsolódásra.

const form = document.querySelector(".chat-szoveg-kuldes");

const inputField = document.querySelector("#szoveg");

const sendBtn = document.querySelector("#ChatUzenet");

const chatBox = document.querySelector(".chatUzenetek");

Az inputFieldet, azaz a szövegmezőt ellenőrizzük, hogy az értéke egyenlő e egy üres stringgel. Ha nem üres, akkor engedélyezzük a küldés gombot, egyébként pedig letiltjuk azt.

inputField.focus();

inputField.onkeyup = ()=>{

if (inputField.value != "") {

sendBtn.disabled = false;

}else{

sendBtn.disabled = true;

}

}

Ezek után megírtuk azt, hogy ha a chatBox változóban megadott elemre, tehát a chatUzenetek osztálynévvel megadott mezőre a weboldalon, ha az egeret rávisszük, akkor az aktív osztályt adja hozzá, ha pedig levisszük az egeret, akkor vegye el ezt az osztályt.

chatBox.onmouseenter = ()=>{

chatBox.classList.add("active");

}

chatBox.onmouseleave = ()=>{

chatBox.classList.remove("active");

}

Létrehoztunk egy url nevű változót, amiben eltároljuk magát a webcímet. Ha nincs megnyitva egy chat ablak, akkor a küldés gomb és a szövegmező ne legyen látható.

let url = window.location.href;

sendBtn.style.display = "none";

inputField.style.display = "none";

Ha a webcím tartalmazza a ’chat=’ szöveget, tehát ha megnyitottunk egy chat ablakot valamelyik felhasználóval, akkor megjelenítsük a küldés gombot és a szövegmezőt is display unset segítségével.

if(url.includes("chat=")){

sendBtn.style.display = "unset";

inputField.style.display = "unset";

A chat felülethez létrehoztuk az AJAX-ot is, hogy automatikusan frissítsen rá minden új változtatásra, üzenetekre. Tehát, ha meg szeretnénk tekinteni az új üzeneteket, akkor ne kelljen folyton ráfrissíteni az oldalra a felhasználónak, hanem automatikusan frissüljön az és azonnal lássuk is az új üzeneteket.

A setInterval() metódus segítségével állítottuk be az időzítést, miszerint 2 másodpercenként fogjuk frissíteni az üzeneteket.

Az AJAX kéréshez egy XMLHttpRequest objektumot hoztunk létre a new XMLHttpRequest() konstruktorral, majd az open() metódussal beállítottuk a kérés típusát (ez esetben GET). Majd az onload eseménykezelőn belül ellenőrizzük a kérés állapotát. Ha a kérés állapota megegyezik az XMLHttpRequest.DONE-al akkor feldolgozzuk a választ. Ha a válasz státusza 200 (OK), akkor a válasz tartalmát eltároljuk a data változóban. Aztán az eredményt megjelenítsük a chatBox innerHTML-ben. Ellenőrizzük, hogy a chatBox elem tartalmazza e az active osztályt, ha nem tartalmazza akkor meghívjuk a scrollToBottom() függvényt, amely legörget a legújabb üzenetekhez, tehát a keret aljára. Végezetül a send() metódus segítségével elküldjük a kérést a szervernek.

//AJAX a chat felülethez

setInterval(() =>{

let xhr = new XMLHttpRequest();

xhr.open("GET", "leker/chatleker.php", true);

xhr.onload = ()=>{

if (xhr.readyState === XMLHttpRequest.DONE) {

if (xhr.status === 200) {

let data = xhr.response;

chatBox.innerHTML = data;

if(!chatBox.classList.contains("active")){

scrollToBottom();

}

}

}

}

xhr.send();

}, 2000)

}

Alatta látható a scrollToBottom funkció, amely hatására legörgetünk az üzenetek aljára.

function scrollToBottom(){

chatBox.scrollTop = chatBox.scrollHeight;

}

Arra is figyeltünk, hogy Enter megnyomására is el tudjuk küldeni az üzenetet. Az inputField-re kötöttünk egy addEventListener-t, amivel érzékeljük, ha gombnyomás történik. Ha a leütött billentyű az Enter akkor az event.preventDefault-al akadályozza meg a weboldal alap működését és reagáljon úgy, mintha egérrel kattintottunk volna a küldés gombra.

inputField.addEventListener("keypress", function(event) {

if (event.key === "Enter") {

event.preventDefault();

sendBtn.click();

}

})

## Kezdőlap

Az oldalt egy alapvető lapvédelemmel egészítettük ki, amely a header() függvény segítségével átirányít minket a bejelentkező oldalra, ha nincs meghatározva a $\_SESSION[’felh\_id’], tehát ha nem vagyunk bejelentkezve nem férhetünk hozzá az oldalhoz. Egy else ágban pedig végrehajtunk minden olyan műveletet amely szükséges az oldal megfelelő megjelenéséhez és használatához.

session\_start();

if (!isset($\_SESSION['felh\_id'])) {

header("Location: ../belepes.php");

exit();

} else {

require("kapcsolat.php");

//Saját profil adatainak lekérése

require("leker/sajatProfil.php");

//felhasználók listája

require('leker/felhLista.php');

}

A require egy olyan utasítás, amely beilleszti egy fájl összes szöveges tartalmát egy másik PHP fájlba. Ez azért egy nagyon hasznos megoldás, mert minden weboldal elkészítése során vannak olyan dolgok, amelyeket többször le kell írnunk, mert több helyen is megjelenik a weboldalon belül, de ezek segítségével rengeteg időt spórolhatunk, mert nem kell újból leírnunk az adott kódrészletet. A require utasítással illesztettük be a kapcsolat.php, sajatProfil.php és a felhLista.php fájlokat. A kapcsolat.php az adatbázishoz való kapcsolódásra szolgál, a sajatProfil.php fájlban az éppen bejelentkezett felhasználó adatait kérjük le. Ennek köszönhetően jelenik meg a felső navigációs sávban a vezetéknév, keresztnév és a profilkép is, de szintén ennek a fájlnak egy részét használjuk az üdvözlő üzenet megjelenítésére is. A felhLista.php fájl a felhasználó listát tartalmazza, amely a bejelentkezett profil típusával ellenkező típusú profilokat jeleníti meg, tehát ha edző profil van éppen bejelentkezve akkor csak a regisztrált kliensek szerepelnek a listában, ha pedig kliens profil van bejelentkezve akkor az edzők listája lesz látható.

