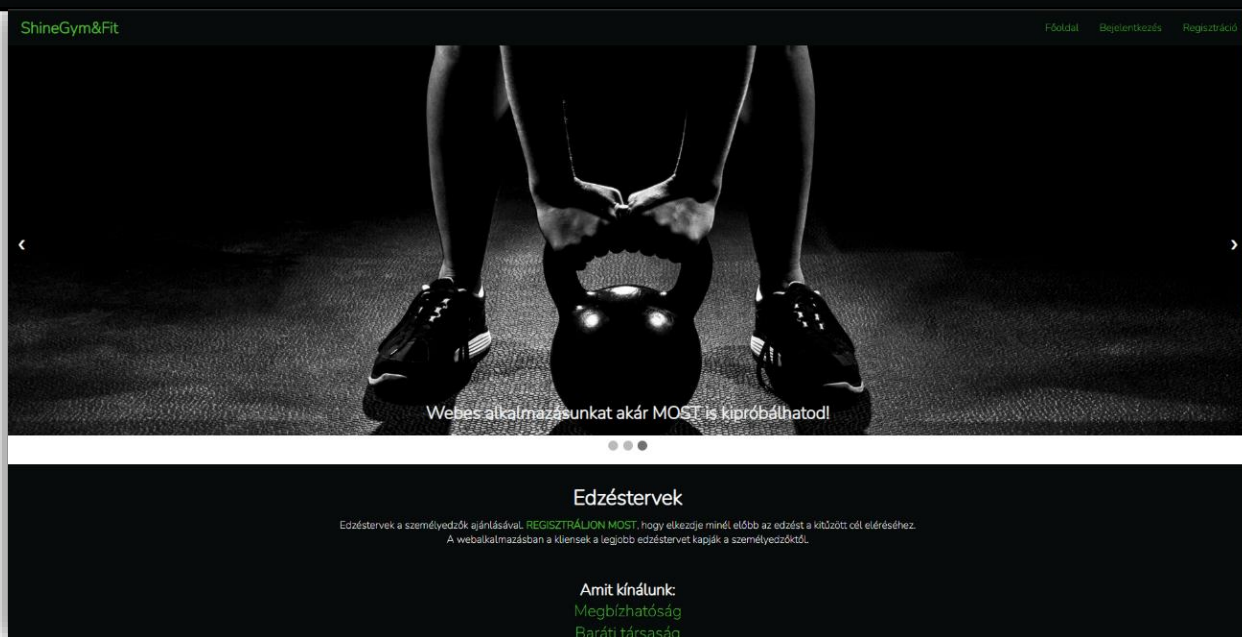


Edzés alkalmazás



Szabó Richárd és Híves Sebastian

Nógrád Megyei Szakképzési

Centrum Szent-Györgyi Albert

Technikum

Szoftverfejlesztő és -tesztelő
technikus

Szakmai azonosító: 5-0613-12-03

Dátum: 2023.04.19.

Tartalom

Bevezető	4
1. Felhasznált technológiák	6
1.1. HTML	6
1.2. CSS.....	6
1.3. JavaScript	6
1.4. MySQL.....	6
1.5. PHP	7
2. Felhasznált szoftverek	8
2.1. Visual Studio Code	8
2.2. XAMPP.....	8
2.3. GitHub.....	8
3. Felhasználói dokumentáció	9
3.1. Főoldal.....	9
3.1.1. Navigációs menü	9
3.1.2. Slideshow/Carousel	10
3.1.3. Edzésterv és étrendösszeállítás leírása.....	10
3.1.4. Számláló (counter).....	10
3.1.5. Lábléc (footer)	10
3.2. Regisztrációs oldal	10
3.3. Bejelentkezés.....	11
3.4. Kezdőlap (Bejelentkezés után).....	12
3.5. Saját profil.....	13
3.6. Edzéstervek	13
3.7. Chat	15
3.8. Kliensek kezelése (edzők számára).....	16
3.9. Edzők kezelése (kliensek számára).....	17

4.	Fejlesztői dokumentáció	18
4.1.	Adatbázis	18
4.1.1.	Szükséges információk	18
4.1.2.	Információk egyedekre osztása	19
4.1.3.	Információs elemek oszlopokká alakítása	20
4.1.4.	Kapcsolatok	21
4.1.5.	Táblák	22
4.1.6.	Adatbázis diagram	27
4.2.	Adatbázis kapcsolat	27
4.3.	Főoldal	28
4.3.1.	Navigációs menü	28
4.3.2.	Carousel	29
4.3.3.	Edzésterv és étrendösszeállítás leírása	30
4.3.4.	Számláló(counter)	31
4.3.5.	Lábléc(footer)	32
4.4.	Chat	32
4.4.1.	Felhasználó lista	32
4.4.2.	Chat felület	33
4.4.3.	Chat üzenetek lekérése adatbázisból	34
4.4.4.	Chat felület JavaScript része	35
4.5.	Kezdőlap	37
4.5.1.	Keresőmező	38
4.5.2.	Lista	39
4.5.3.	Jegyzetelés funkció kliens típusú felhasználóknak	39
4.5.4.	tevTeljTorles() függvény	40
4.6.	sajatProfil.php – Saját profil adatainak lekérdezése	41
4.6.1.	Lekérdezés	41

4.6.2	Kimenet és profilkép összeállítása	41
4.7	felhLista.php – (Felhasználó lista a kezdőlap)	43
4.8	feljegyzesek.php - Feljegyzések lekérése	46
4.8.1	Lapozó	46
4.8.2	Shorter függvény	46
4.8.3	Feljegyzések lekérdezése és kimenet összeállítása	47
4.5.	feljModositas.php – Feljegyzés módosítása/szerkesztése	48
4.9	feljTorlese.php – Feljegyzés törlése	49
4.10	feljRogzítése.php – Feljegyzés rögzítése	49
4.11	edzesterv.php (Edzésterv oldal)	50
4.11.1	Edző típusú profil esetén	50
4.11.2	Kliens típusú profil esetén	52
4.12	Saját profil törlése	54
4.13	. Lehetőség a teljes edzésterv szerkesztésére	56
5	Tesztelés	61
	Összefoglalás	62
	Források	63
	Ábrajegyzék	65

Bevezető

Csapatunk két főből áll, Hives 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 keresgeltünk 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 edzőkkel 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.