Üdvözlő üzenet, a $vnev és $knev változókat a sajatProfil.php fájlban hoztuk létre, ezeket is onnan kapja meg.

<h1>Üdvözöljük <span><?php echo "{$vnev} {$knev}!"; ?></span></h1>

### Keresőmező

Egy <h2> címsorban a listának megfelelően írjuk ki, hogy mégis mit kereshetünk. Egy $\_SESSION-ben meghatározott változóval ellenőrizzük a profil típusát, ha a típusa edző akkor klienseket, ha pedig a típusa kliens akkor edzőket keresünk.

<div class="felh-lista scrollbar">

<h2><i class="fa fa-search" aria-hidden="true"></i> <?php $\_SESSION['p\_tipus'] == "edző" ? print("Kliensek") : print("Edzők");?> keresése</h2>

Formon belül hozunk létre egy kereső típusú inputot kifejezes azonosítóval, ez tartalmazza majd a keresendő kifejezést. Az input submit pedig magát a küldés(Keresés) gombját jelenti.

<form method="post">

<input type="search" name="kifejezes" id="kifejezes" placeholder="Írjon be egy nevet a kereséshez">

<input class="kereses-gomb" type="submit" value="Keresés">

Ha a kifejezés változó nem üres, akkor megjelenítünk egy gombot, amely törli a keresett kifejezést és újra a teljes listát jeleníti meg.

<?php

$kifejezes != "" ? print("<button id=\"kereses-vissza\" onclick=\"$kifejezes = ''\"><i class=\"fa fa-arrow-left\" aria-hidden=\"true\"></i> Vissza</button>") : "";

Az előző gombhoz hasonlóan, ha a kifejezés nem üres megjelenítünk egy szöveget, amely kiírja, hogy találatok a megadott kifejezésre, ezzel is feldobva az alkalmazás használatának élményét.

#### ábra Keresés minta

$kifejezes != "" ? print("<p>Találatok <span>\"{$kifejezes}\"</span> kifejezésre:</p>") : '';

?>

</form>

### Lista

A keresőmező alá pedig kiírjuk magát a lista tartalmát a kimenet változóból, amit a felhLista.php fájlban hoztunk létre és töltöttük fel adatokkal.

<?php echo $kimenet ?>

</div>

### Extra funkció kliens típusú felhasználóknak

Ha a profil típusa kliens require segítségével beillesszük a tevekenysegek.php tartalmát.

<?php

if($\_SESSION['p\_tipus'] == "kliens"){

require("leker/tevekenysegek.php");

Ha a webcímben szerepel egy tevékenység azonosító, akkor lekérdezzük az adatait az adott azonosítójú tevékenységnek, és meg is jelenítsük azt.

if(isset($\_GET['tev'])){

$tevAzon = mysqli\_real\_escape\_string($dbconn, $\_GET['tev']);

$valTevsql = mysqli\_query($dbconn, "SELECT datum, leiras FROM tevekenysegek WHERE tev\_id = '{$tevAzon}'");

$sTev = mysqli\_fetch\_assoc($valTevsql);

Felület összeállítása $tevTeljes változóba

$tevTeljes = "<div class=\"tevTeljes\">

<p class=\"tdatum\">{$sTev['datum']}</p>

<div class=\"tleiras\">".

$sTev['leiras']

."</div>

Bezárás gomb, a felugró ablak jobb felső sarkába van rendezve, kattintásra elrejti a felugró ablakot.

<i class=\"fa fa-times bezar\" aria-hidden=\"true\" onclick=\"tevTeljesBezar()\" title=\"Bezárás\"></i>

A felugró ablakban lehetőségünk van szerkeszteni vagy törölni az adott tevékenységet.

<div class=\"tevTeljGombok\">

Törlés gomb, kattintásra megjelenik egy felugró ablak, ahol meg kell erősítenünk, hogy biztosan törölni szeretnénk az adott tevékenységet. A törlés gombra meghívjuk a tevTejlTorles() nevű függvényt, amelynek paraméterül adtuk az adott tevékenység dátumát és azonosítóját.

<button onclick=\"tevTejlTorles('{$sTev['datum']}', {$tevAzon})\" title=\"Törlés\"><i class=\"fa fa-trash-o\" aria-hidden=\"true\"></i></button>

A módosítás gombra kattintva eljuthatunk a tevModositas.php oldalra, ahol az adott tevékenység dátumát és leírását is tudjuk módosítani. A webcímhez hozzáfűzzük a tevékenység azonosítóját, hogy tudjuk melyik tevékenységet kell majd módosítani.

<button title=\"Módosítás\" onclick=\"location.href='muveletek/tevModositas.php?tevid={$tevAzon}'\"><i class=\"fa fa-pencil-square-o\" aria-hidden=\"true\"></i></button>

</div>

</div>";

Miután mindent összefűztünk a $tevTeljes változóba kiírjuk azt.

print $tevTeljes;

}

}

?>

### tevTeljTorles() függvény

function tevTejlTorles(datum, tevazon){

Swal.fire({

title: 'Biztosan törölni szeretné?',

text: "Ezzel törlődik "+ datum +" napján rögzített tevékenysége! Ezt a műveletet nem tudja majd visszavonni!",

icon: 'warning',

showCancelButton: true,

confirmButtonColor: '#55be3b',

cancelButtonColor: '#080B0C',

confirmButtonText: 'Törlés',

cancelButtonText: 'Mégsem'

}).then((result) => {

Ha a törlés gombra kattintottunk, akkor az oldal átirányít minket a tevTorlese.php oldalra.

if (result.isConfirmed) {

location.href = "muveletek/tevTorlese.php?tevaz=" + tevazon;

}

})

}

## sajatProfil.php – Saját profil adatainak lekérdezése

### Lekérdezés

Egy sql változóban összeállítottuk a lekérdezést, amiben lekérjük a bejelentkezett profil adatait azonosító alapján, amit $\_SESSION-ben határoztunk meg korábban. Miután megtörtént a lekérdezés külön-külön változókban tároljuk el az eredményét.

$felh\_id = $\_SESSION['felh\_id'];

$sql = "SELECT vnev, knev, email, profil\_tipus, kep, nem, bemutatkozo, telefon

FROM felhasznalok

WHERE felhasznalo\_id = {$felh\_id}";

$eredmeny = mysqli\_query($dbconn, $sql);

$sor = mysqli\_fetch\_assoc($eredmeny);

$vnev = $sor['vnev'];

$knev = $sor['knev'];

$email = $sor['email'];

$profilTipus = $sor['profil\_tipus'];

$kep = $sor['kep'];

$nem = $sor['nem'];

$bemutatkozo = $sor['bemutatkozo'];

$telefon = $sor['telefon'];

### Kimenet és profilkép összeállítása

Egy profilkép változóba fűzzük össze a megjelenítendő képet, ezt a változót használjuk majd például a felső navigációs sávban a profilkép megjelenítésére, vagy a saját profil oldalnál is (sProfil.php).

$profilkep = "<img src=\"../pics/profile/" . $kep . "\" alt=\"\">";

A kimenet változóban a saját profil oldalának tartalmát állítsuk össze főként táblázatos megjelenítésben. Az adatokon kívül hozzáadunk még két gombot, amelyből az egyik a kezdőlapra vezet vissza, a másik pedig elvezet egy másik oldalra, ahol a saját adatainkat tudjuk majd módosítani (például: név, e-mail, jelszó, bemutatkozó szöveg... stb.).

$kimenet = "

<div class=\"felh-nev\">

<button id=\"spBtnVissza\" onclick=\"location.href='kezdolap.php';\"; title=\"Vissza a kezdőlapra\"><i class=\"fa fa-arrow-left\" aria-hidden=\"true\"></i> Vissza</button>

<p class=\"nev\">{$vnev} {$knev}</p>

<button id=\"btnAdatokSz\" onclick=\"location.href='muveletek/sAdatModosit.php'\" title=\"Adatok módosítása\"><i class=\"fa fa-pencil-square-o\" aria-hidden=\"true\"></i></button>

</div>

Profilkép és táblázatos rész:

<div class=\"felh-adatok\">

<div class=\"fadatok-pkep\">

<p>Profilkép</p>

<div class=\"kep\"><img src=\"../pics/profile/" .$kep. "\"></div>

</div>

<div class=\"adatok-tabla\">

<table>

<tr>

<th>Vezetéknév:</th>

<td>{$vnev}</td>

</tr>

<tr>

<th>Keresztnév:</th>

<td>{$knev}</td>

</tr>

<tr>

<th>E-mail:</th>

<td>{$email}</td>

</tr>

<tr>

<th>Profil típusa:</th>

<td>{$profilTipus}</td>

</tr>

<tr>

<th>Nem:</th>

<td>{$nem}</td>

</tr>";

Ha a bejelentkezett felhasználó profiljának típusa kliens ellenőriznünk kell van e megadva bemutatkozó szövege és telefonszáma, mivel regisztrációkor ezt a két adatok kliens profilnál nem kötelező megadni.

//Ha a profil típusa kliens ellenőrizzük hogy van e megadva bemutatkozó szövege és telefonszáma

if($profilTipus == "kliens"){

$kimenet .= "<tr>

<th>Telefon:</th>";

$telefon == "" ? $kimenet .= "<td>Nincs megadva</td>" : $kimenet .= "<td>{$telefon}</td>";

Ha a telefon változó üres a kimenet változóhoz az fűzzük hozzá hogy nincs megadva, egyébként pedig a megadott telefonszámot.

$kimenet .= "</tr>

</table>

</div>

<div class=\"bemutatkozo\">

<h2>Bemutatkozó:</h2>";

strlen($bemutatkozo) < 50 ? $kimenet .= "<p>Nincs megadva</p>" : $kimenet .= "<p>{$bemutatkozo}</p>";

Hasonlóan működik a bemutatkozó változó ellenőrzése is, csak itt az strlen() függvénnyel megszámoljuk a változó hosszát és ha a hossza kisebb mint 50 karakter, akkor a nincs megadva szöveget adjuk hozzá a kimenet változóhoz. Erre azért van szükség, mivel a redactor, amit szabad textarea szövegeknél használtunk alapból hozzáad pár <p> elemet és néhány sortörést is <br>, hogy megfelelően jelenjen majd meg a megadott szöveg. Ezért itt nem működik az, ha azt vizsgáljuk üres-e a változó. Regisztrációkor pedig alapból meghatározzuk, hogy a bemutatkozó nem lehet rövidebb 50 karakternél.

$kimenet .= "</div>";

}

Ha a bejelentkezett profil típusa kliens, akkor nincs szükségünk ellenőrzésre, ebben az esetben csak egyszerűen hozzáfűzzük a kimenet változóhoz az adatokat.

else{

$kimenet .= "<tr>

<th>Telefon:</th>

<td>{$telefon}</td>

</tr></table>

</div>

<div class=\"bemutatkozo\">

<h2>Bemutatkozó:</h2>

<p>{$bemutatkozo}</p>

</div>";

}

## felhLista.php – (Felhasználó lista a kezdőlapon)

Ha elküldtük a form tartalmát post metódussal ki tudjuk olvasni a kifejezést. Itt ellenőrizzük az isset() függvénnyel , hogy létezik-e a $\_POST[’kifejezes’] és ha létezik eltároljuk az a $kifejezés változóban, egyébként pedig a $kifejezés változónak üres értéket adunk.

$kifejezes = (isset($\_POST['kifejezes'])) ? $\_POST['kifejezes'] : "";

A $felhTípus változóban tároljuk el, hogy milyen típusú profilokat szeretnénk majd lekérdezni. If segítségével ellenőrizzük hogy a profil típusa edző e, ha ez a típus edző akkor kliens értéket adunk a változónak egyébként pedig edző értéket.

if($\_SESSION['p\_tipus'] == "edző"){

$felhTipus = "kliens";

}

else{

$felhTipus = "edző";

}

Összes megfelelő típusú felhasználó lekérdezése. CONCAT() függvénnyel összefűzzük a vezetéknevet és a keresztnevet, ha ebben az összefűzésben szerepel a keresett kifejezés, akkor azokat az adatokat fogjuk csak lekérdezni, ahol az összefűzésben szerepel valahol a keresett kifejezés.