1. Felhasznált technológiák

1.1. 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. Ez azt is 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.

1.2. CSS

A CSS (Cascading Style Sheets) határozza meg a különböző elemek megjelenési tulajdonságait. A CSS segítségével tudjuk a weboldalakot 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 gyorsítótárában 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.

1.3. 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.

1.4. 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.

1.5. 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 segítségével egy statikus weboldalt dinamikus webes alkalmazássá alakíthatunk. Beolvashatunk vele 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, pl. egy az oldalra belépett felhasználó milyen adatokat érjen el, milyen jogosultságokat kapjon.

2. Felhasznált szoftverek

2.1. 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 Intheclipse-t, Auto Rename Tag-et.

2.2. XAMPP

Mivel helyi gépen dolgoztunk így szükségünk volt egy webkiszolgálóra. A 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.

2.3. 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.

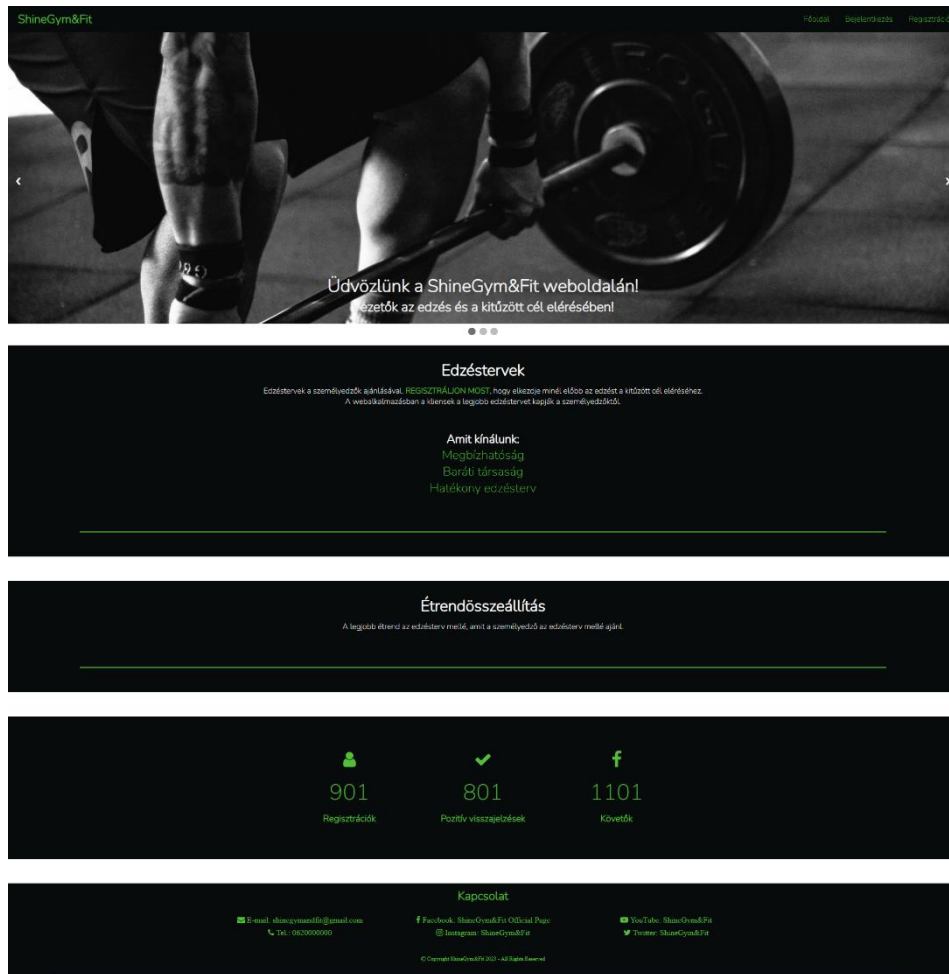
A mi projektünknek Richárd hozott létre a GitHub-on egy repository-t, amelyhez hozzáadta Sebastiant. Így mindketten hozzáférünk, és ahogyan dolgoztunk a programunkon, rendszeresen ún. commit-ot írtunk, amely mentette a változásokat a Github-ra is, mivel összekötöttük a fejlesztői környezettel. Így tudtuk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

3. Felhasználói dokumentáció

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

3.1. Főoldal

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



1) ábra Főoldal

3.1.1. Navigációs menü

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

3.1.2. Slideshow/Carousel

A weboldalon található egy 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 közepén lent a weboldalra látogatók üdvözlésére szolgáló leírás.

3.1.3. 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 a legmegfelelőbb étrendet is megkaphatják az alkalmazásban a regisztráció után, melyet az általuk választott személyi edző készít el részükre.

3.1.4. 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.

3.1.5. 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 feltételeket is.

3.2. 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, illetve ha sérülése van, akár valamilyen betegsége, azt ide felviheti. Ha sikeresen beregisztráltunk, már jelentkezhetünk is be.

Jelentkezzen be!' is at the very bottom." data-bbox="161 180 878 450"/>

2) ábra Regisztrációs felület

3.3. Bejelentkezés

Regisztráljon itt!'." data-bbox="161 540 878 726"/>

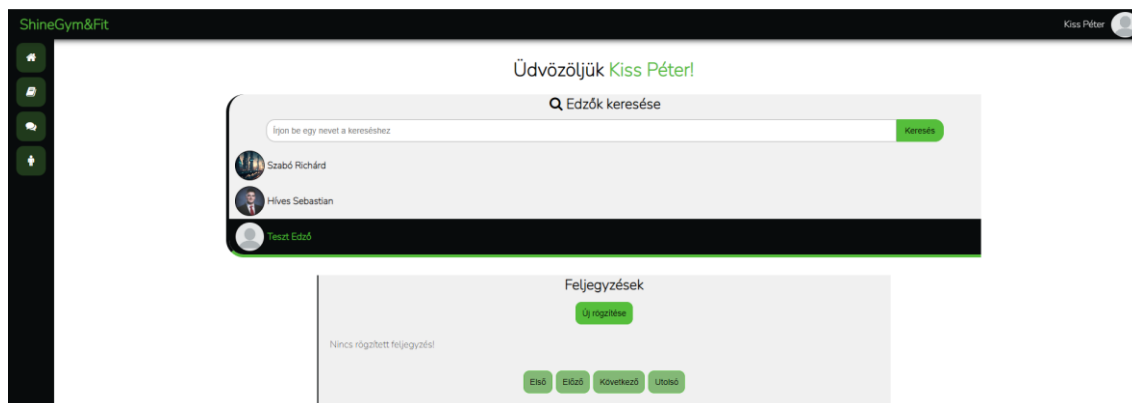
3) ábra Bejelentkezés felülete

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.