$fosszes = mysqli\_query($dbconn, "SELECT felhasznalo\_id, vnev, knev, kep FROM felhasznalok WHERE profil\_tipus = '{$felhTipus}' AND CONCAT(vnev, ' ', knev) LIKE '%{$kifejezes}%'");

Mielőtt összeállítsuk a listát a kimenet változóba, ellenőrizzük, hogy van-e már valamilyen kapcsolata az adott profillal a bejelentkezett felhasználónak.

$kimenet = "";

while($felh = mysqli\_fetch\_assoc($fosszes)){

Ellenőrizzük, hogy az Edző-Kliens kapcsolatok táblában van-e olyan kapcsolat amiben az egyik azonosító megegyezik a bejelentkezett profil azonosítójával, a másik azonosító pedig az éppen lekérdezett profil azonosítójával azonos.

$fEll = "SELECT elfogadva FROM edzoklienskapcs

WHERE fogado\_az = {$\_SESSION['felh\_id']} AND kuldo\_az = {$felh['felhasznalo\_id']}

OR kuldo\_az = {$\_SESSION['felh\_id']} AND fogado\_az = {$felh['felhasznalo\_id']}";

$fEllEredmeny = mysqli\_query($dbconn, $fEll);

$fEllSor = mysqli\_fetch\_assoc($fEllEredmeny);

Ha van ilyen, akkor az fvan változónak 1-et adunk értékül és eltárljuk az elfogadás állapotát (el van-e fogadva vagy sem [true/false érték]) az felfogadva változóban.

$fvan = 0;

if(mysqli\_num\_rows($fEllEredmeny) > 0){

$fvan = 1;

$felfogadva = $fEllSor['elfogadva'];

}

Elkezdjük a kimenet változóban összeállítani a listát.

$kimenet .= "<div class=\"felhKeret\">

<a href=\"profilAdatok.php?felhasznalo\_id=" .$felh['felhasznalo\_id']." \">

<div class=\"felh\">

<div class=\"pkep pkep-meret\"><img src=\"../pics/profile/" .$felh['kep']. "\"></div>

<p>{$felh['vnev']} {$felh['knev']} ";

Ha van az adott felhasználóval kapcsolat és az el is van fogadva, akkor profil típusnak megfelelően egy kiegészítő szöveget fűzünk hozzá a kimenethez (Már kliens/edző).

if($fvan == 1){

if($felfogadva == 1){

if($\_SESSION['p\_tipus'] == "edző"){

$kimenet .= "(Már a kliensei közé tartozik)";

}

if($\_SESSION['p\_tipus'] == "kliens"){

$kimenet .= "(Már az edzői közé tartozik)";

}

}

Ha nincs elfogadva de már felkértük az adott felhasználót, vagy ő kér fel minket, akkor a felkéve szöveget fűzzük a kimenethez.

(Ha nincs semmilyen kapcsolat az adott felhasználóval, sem elfogadott, sem függőben lévő, akkor nem fűzünk hozzá semmilyen kiegészítő szöveget a kimenethez.)

else{

$kimenet .= "(Felkérve)";

}

}

$kimenet .= "</p>

</div></a></div>";

}

## tevekenysegek.php - Tevékenységek lekérése

### Lapozó

Ha létezik a $\_GET['lap'], akkor a $lap változónak értékül adjuk amit kiolvastunk, egyébként a $lap változónak 1-et adunk értékül, így amikor betöltődik az oldal alapból az első lap fog megjelenni.

if (isset($\_GET['lap'])) {

$lap = $\_GET['lap'];

} else {

$lap = 1;

}

Meghatározzuk, hány elemet szeretnénk egy darab lapon látni (jelen esetben ez 3).

$no\_of\_records\_per\_page = 3;

$offset = ($lap-1) \* $no\_of\_records\_per\_page;

Lekérdezzük, hogy hány sorból áll a tevékenységek tábla, ahol a felhasználó azonosító megegyezik a bejelentkezett felhasználó azonosítójával.

$total\_pages\_sql = "SELECT COUNT(tev\_id) FROM tevekenysegek WHERE felhasznalo\_id = {$\_SESSION['felh\_id']}";

$result = mysqli\_query($dbconn,$total\_pages\_sql);

$total\_rows = mysqli\_fetch\_array($result)[0];

ceil() függvény segítségével meghatározzuk, hogy hány lapot fog kitölteni az összes lekérdezés.

$total\_pages = ceil($total\_rows / $no\_of\_records\_per\_page);

### Shorter függvény

function shorter($text, $chars\_limit){

Ha a megadott szöveg karaktereinek hossza nagyobb, mint a megadott limit, akkor lerövidíti azt és eltárolja a $new\_text változóba, mielőtt visszaadná az új szöveget még a trim() függvény segítségével eltávolítja a szöveg elejéről és végéről a szóköz vagy egyéb hasonló karaktereket.

if (mb\_strlen($text) > $chars\_limit){

$new\_text = mb\_substr($text, 0, $chars\_limit);

$new\_text = trim($new\_text);

return $new\_text . "...";

}

Ha a megadott szöveg karaktereinek hossza kisebb, mint a megadott limit csak szimplán visszadobja nekünk az eredeti szöveget.

else{

return $text;

}

}

### Tevékenységek lekérdezése és kimenet összeállítása

Lekérdezzük a tevékenység azonosítóját, dátumát (melyik napra szól) és leírását, ahol a felhasználó azonosítója megegyezik a bejelentkezett felhasználó azonosítójával. Sorba rendezzük dátum szerint visszafelé és lekorlátozzuk a lapok tartalmának megfelelő megjelenítéshez.

$sql = "SELECT tev\_id, datum, leiras

FROM tevekenysegek

WHERE felhasznalo\_id = {$\_SESSION['felh\_id']}

ORDER BY datum DESC

LIMIT $offset, $no\_of\_records\_per\_page";

$res\_data = mysqli\_query($dbconn,$sql);

A tevKimenet változóba elkezdjük összeállítani a tevékenységeket, hozzáadunk 1 gombot is ami elvezet 1 oldalra (tevRogzitese.php) ahol új tevékenységet lehet rögzíteni.