3.4. 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.



4) 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.

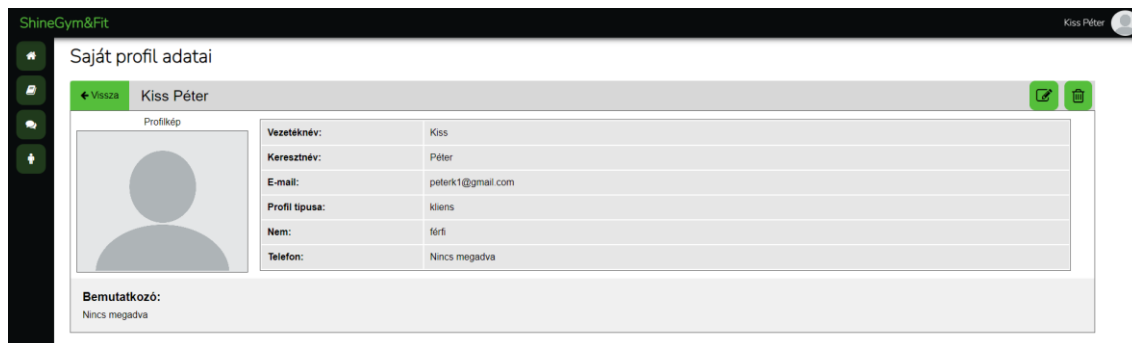


5) ábra Felhasználó adatlapja

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

3.5. Saját profil

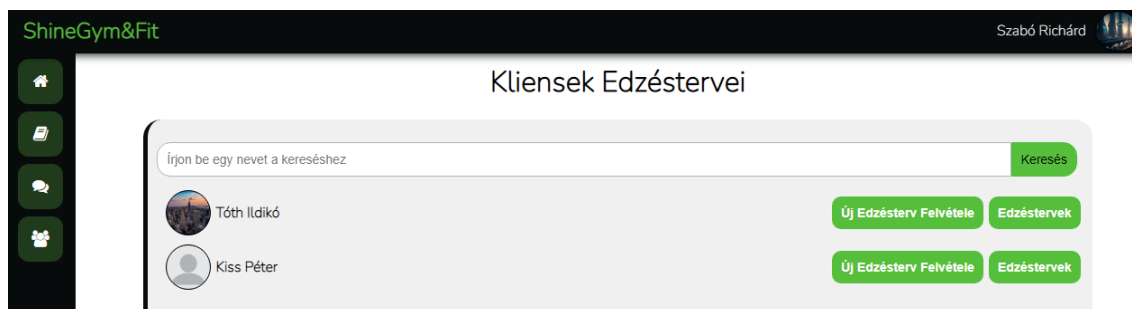
A felhasználók a felső navigációs sávon szereplő profiljukra kattintva tudják az adataikat megtekinteni, mint a feltöltött fotó, név, e-mail cím, profil típusa, nem és a telefonszám. Itt lehetőség van ezen adatok szerkesztésére is a jobb felső sarokban lévő gombra kattintva.



6) ábra Saját profil

3.6. Edzéstervek

Miután küldtünk vagy kaptunk egy felkérést és azt elfogadtuk, az Edzéstervek menüpontban az edző fel tudja venni a kliens edzéstervét és étrendjét. Itt az edző ki tudja választani, hogy melyik kliensének 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).



7) 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.

Étrend

Hány napra szeretne étrendet rögzíteni?*

1

Melyik napra?*

Minden alkalomra

Étrend az adott napra:*

</> | | **B** | *I* | U | | | | | | | | | | | | | | | | | | | |

1. étkezés

– 150 g teljes kiőrlésű kenyér (2 szelet), 1 főtt tojás, 1 nagyobb paradicsom, 100 g csirkemell sonka szelet, 1 evőkanál lenmag olaj

750 kcal, 50 g fehérje, 67 g szénhidrát, 30 g zsír

2. étkezés

– 1 db nagyobb alma

66 kcal, 0,8 g fehérje, 14 g szénhidrát, 0 g zsír

3. étkezés

– 150 g teljes őrlésű tészta, 80 g csirkemell filé, 50 g zsírszegény sajt

700 kcal, 53 g fehérje, 102 g szénhidrát, 8,8 g zsír

4. étkezés (edzés előtt)

– 1 db közepes banán

*A csillaggal *jelölt mezők kitöltése kötelező!*

[Küldés](#)

9) ábra Étrend felvitele

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

ShineGym&Fit Kiss Péter

Kiss Péter Edzéstervei

Keresés [Keresés](#)