$tevKimenet = "<div class=\"tevk\"><h2>Tevékenységek</h2>

<button onclick=\"location.href='tevRogzitese.php'\">Új rögzítése</button>

<div class=\"tevekenysegek\">";

Ha a lekérdezésünk tartalmaz sorokat, tehát van rögzített tevékenysége a bejelentkezett felhasználónak, akkor minden ilyet hozzáfűzünk a tevKimenet változóhoz.

if(mysqli\_num\_rows($res\_data) != 0){

while($sorTev = mysqli\_fetch\_array($res\_data)){

$tDatum = $sorTev['datum'];

$tLeiras = $sorTev['leiras'];

Ha van meghatározva get metódusban lap, akkor a webcímhez hozzáfűzzük még a tevékenység azonosítóját is, alap esetben ugyanis felülírná a webcímet és újra az első oldalra vezetne, amikor rákattintunk egy tevékenységre. Egyébként ha nincs meghatározva lap akkor csak a tevékenység azonosítóját adjuk a webcímhez, amivel a teljes tevékenység fog majd megjelenni.

isset($\_GET['lap']) ? $link = "?lap={$lap}&tev={$sorTev['tev\_id']}" : $link = "?tev={$sorTev['tev\_id']}";

A tev osztályt tartalmazó div-re kattintva megjelenik a teljes tevékenység egy felugró ablakban.

$tevKimenet .= "<div class=\"tev\" onclick=\"location.href='{$link}'\">

<p class=\"tdatum\">{$tDatum}</p>

A tevékenység szövegének rövidítéséhez egy shorter nevű függvényt használtunk, amelynek ha paraméterül adjuk a szöveget és a kívánt karaktert(hosszt) lerövidíti a megadott karakter számára azt. Használtuk még a strip\_tags() beépített függvényt is, mivel a redactor alapvetően használja a html elemeket (p, br, h1, div, table.. stb.) és ha pont egy lezáró tag előtt vágjuk el a szöveget az problémát okoz az oldal megjelenésében.

<div class=\"tleiras\">". shorter(strip\_tags($tLeiras), 190) ."</div>

</div>";

}

}

Ha a lekérdezés nem tartalmaz sorokat, akkor a nincs rögzített tevékenység szöveg fog megjelenni a felhasználónak.

else{

$tevKimenet .= "<p id=\"nincsRtev\">Nincs rögzített tevékenység!</p>";

}

$tevKimenet .= "</div>";

## tevModositas.php – Tevékenység módosítása/szerkesztése

A modID változóban eltároljuk a webcímből kiolvasott azonosítót.

$modID = mysqli\_real\_escape\_string($dbconn, $\_GET['tevid']);

Lekérdezzük az adott azonosítójú tevékenység adatait és a beviteli mezőknek kell értékül adnunk.

$sql = mysqli\_query($dbconn, "SELECT datum, leiras FROM tevekenysegek WHERE tev\_id = {$modID}");

$sor = mysqli\_fetch\_assoc($sql);

$kDatum = $sor['datum'];

$kLeiras = $sor['leiras'];

Ha rányomtunk a módosítás gombra kiolvassuk a beviteli mezőkből a módosított adatokat és frissítsük az adatbázisban az adatokat.

if(isset($\_POST['modosit'])){

$pdatum = $\_POST['dat'];

$pleiras = $\_POST['leir'];

$sqlUpdate = mysqli\_query($dbconn, "UPDATE tevekenysegek SET datum = '{$pdatum}', leiras = '{$pleiras}' WHERE tev\_id = {$modID}");

header("Location: ../kezdolap.php");

}

## tevTorlese.php – Tevékenység törlése

require("../kapcsolat.php");

A $torlendo változóban eltároljuk a webcímből kiolvasott tevékenység azonosítóját.

$torlendo = mysqli\_real\_escape\_string($dbconn, $\_GET['tevaz']);

Aztán végrehajtunk egy mysql parancsot, amiben töröljük az adott tevékenységet a tevekenysegek táblából, ahol a tevékenység azonosítója megegyezik a webcímből kiolvasott azonosítóval.

$sql = mysqli\_query($dbconn, "DELETE FROM tevekenysegek WHERE tev\_id = {$torlendo}");

Miután megtörtént a törlés átirányítsuk az oldalt a kezdőlapra header() függvény segítségével.

header("Location: ../kezdolap.php");

## tevRogzitese.php – Tevékenység rögzítése

Ha rányomtunk a rögzítés gombra, tehát létezik a $\_POST[’rogzites’] akkor kiolvassuk a dátumot és az azonosítót, majd eltároljuk azokat 1-1 változóban.

if(isset($\_POST['rogzites'])){

$datum = $\_POST['datum'];

$leiras = $\_POST['tev'];

Miután kiolvastuk az adatokat, végrehajtunk egy mysql parancsot, amiben beszúrjuk a kitöltött adatokat és a bejelentkezett felhasználó azonosítóját beszúrjuk a tevékenységek táblában.

$sql = mysqli\_query($dbconn, "INSERT INTO tevekenysegek (felhasznalo\_id, datum, leiras) VALUES ('{$\_SESSION['felh\_id']}', '{$datum}', '{$leiras}')");

Egy session változóban eltároljuk, hogy sikeres a rögzítés és miután elnavigáltunk a főoldalra meg tudjuk majd jeleníteni ezt az üzenetet.

$\_SESSION['tevrogz'] = "<p>Sikeres rögzítés!</p>";

header("Location: kezdolap.php");

}

Megjelenítés a főoldalon (kezdolap.php):

<div class="sikeres">

<?php

Ha létezik a tevrogz globális session változó, akkor kiírjuk azt és miután kiírtuk hatástalanítjuk a változót az unset() függvénnyel.

if(isset($\_SESSION['tevrogz'])){

print $\_SESSION['tevrogz'];

unset($\_SESSION['tevrogz']);

}

?>

</div>

## edzesterv.php (Edzéstervek oldal)

### Edző típusú profil esetén

Mivel ennél az oldalnál 2 különböző felületet kell megjelenítenünk az edző és kliens felhasználóknak, egy $\_SESSION változóból kiolvassak a már korábban (bejelentkezésnél) eltárolt profil típusát. Ez alapján 2 jól elkülöníthető felületet tudunk megjeleníteni a mindkét típusú felhasználónak.

if($\_SESSION['p\_tipus'] == "edző"){

Ha létezik a $\_POST globális változóban eltárolt kifejezés, akkor eltároljuk azt egy egyszerű $kifejezes változóban, majd a $felulet változóban összeállítjuk a keresésre szolgáló formot.

$kifejezes = (isset($\_POST['kifejezes'])) ? $\_POST['kifejezes'] : "";

$felulet = "

<h1>Kliensek Edzéstervei</h1>

<div class=\"sKliensekL scrollbar\">

<form method=\"post\">

<input type=\"search\" name=\"kifejezes\" id=\"kifejezes\" placeholder=\"Írjon be egy nevet a kereséshez\">

<input class=\"kereses-gomb\" type=\"submit\" value=\"Keresés\">";

Ha van keresett kifejezés, akkor a $felulet változóhoz hozzáfűzünk egy gombot, amit a keresett kifejezés törlésére szolgál, tehát ha rányomunk újra a teljes lista fog megjelenni.

$kifejezes != "" ? $felulet .= "<button id=\"kereses-vissza\" class=\"kereses-gomb\" onclick=\"$kifejezes = ''\"><i class=\"fa fa-arrow-left\" aria-hidden=\"true\"></i> Vissza</button>" : "";

Ezen kívül hozzáadunk még egy apró szöveget, amely megjeleníti, hogy mire kerestünk pontosan.

$kifejezes != "" ? $felulet .= "<p>Találatok <span>\"{$kifejezes}\"</span> kifejezésre:</p>" : '';

$felulet .= "</form>";

Az Edző-Kliens kapcsolatok táblából lekérdezzük a már meglévő (elfogadott) kapcsolatok alap adatait (felkérés küldőjének és fogadójának azonosítója), ahol a bejelentkezett profil azonosítója megegyezik a fogadó vagy küldő azonosítójával.

$sql = "SELECT kuldo\_az, fogado\_az, elfogadva FROM edzoklienskapcs WHERE elfogadva = 1 AND kuldo\_az = {$\_SESSION['felh\_id']} OR fogado\_az = {$\_SESSION['felh\_id']}";

$eredmeny = mysqli\_query($dbconn, $sql);

Mielőtt while ciklus használatával végig megyünk a találatokon ellenőrizzük, hogy nem nulla sorból áll a lekérdezés.

if(mysqli\_num\_rows($eredmeny) != 0){

while($sor = mysqli\_fetch\_assoc($eredmeny)){

$kuldoaz = $sor['kuldo\_az'];

$fogadoaz = $sor['fogado\_az'];

Ha a bejelentkezett profil azonosítója megegyezik a küldő azonosítójával, akkor a fogadó azonosítója alapján le kell kérnünk a fogadó alap adatait, hogy később megjelenítsük azt.

CONCAT() függvénnyel összefűzzük a vezetéknevet és a keresztnevet, ha ebben az összefűzésben szerepel a keresett kifejezés, akkor azokat az adatokat fogjuk csak lekérdezni, ahol az összefűzésben szerepel valahol a keresett kifejezés.

if($kuldoaz == $\_SESSION['felh\_id']){

$sql2 = "SELECT felhasznalo\_id, vnev, knev, kep, fogado\_az FROM edzoklienskapcs

INNER JOIN felhasznalok ON felhasznalo\_id = fogado\_az

WHERE CONCAT(vnev, ' ', knev) LIKE '%{$kifejezes}%'

AND kuldo\_az = {$\_SESSION['felh\_id']} AND fogado\_az = {$fogadoaz} AND elfogadva = 1";

$kerdezendo = "fogado\_az";

}

Ha a fogadó azonosítójával egyezik meg, akkor a küldő adatait kérdezzük le.

else if($\_SESSION['felh\_id'] == $fogadoaz){

$sql2 = "SELECT felhasznalo\_id, vnev, knev, kep, kuldo\_az FROM edzoklienskapcs

INNER JOIN felhasznalok ON felhasznalo\_id = kuldo\_az

WHERE CONCAT(vnev, ' ', knev) LIKE '%{$kifejezes}%'

AND fogado\_az = {$\_SESSION['felh\_id']} AND kuldo\_az = {$kuldoaz} AND elfogadva = 1";

$kerdezendo = "kuldo\_az";

}

Ha létezik az előbbi feltételekben létrehozott $sql2 akkor végrehajtjuk a lekérdezést.

if(isset($sql2)){

$eredmeny2 = mysqli\_query($dbconn, $sql2);

$sor2 = mysqli\_fetch\_assoc($eredmeny2);

if($sor2 != 0){

A $kliens változóban állítsuk össze a listát a meglévő kliensekről.

$kliens = "

<div class =\"kliens\">

A profiljára kattintva elnavigál minket az oldal a kiválasztott profil adatlapjához, ahol megtekinthetők az alap adatok, lehetőség van a csevegéshez, edzéstervek megtekintéséhez navigálni.

<a href=\"profilAdatok.php?felhasznalo\_id=" .$sor2['felhasznalo\_id']. "\" title=\"Profil megtekintése\">

<div class=\"felh\">

<div class=\"pkep pkep-meret\"><img src=\"../pics/profile/" .$sor2['kep']. "\"></div>

<p>{$sor2['vnev']} {$sor2['knev']}</p>\n

</div>

</a>

A felületen megjelenítünk majd 2 gombot ahol az egyiknél új edzéstervet/étrendet rögzíthetünk a kiválasztott kliensüknek, a másiknál pedig a már meglévő bejelentkezett profil által megírt edzéstervek láthatóak (ha van). Mindkét gombnál $\_GET használatával fogjuk majd kiolvasni a szükséges adatokat ezért onclick metódussal csak szimplán elnavigálunk az erre a célra létrehozott oldalra és hozzáfűzzük a szükséges azonosítókat.

<div class=\"gombok\">

<button onclick=\"location.href='edzesterv-felvitel.php?felvitel=". $sor2[$kerdezendo] ."'\">Új Edzésterv Felvétele</button>

<button onclick=\"location.href='etervM.php?kliens=". $sor2[$kerdezendo] ."'\">Edzéstervek</button>

</div>

</div>";

$felulet .= $kliens;

Ha a lekérdezés nem tartalmaz sorokat, akkor az azt jelenti hogy a bejelentkezett edző típusú profilnak még nincs hozzá tartozó kliense, ennek szemléltetésére megjelenítünk a lista helyén egy erre megfelelő üzenetet.

} else{

$felulet .= "<p class=\"nincsKl\">Önnek még nincs egy kliense sem!</p>";

}

$felulet .= "</div>";

}

### Kliens típusú profil esetén

Ha a profil típusa nem edző akkor az else ágban a kliens típusú profilhoz megfelelő felületet állítsuk össze.

else{

Elkezdjük a felületet összeállítani egy h1-es címsorral, amiben megjelenik a bejelentkezett profil vezetékneve és keresztneve is.