Edzésterv neve
Erőnövelő edzésterv

Leírás
Edzésterv heti 3 napraÉtrend a hét minden napjára

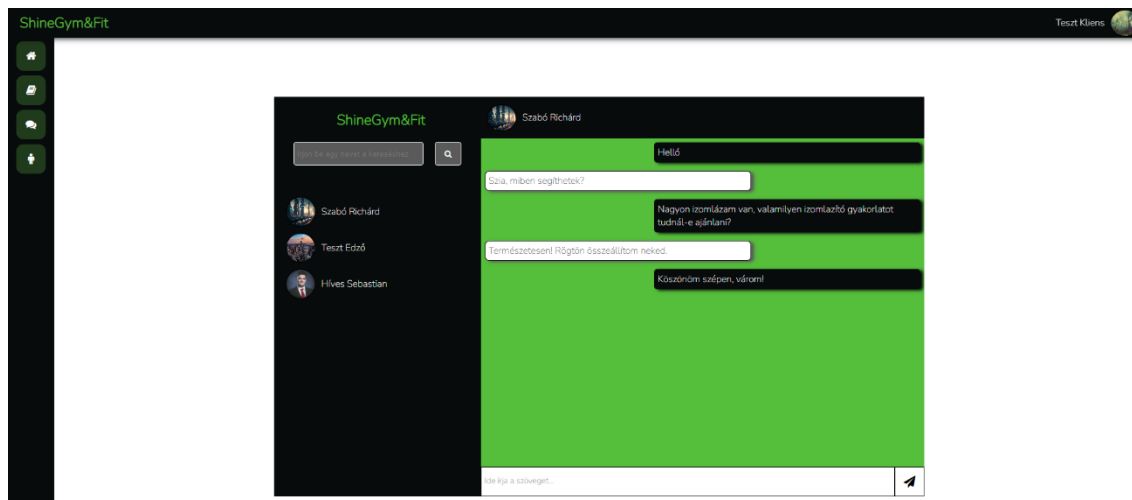
Szabó Richárd

10) ábra Edzéstervek oldal (kliens profil)

3.7. Chat

A Chat menüpontot megnyitva kliensek és edzők tudnak egymással privát üzenetben kommunikálni. A bal oldalon választható ki, hogy kivel szeretnénk csevegni, ha kiválasztottuk és rákattintottunk, a jobb oldalon meg is jelenik az adott felhasználóhoz

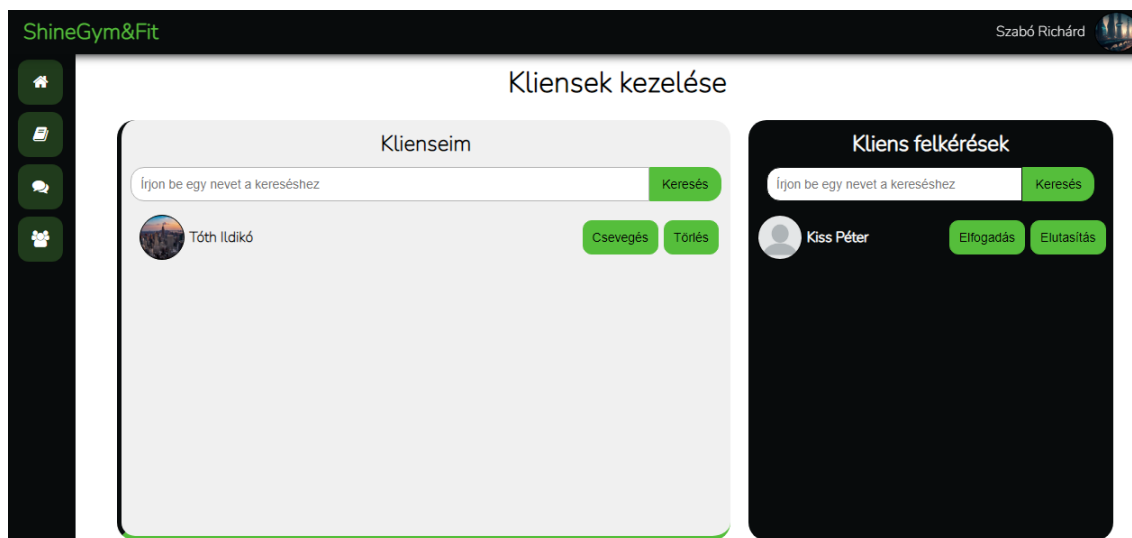
tartozó chat felület. Betöltődnek a korábban küldött üzenetek (ha vannak), megjelenik az üzenet írásához használható mező és a küldés gomb.



11) ábra Chat

3.8. Kliensek kezelése (edzők számára)

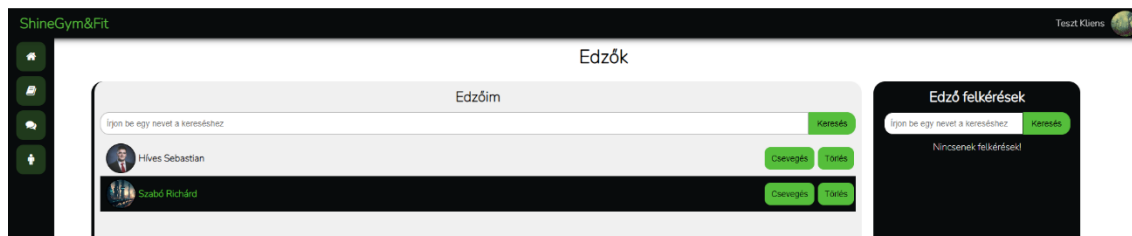
Ha már valaki felkért bennünket, azt a jobb oldali listában fogadhatjuk, illetve utasíthatjuk el a felkérését. Bal oldalon a már meglévő klienseket kezelhetjük, itt van lehetőség a kapcsolat felbontására is a törlés gombra kattintva. A kapcsolat törlésénél törlődnek az ahhoz tartozó edzéstervek és étrendek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



12) ábra Kliensek kezelése

3.9. Edzők kezelése (kliensek számára)

Az edzők kezelése a kliensek számára szolgál, mivel edző edzőt nem kezelhet. Ennek értelmében pedig a felület ugyanaz, 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.



13) ábra Edzők kezelése

4. Fejlesztői dokumentáció

4.1. Adatbázis

Elsőkörben az adatbázisunk működését és tervezési lépéseit mutatjuk be részletesen.

4.1.1. 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: vezetéknev, 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).

Feljegyzések/jegyzetek: Úgy tervezzük, hogy kliens típusú felhasználóknak lehetőséget biztosítunk majd 1 jegyzetelő felületre, ahol felírhatja a napi teendőit az edzéssel vagy étrenddel kapcsolatosan. Mindehhez el kell tárolnunk, hogy melyik napra rögzítette a feljegyzést és magát a jegyzet tartalmát/szövegét.

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. A 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.

4.1.2. 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 vezetéknev, 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.

Feljegyzések egyed: Itt tároljuk a felhasználók jegyzeteit tehát, hogy melyik napra rögzítette (dátum) és a jegyzetet.

4.1.3. Információs elemek oszlopokká alakítása¹

Felhasználók
felhasználó_id
vezetéknév
keresztnev
nem
e-mail
jelszó
profil típus
profilkép
bemutakozó
telefon

Feljegyzések
felj_id
felhasználó_id
dátum
leírás

Üzenetek
üzenet_id
kimenő_id
bejövő_id
üzenet

EdzőKlienskapcs
kapcs_id
küldő_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

¹ kék színnel az elsődleges kulcsot, narancs színnel az idegen kulcsot jelöltük

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

4.1.4. Kapcsolatok

Üzenetek N:1 Felhasználók

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

Feljegyzések N:1 Felhasználók

Egy felhasználóhoz több rögzített feljegyzés, de egy adott feljegyzés 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.

4.1.5. Táblák

felhasznalok		
mező neve	típusa	leírás
felhasznalo_id	int(11)	felhasználó azonosítója
vnev	varchar(50)	vezetéknév
knev	varchar(50)	keresztnev
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)
bemutakozozo	text	bemutakozó szöveg (HTML kód a szöveg formázása miatt)
telefon	varchar(16)	telefonszám (külföldi és belföldi)

felhasznalo_id	vnev	knev	email	jelszo	profil_tipus	kep	nem	bemutakozozo	telefon
1	Szabó	Richárd	pelda@gmail.com	\$2y\$10\$2G7wd.	edző	1679050736.jpeg	1	<p>Személyre szabott edzéssterveket készítek az ügy...	+36201234567
2	Híves	Sebastian	sebihives2001@gmail.com	\$2y\$10\$LMEes.	edző	1680080537.jpeg	1	<p>Több éves edzői szakmai tapasztalattal rendelke...	+4219156297
5	Tóth	Ildikó	tildoko34@gmail.com	\$2y\$10\$eFslm+	kliens	1680164255.jpeg	0		NULL










14) ábra Felhasználók tábla

üzenetek		
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

uzenet_id	kimeno_id	bejovo_id	uzenet
110	3	1	Helló
111	1	3	Szia, miben segíthetek?
112	3	1	Nagyon izomlázam van, valamilyen izomlazító gyakor...
113	1	3	Természetesen! Rögtön összeállítom neked.
114	3	1	Köszönöm szépen, várom!

15) ábra Üzenetek tábla

Feljegyzések		
mező neve	típusa	leírás
felj_id	int(11)	feljegyzés/jegyzet azonosítója
felhasznalo_id	int(11)	felhasználó azonosítója
datum	date	dátum (mikor rögzítették a feljegyzést)
leiras	text	jegyzet (HTML kód a szöveg formázása miatt)

			felj_id	felhasznalo_id	datum	leiras
<input type="checkbox"/>	 Módosítás	 Másolás	 Törölés	1	9	2023-04-04 <p>Mai edzésterv elvégzése</p>
<input type="checkbox"/>	 Módosítás	 Másolás	 Törölés	3	5	2023-04-13 <p>Mai és holnapiétrend</p>
<input type="checkbox"/>	 Módosítás	 Másolás	 Törölés	6	5	2023-04-14 <p>teszt példa</p>

16) ábra Feljegyzések 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_kezdet	datetime	kapcsolat kezdetének dátuma

kapcs_id	kuldo_az	fogado_az	elfogadva	felkeres_datuma	kapcs_kezdet
22	3	1	1	2023-03-28 11:29:06	2023-03-28 11:29:40
23	13	3	1	2023-03-29 11:02:52	2023-03-29 11:03:17
24	13	3	0	2023-03-29 12:21:06	0000-00-00 00:00:00

17) 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

terv_id	neve	leiras	kapcs_id
25	Tömegnövelő edzés	Minden izomcsoportot egyszer kell edzened egy héte...	22
26	Kezdő edzés	Ez az edzés kezdőknek a legideálisabb, illetve...	22
28	Szálkásító edzés	A szálkásítás lényege a test zsírvésszítés maximaliz...	24

18) ábra Terv tábla

edzestervek		
mező neve	típusa	leírás
edzes_id	int(11)	edzés terv azonosítója
nap	varchar(10)	nap (a hét melyik napjára szól az edzés terv)
edzesterv	text	edzés terv leírása (HTML kód a szöveg formázása miatt)
terv_id	int(11)	terv azonosítója

edzes_id	nap	edzesterv	terv_id
29	Hétfő	<p>Mell: Fekvenyomás egykezes súlyzóval vagy rúdd...	25
30	Szerda	Hát-bicepsz Itt nagyon fontos, hogy a 2. napot az...	28
31	Péntek	Váll-láb Váll: Oldalemelés – 4 sorozat, 8-10 ism...	26