$felulet = "<h1>{$vnev} {$knev} Edzéstervei</h1>";

A $kliensID változóban eltároljuk a bejelentkezett profil azonosítóját a korábban meghatározott globális session változóból.

$kliensID = $\_SESSION['felh\_id'];

Adatbázisból lekérjük a bejelentkezett profilhoz tartozó tervek alap adatait.

$eredmeny = mysqli\_query($dbconn, "SELECT terv.terv\_id, terv.neve, terv.leiras, terv.kapcs\_id, kuldo\_az, fogado\_az

FROM terv INNER JOIN edzoklienskapcs ON edzoklienskapcs.kapcs\_id = terv.kapcs\_id

WHERE kuldo\_az = '{$kliensID}'

OR fogado\_az = '{$kliensID}'");

$etervKi = "<div class=\"edzestervek\">";

Ha van a lekérdezésnek eredménye while ciklussal végig megyünk az adatokon.

if(mysqli\_num\_rows($eredmeny) != 0){

while($sor = mysqli\_fetch\_assoc($eredmeny)){

$kuldoAz = $sor['kuldo\_az'];

$fogadoAz = $sor['fogado\_az'];

Ha a kuldő azonosító azonos a kliens azonosítójával(bejelentkezett profil), akkor az $edzoID változóban eltároljuk a fogadó azonosítóját, egyébként pedig a küldő azonosítóját tároljuk el benne.

if($kuldoAz == $kliensID){

$edzoID = $fogadoAz;

}

else{

$edzoID = $kuldoAz;

}

$etID = $sor['terv\_id'];

Az előbb meghatározott $edzoID segítségével lekérjük az adatbázisból az edző vezetéknevét és keresztnevét.

$edzoneve = mysqli\_query($dbconn, "SELECT vnev, knev FROM felhasznalok WHERE felhasznalo\_id = {$edzoID}");

$eneve = mysqli\_fetch\_assoc($edzoneve);

$eVnev = $eneve['vnev'];

$eKnev = $eneve['knev'];

Miután minden szükséges adatot lekértünk összeállítjuk az edzéstervek megjelenését, melyet a „$etervKi” változóba fűztünk össze.

$etervKi .= "<a href=\"teljeset.php?edzesterv={$etID}\">

<div class=\"edzesterv\">

<div class=\"etneve\">

<p>Edzésterv neve</p>

<h3>{$sor['neve']}</h3>

</div>

<div class=\"etleirasa\">

<p>Leírás<br>". shorter(strip\_tags($sor['leiras']), 150)."</p>

</div>

<div class=\"etkitol\">

<p>Edző neve</p>

<h3>{$eVnev} {$eKnev}</h3>

</div>

</div>

</a>";

}

}

Egyébként, ha a lekérdezésnek nincs eredménye, tehát nincs egy megírt edzésterve sem a kliensnek, akkor egy eligazító üzenetet küldünk ki számára:

else{

$etervKi .= "

<div class=\"etervKozep\">

<div class=\"eTervSegitseg\">

<p class=\"etSegitsegC\">Önnek még nincs egy edzésterve sem!</p>

<p>Edzéstervet az Edző típusú profillal rendelkező felhasználóktól tud kérni.</p>

<ol>

<li>Kérjen fel egy edzőt</li>

<li>Ha az edző elfogadta a felkérését, chat részben meg tudják beszélni a további részleteket (milyen edzéstervet / étrendet szeretne, korábbi sérülések, betegségek stb.)</li>

<li>Amikor mindezt megbeszélték, az edző megírja a személyre szabott edzéstervet/étrendet, amint készen van az edzéstervek menüpontban fogja tudni megtekinteni.</li>

</ol>

</div>

</div>";

}

$etervKi .= "</div>";

$felulet .= $etervKi;

}

## Shorter függvény

A shorter függvényt hosszabb szövegek rövidítésére használjuk, hogy egy apró betekintő szöveget tudjunk megjeleníteni a kívánt kis méretű divben, p tagben.

function shorter($text, $chars\_limit){

Ha a megadott szöveg karaktereinek hossza nagyobb, mint a megadott limit, akkor lerövidíti azt és eltárolja a $new\_text változóba, mielőtt visszaadná az új szöveget még a trim() függvény segítségével eltávolítja a szöveg elejéről és végéről a szóköz vagy egyéb hasonló karaktereket.

if (mb\_strlen($text) > $chars\_limit){

$new\_text = mb\_substr($text, 0, $chars\_limit);

$new\_text = trim($new\_text);

return $new\_text . "...";

}

Ha a megadott szöveg karaktereinek hossza kisebb, mint a megadott limit csak szimplán visszadobja nekünk az eredeti szöveget.

else{

return $text;

}

}

# Tesztelés

A weboldalaink HTML kódrészleteit a <https://validator.w3.org/> oldalán ellenőriztük. Habár a böngészőben hibátlanul jelentek meg az oldalaink, ennek ellenére találtunk pár apró hibát. Az egyik az volt, hogy a főoldalon az img elemekről hiányzott az „alt” alternatív szöveg megadása, arra az esetre, ha nem töltődne a kép a böngészőben letöltéskor, és egy img elemen a kép neve tartalmazott egy szóközt. Ezeket a hibákat kijavítottuk.

A CSS kódrészleteket a <https://jigsaw.w3.org/css-validator/> oldalán ellenőriztük. A CSS-nél előnyünkre szolgált, hogy a Visual Studio Code észlelte a hibát, így még a programozás folyamán ki tudtuk javítani. Így a validátor nem talált hibát a CSS kódban.

A Javascript kód írása közben console.log() segítségével folyamatosan ellenőriztük, hogy sikeresen tároltuk e el az adatokat a változókban, és azt az eredményt kapjuk e, amelyet várunk. A böngésző konzoljában folyamatosan figyelemmel követtük, hogy nem észlel-e hibát a programban. A folyamatos ellenőrzés és javascript kód ellenőrzése azért fontos, mert ha csak a legvégén ellenőrizzük és hibát tartalmaz a program, akkor már sokkal nehezebb megtalálni, mint folyamatában.