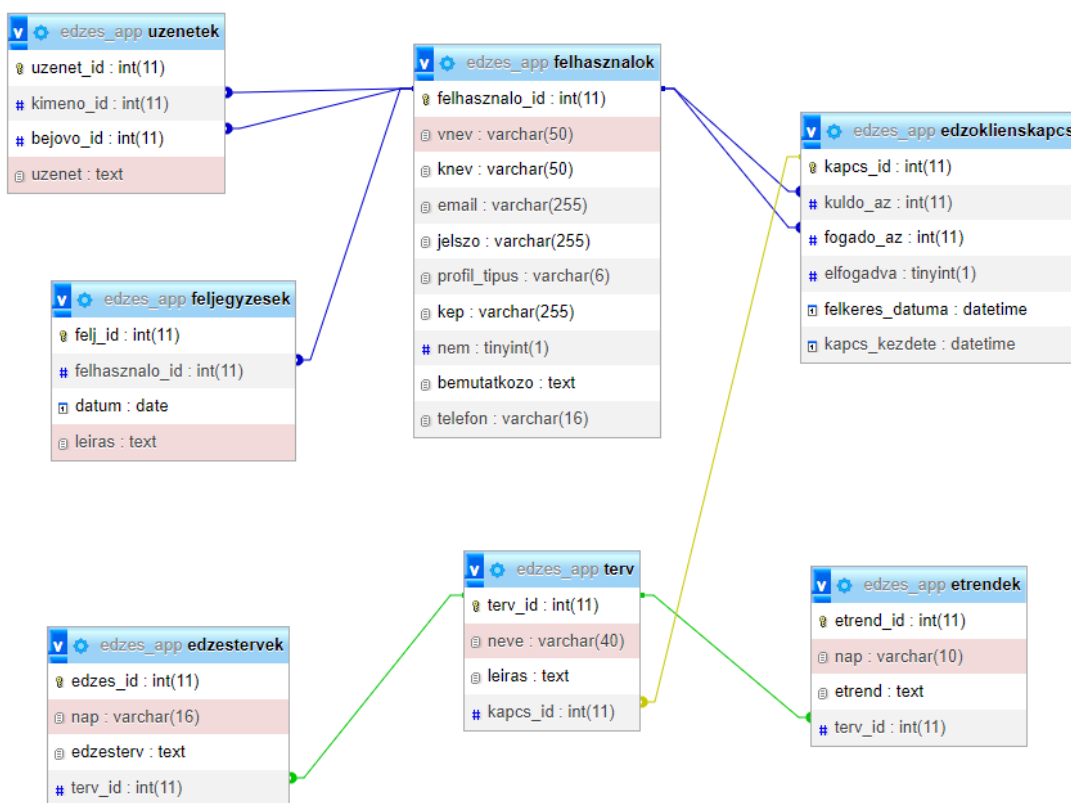
19) ábra Edzéstervek tábla

étrendek		
mező neve	típusa	leírás
étrend_id	int(11)	étrend azonosítója
nap	varchar(10)	nap (a hét melyik napjára szól az étrend)
étrend	text	étrend leírása (HTML kód a szöveg formázása miatt)
terv_id	int(11)	terv azonosítója

étrend_id	nap	étrend	terv_id
26	Hétfő	<p>Táplálkozás a szálkásítás szakaszában Tápanyag...	25
27	Péntek	Napi négy étkezést találsz, reggelit, ebédet, uzso...	28
28	Szerda	Reggeli: Cukormentes fonott kalács Ebéd: ORJALE...	28

20) ábra Étrendek tábla

4.1.6. Adatbázis diagram



21) ábra Adatbázis diagram

4.2. 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 létrehozuk az adatbázis kapcsolatot, paraméterül adjuk az adatbázis server elérési útvonalát (localhost), felhasználónevet (root), jelszót (jelen esetben ez üres) és az adatbázis nevét, amihez kapcsolódni szeretnénk (edzes_app).

```

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 karakterkódolást.

4.3. Főoldal

4.3.1. 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 az 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>
```

4.3.2. 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 áttűnéses 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 egy 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 másodpercenként 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);
}

```

4.3.3. 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 állítottuk be. A margókat és a paddinget 50 pixelesre állítottuk, hogy automatikus legyen. A felső margóhoz 15 pixelt adtunk 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 25 pixeles a betűméret.

4.3.4. 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 ez esetben a counter. A sebességet eltároltuk 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 beírhatjuk az értéket. Az inc változóban Math.floor metódus segítségével kiszámoljuk a számláló értékeit, és a számok, amiket kiszámolt, eredményként itt tárolja el szelekcióval, és ha a count kisebb, mint a target és az inc nagyobb, mint 0, akkor a counter innerTextjébe, 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();
});

```

4.3.5. Lábléc/footer

A footerben található minden elérhetőség. Egy footer tagen belül vannak két osztállyal dívek. 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.

4.4. Chat

Az első sorban ellenőrizzük, hogy van-e bejelentkezett felhasználó. Ha nincs, akkor visszairányítjuk 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 sajátProfil.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");
}

```

4.4.1. 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 vezetéknev és keresztnév összefűzésében.

```

$fosszesChat = mysqli_query($dbconn, "SELECT * FROM felhasznalok WHERE
profil_tipus = '{$felhChat}' AND CONCAT(vnev, ' ', knev) LIKE
'%{$skifejezesChat}%");

```

Majd while ciklus segítségével végigmegyünk a lekérdezett eredményeken és összeállítjuk 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>";
}

```

4.4.2. Chat felület

Ha rákattintunk valakire a bal oldali listában, az hozzáadja az adott felhasználó azonosítóját a webcímhez, amit \$_GET metódus segítségével kiolvassuk belőle. Ha létezik egy ilyen azonosító a webcímbe, 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 a 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 elvezetjük a felhasználót egy hiba.html oldalra, hiszen edző edzővel illetve kliens-klienssel nem cseveghet.

```
} else{
    header("Location: hiba.html");
}

}
```

4.4.3. Chat üzenetek lekérése adatbázisból

Először is elindítjuk a munkamenetet a session_start() függvénnyel majd létrehozuk 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 végigmegyünk 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['kimenno_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;  
}
```

4.4.4. Chat felület JavaScript része

Elsőnek kiválasztjuk és eltároljuk a szükséges DOM elemeket külön konstans változóknak, 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 megjelenítjük 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();}
})

```

4.5 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 végrehajtottunk minden olyan műveletet, amely szükséges az oldal megfelelő megjelenéséhez és használatához.

```

session_start();
if (!isset($_SESSION['felh_id'])) {
    header("Location: ../belep.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 írunk, mert több helyen is megjelenik a weboldalon belül, de ezek segítségével rengeteg időt spórolhatunk, mert nem kell újból leírunk az adott kódrészletet. A require utasítással illesztettük be a kapcsolat.php, sajátProfil.php és a felhLista.php fájlokat. A kapcsolat.php az adatbázishoz való kapcsolódásra szolgál, a sajátProfil.php fájlban az éppen bejelentkezett felhasználó adatait kérjük le. Ennek köszönhetően jelenik meg a felső navigációs sávban a vezetéknev, keresztnév és a profilkép is, de szintén ennek a fájlban 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 sajátProfil.php fájlban hoztuk létre, ezeket is onnan kapja meg.

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

4.5.1 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>
```

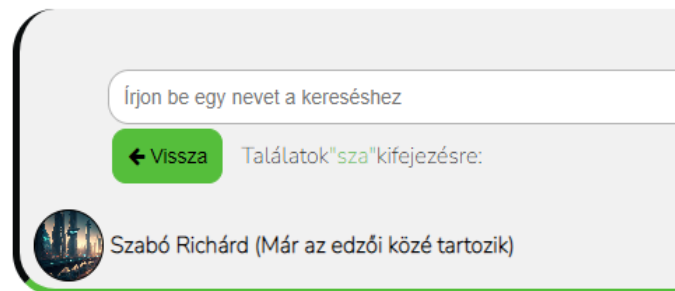
Formon belül hozunk létre egy kereső típusú inputot kifejezes azonosítóval, ez tartalmazza majd a keresendő kifejezést. Az input submit pedig magát a küldés (Keresés) gombját jelenti.

```
<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 kattintásra törli a keresett kifejezést és újra a teljes listát jeleníti meg.

```
$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.



22) ábra Keresés minta

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

?>

</form>
```

4.5.2 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>
```

4.5.3 Jegyzetelés funkció kliens típusú felhasználóknak

Ha a profil típusa kliens, require segítségével beillesztjük a feljegyzések.php tartalmát.

```
<?php

if($_SESSION['p_tipus'] == "kliens"){
    require("leker/feljegyzések.php");
```

Ha a webcímben szerepel egy feljegyzés azonosító, akkor lekérdezzük az adatait az adott azonosítójú feljegyzésnek, és meg is jelenítjük azt. Ennek hatására jelenik meg a teljes formázott jegyzet egy nagyobb ablakban.

```
if(isset($_GET['felj'])){
    $tevAzon = mysqli_real_escape_string($dbconn, $_GET['felj']);
    $valTevsql = mysqli_query($dbconn, "SELECT datum, leiras FROM
feljegyzések WHERE felj_id = '{$tevAzon}' AND felhasznalo_id =
{$_SESSION['felh_id']}");
    $sTev = mysqli_fetch_assoc($valTevsql);
```


Ha van a lekérdezésnek eredménye, tehát a lekérdezett sorok száma nem egyenlő nullával, akkor elkezdjük összeállítani a megjelenítendő felületet a \$tevTeljes változóba.

```
if(mysqli_num_rows($valTevsql) != 0){
    $tevTeljes = "<div class=\"tevTeljes\">
        <p class=\"tdatum\">{$sTev['datum']}</p>
        <div class=\"tleiras\">".
            $sTev['leiras']
        ."</div>
```

Bezárás gomb, a felugró ablak jobb felső sarkába van rendezve, kattintásra elrejtí 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 feljegyzést.

```
<div class=\"tevTeljGombok\">
```

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

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

A módosítás gombra kattintva eljuthatunk a feljModositas.php oldalra, ahol az adott feljegyzés dátumát és leírását is tudjuk módosítani. A webcímhez hozzáfűzzük a feljegyzés azonosítóját, hogy tudjuk, melyiket kell majd módosítani.

```
<button title=\"Módosítás\"
onclick=\"location.href='muveletek/feljModositas.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; } } ?>
```

4.5.4 tevTeljTorles() függvény

```
function tevTeljTorles(datum, tevazon){
    Swal.fire({
        title: 'Biztosan törölni szeretné?',
```

```

        text: "Ezzel törlődik "+ datum +" napján rögzített feljegyzése! 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 `feljTorlese.php` oldalra.

```

    if (result.isConfirmed) {
        location.href = "muveletek/feljTorlese.php?tevaz=" + tevazon;
    })
}

```

4.6 sajátProfil.php – Saját profil adatainak lekérdezése

4.6.1 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'];

```

4.6.2 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 állítjuk 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 adatot 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 azt fűzzük hozzá hogy nincs megadva, egyébként pedig a megadott telefonszámot.

```
$kimenet .= "</tr>
</table>
</div>
<div class=\"bemutakozo\">
<h2>Bemutakozó:</h2>";
strlen($bemutakozo) < 50 ? $kimenet .= "<p>Nincs megadva</p>" : $kimenet .=
"<p>{$bemutakozo}</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, alpból hozzáad pár <p> elemet és néhány sortörést is
, hogy megfelelően jelenjen majd meg a megadott szöveg. Ezért itt nem működik az, ha azt vizsgáljuk, üres-e a változó. Regisztrációkor pedig alpból 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=\"bemutakozo\">
<h2>Bemutakozó:</h2>
<p>{$bemutakozo}</p>
</div>";}
```

4.7 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 \$felhTipus 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_típus'] == "edző"){  
    $felhTipus = "kliens";  
}  
else{  
    $felhTipus = "edző";}
```

Összes megfelelő típusú felhasználó lekérdezése. CONCAT() függvénnyel összefűzzük a vezetéknévet és a keresztnévet, 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 összeállítjuk 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 eltároljuk az elfogadás állapotát (el van-e fogadva vagy sem [true/false érték]) az elfogadva változóban.

```
$fvan = 0;
```

```

if(mysqli_num_rows($fElEredmeny) > 0){
    $fvan = 1;
    $felfogadva = $fElISor['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 felkérve 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>";}

```

4.8 feljegyzések.php - Feljegyzések lekérése

4.8.1 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 alapértelmezésben 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 feljegyzések tábla, ahol a felhasználó azonosító megegyezik a bejelentkezett felhasználó azonosítójával.

```
$total_pages_sql = "SELECT COUNT(felj_id) FROM feljegyzesek WHERE  
felhasznalo_id = {'$_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);
```

4.8.2 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 szóközt 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 visszaadja nekünk az eredeti szöveget.

```
        else{  
            return $text;  
        }  
    }
```

4.8.3 Feljegyzések lekérdezése és kimenet összeállítása

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

```
$sql = "SELECT felj_id, datum, leiras
FROM feljegyzesek
WHERE felhasznalo_id = {$_SESSION['felh_id']}
ORDER BY datum DESC
LIMIT $offset, $no_of_records_per_page";
$res_data = mysqli_query($dbconn,$sql);
```

A feljKimenet változóba elkezdjük összeállítani a feljegyzéseket, hozzáadunk 1 gombot is, ami elvezet egy oldalra (feljRogzítése.php), ahol új jegyzetet lehet rögzíteni.

```
$feljKimenet = "<div class=\"tevk\"><h2>Feljegyzések</h2>
<button onclick=\"location.href='feljRogzítése.php'\">Új rögzítése</button>
<div class=\"feljegyzesek\">";
```

Ha a lekérdezésünk tartalmaz sorokat, tehát van rögzített feljegyzése a bejelentkezett felhasználónak, akkor minden ilyen hozzáfűzünk a feljKimenet 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 feljegyzés azonosítóját is, alap esetben ugyanis felülrná a webcímet és újra az első oldalra vezetne, amikor rákattintunk egy jegyzetre. Egyébként, ha nincs meghatározva lap, akkor csak a feljegyzés azonosítóját adjuk a webcímhez, amivel a teljes jegyzet fog majd megjelenni.

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

A felj osztályt tartalmazó div-re kattintva megjelenik a teljes jegyzet egy felugró ablakban.