A PHP kódok ellenőrzését a <https://www.phptools.online/php-checker> oldalán is ellenőriztük. Habár egyrészt, a Visual Studio Code, ha hibát észlel a PHP kódrészletekben piros aláhúzással jelzi. Továbbá több kiegészítő modult is letöltöttünk a VSC-hez, amelyek segítik a debugolást a php programozás során.

A másik segítségünk az az, hogy weboldalon szintén azonnal megjelenik a hiba üzenet, legyen az akár szintaktikai hiba, vagy akár valamilyen egyéb hiba. Print\_r –el és a var\_dump()-al folyamatosan teszteltük a kódunkat, hogy hozzá jutunk e azokhoz az adatokhoz, amelyekre épp szükségünk van. Ennek eredményeképpen a PHP kód is hibátlanul működik minden oldalon, a böngészőben nem jelenik meg sehol hiba, bármilyen műveletet is hajtunk végre.

# Összefoglalás

A közös munkánk során mindent meg tudtunk valósítani, amit elterveztünk. A főoldal készítésének első fázisában meggyűlt a bajunk a carousel helyes működésével. Gondunk volt, az automatikus lapozás működtetésével is, ugyanis hiába kattintottunk a fejléc képek két oldalán lévő nyilakra, nem váltott át a következő képre. A probléma forrását, a hozzá tartozó Javascript fájlban találtuk meg, és sikeresen ki is javítottunk.

A következő akadályba a chat felület elkészítésekor ütköztünk. A probléma az volt, hogy nem tudtunk automatikusan legördíteni a legújabb üzenethez. A megoldást CodingNepal youtube csatornáján találtunk, ahol részletesen bemutatta a chat működtetését, és ennek a segítségével sikeresen megoldottuk a problémát.

Természetesen maradt még számos lehetőség, amellyel majd érdemes lesz tovább fejleszteni a weboldalunkat. Az egyik és a legfontosabb majd a fizetési rendszer megoldása lesz, hiszen az edzőknek biztosítani kell a bevételt, a klienseknek pedig egy kényelmes fizetési lehetőséget pl. Paypalon keresztül.

Ezen kívül az oldal fenntartásának is vannak költségei, mint szerverbérlet és mi munkának is, hiszen a weboldalt folyamatosan fejleszteni és karbantartani szükséges. Ezeknek a költségeknek a fedezetét úgy tudnánk megoldani, hogy amikor a vendég fizet az edzőnek, akkor abból 10-20% jutalékot számolnánk fel minden fizetés alkalmával.

Tervezünk hozzá mobil applikációt is írni, de ehhez még folytatnunk kell a tanulást.

További fejlesztési lehetőség szintén annak megoldása, hogy a kliens ki tudja választani az edzést, amit el szeretne végezni, és ha végzett az edzéssel, be tudja pipálni azt a tevékenységet, amelyet elvégzett, melyet az edző is ellenőrizhet. Ehhez az elképzeléshez az adatbázisunkat is bővíteni kell majd.

A vizsgaremekünk elkészítéséhez natív PHP-t használtunk, de a későbbiekben talán érdemes lenne egy manapság divatos keretrendszerbe pl. Laravelbe áthelyezni.

Átléphetnénk az országhatárokat is, ha több nyelvet lehetne választani. Mivel Sebastian szlovák, így mindenképpen érdemes lenne kihasználnunk a szlovák nyelv tudását is. Mivel mindketten tanultunk angolul, így az lenne a harmadik választható nyelv majd az oldalunkon.

# Források

1. CodingNepal: Chat Application using PHP with MySQL & JavaScript Link: https://www.youtube.com/watch?v=VnvzxGWiK54&t=7326s
2. Máté Balázs: A JavaScript definíciója Link: https://matebalazs.hu/javascript.html
3. Stackoverflow: XMLHttpRequest error while using http post flutter web Link: https://stackoverflow.com/questions/67253808/xmlhttprequest-error-while-using-http-post-flutter-web
4. Webiskola: MI AZ A PHP? A PHP FOGALMA ÉS BEMUTATÁSA Link: https://webiskola.hu/php-ismeretek/mi-az-a-php-fogalma-bemutatasa/
5. Wikipedia: CSS Link: https://hu.wikipedia.org/wiki/CSS
6. Wikipedia: GitHub Link: https://hu.wikipedia.org/wiki/GitHub
7. Wikipedia: HTML Link: https://hu.wikipedia.org/wiki/HTML
8. Wikipedia: MySQL Link:https://hu.wikipedia.org/wiki/MySQL
9. Wikipedia: PHP Link: https://hu.wikipedia.org/wiki/PHP
10. Wikipedia: Visual Studio Code Link: https://hu.wikipedia.org/wiki/Visual\_Studio\_Code
11. Wikipedia: XAMPP Link: https://hu.wikipedia.org/wiki/XAMPP

# Ábrajegyzék

[1) ábra Főoldal 9](#_Toc132125813)

[2) ábra Regisztrációs felület 11](#_Toc132125814)

[3) ábra Bejelentkezési felület 11](#_Toc132125815)

[4) ábra Kezdőlap 12](#_Toc132125816)

[5) ábra Felhasználó adatlapja 13](#_Toc132125817)

[6) ábra Saját profil 13](#_Toc132125818)

[7) ábra Edzéstervek oldal (edző profil) 14](#_Toc132125819)

[8) ábra Edzésterv felvitele 15](#_Toc132125820)

[9) Edzéstervek oldal (kliens profil) 16](#_Toc132125821)

[10) ábra Chat 16](#_Toc132125822)

[11) ábra Kliensek kezelése 17](#_Toc132125823)

[12) ábra Edzők kezelése 17](#_Toc132125824)

[13) ábra Felhasználók tábla 21](#_Toc132125825)

[14) ábra Üzenetek tábla 22](#_Toc132125826)

[15) ábra Tevékenységek tábla 22](#_Toc132125827)

[16) ábra Edző-Kliens kapcsolatok tábla 23](#_Toc132125828)

[17) ábra Terv tábla 24](#_Toc132125829)

[18) ábra Edzéstervek tábla 24](#_Toc132125830)

[19) ábra Étrendek tábla 25](#_Toc132125831)

[20) ábra Adatbázis diagram 25](#_Toc132125832)

[21) ábra Visszalépés gomb 41](#_Toc132125833)