```
$feljKimenet .= "<div class=\"felj\" onclick=\"location.href='{ $link }'\">
<p class=\"tdatum\">{$tDatum}</p>
```


A feljegyzés 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 feljegyzés szöveg fog megjelenni a felhasználónak.

```
else{
    $feljKimenet .= "<p id='nincsRtev'>Nincs rögzített feljegyzés!</p>";
}
$feljKimenet .= "</div>";
```

4.5. feljModositas.php – Feljegyzés módosítása/szerkesztése

A modID változóban eltároljuk a webcímből kiolvasott azonosítót.

```
$modID = mysqli_real_escape_string($dbconn, $_GET['tevid']);
```

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

```
$sql = mysqli_query($dbconn, "SELECT datum, leiras FROM feljegyzesek
WHERE felj_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 frissítjük az adatbázisban az adatokat.

```
if(isset($_POST['modosit'])){
    $pdatum = $_POST['dat'];
    $pleiras = $_POST['leir'];

    $sqlUpdate = mysqli_query($dbconn, "UPDATE feljegyzesek SET datum =
'{$pdatum}', leiras = '{$pleiras}' WHERE felj_id = {$modID}");
    header("Location: ../kezdolap.php");
}
```

4.9 feljTorlese.php – Feljegyzés törlése

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

A \$storlendo változóban eltároljuk a webcímből kiolvasott feljegyzés azonosítóját.

```
$storlendo = mysqli_real_escape_string($dbconn, $_GET['tevaz']);
```

Aztán végrehajtunk egy mysql parancsot, amiben töröljük az adott feljegyzést a feljegyzések táblából, ahol a feljegyzés azonosítója megegyezik a webcímből kiolvasott azonosítóval.

```
$sql = mysqli_query($dbconn, "DELETE FROM feljegyzesek WHERE felj_id = {$storlendo}");
```

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

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

4.10 feljRogzitesi.php – Feljegyzés rögzítése

Ha rányomtunk a rögzítés gombra, tehát létezik a \$_POST['rogzites'] akkor kiolvassuk a dátumot és az azonosítót, majd eltároljuk azokat egy-egy 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 feljegyzések táblába.

```
$sql = mysqli_query($dbconn, "INSERT INTO feljegyzesek (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>

```

4.11 edzesterv.php (Edzésterv oldal)

4.11.1 Edző típusú profil esetén

Mivel ennél az oldalnál 2 különböző felületet kell megjelenítenünk az edző és kliens felhasználóknak, egy \$_SESSION változóból kiolvassuk a már korábban (bejelentkezésnél) eltárolt profil típusát. Ez alapján 2 jól elkülöníthető felületet tudunk megjeleníteni 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 végigmegyünk a találatokon, ellenőrizzük, hogy nem nulla sorból áll-e 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 keresztnévet, 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 állítjuk 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 két gombot, ahol az egyiknél új edzéstervet/étrendet rögzíthetünk a kiválasztott kliensüknek, a másiknál pedig a már meglévő bejelentkezett profil által megírt edzéstervek láthatóak (ha vannak). 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=\". $sor2[$kerdezo] . \"\">Új Edzésterv Felvétele</button>  
    <button onclick=\"location.href='etervM.php?kliens=\". $sor2[$kerdezo] . \"\">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>\";  
}
```

4.11.2 Kliens típusú profil esetén

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

```
else{
```

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

```
$felulet = "<h1>{$vnev} {$sknev} 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}");
```

```
$tervKi = "<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 küldő 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 keresztnévét.

```
$edzoneve = mysqli_query($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 „\$tervKi” változóba fűztünk össze.

```
$tervKi .= "<a href=\"teljeset.php?edzesterv={$etID}\">
<div class=\"edzesterv\">
<div class=\"etneve\">
```

```

        <p>Edzésterve 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="eTervSegitsege">
        <p class="etSegitsegeC">Önnek még nincs egy edzésterve sem!</p>
        <p>Edzésterve 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 edzésterve / étrendet szeretne, korábbi sérülések,
betegségek stb.)</li>
            <li>Amikor mindezt megbeszélték, az edző megírja a személyre
szabott edzésterve/étrendet, amint készen van az edzéstervek menüpontban fogja tudni
megtekinteni.</li>
        </ol>
    </div>
    </div>";
}
$etervKi .= "</div>";
$felulet .= $etervKi;
}

```

4.12 Saját profil törlése

Úgy döntöttünk, lehetőséget adunk a felhasználóknak a saját profiljuk törlésére, ha már nem szeretnék többet használni az alkalmazást, vagy esetleg törölnék a régi profiljukat és újat hoznának létre valamiért. Ehhez elsősorban a sajátProfil.php fájlunkban bővítettük a \$kimenet változót egy gombbal, ahol onclick eseményre reagálva hívjuk meg a sajátProfTorles() nevű függvényt.

```
<button id=\"btnProfTorles\" onclick=\"sajatProfTorles()\" title=\"Saját profil törlése\"><i class=\"fa fa-trash-o\" aria-hidden=\"true\"></i></button>
```

Aztán létrehoztuk a sProfilTorles.php fájlt, ami az adatbázisból való törlésért felelős. Itt miután elindítottuk a munkamenetet a session_start() függvénnyel, egy if szerkezetben ellenőrizzük, hogy létezik-e a \$_SESSION['felh_id'], amit belépéskor azonnal létre is hozunk és értékül adjuk neki a bejelentkezett felhasználó azonosítóját. Tehát ha nem létezik, akkor header() függvény segítségével visszairányítjuk a felhasználót a bejelentkezés oldalára. Egyébként miután létrehoztuk a kapcsolatot az adatbázissal, végrehajtunk egy mysql parancsot, amiben töröljük a felhasználót a sessionben meghatározott azonosító alapján, aztán elnavigálunk az alkalmazás fő oldalára.

```
<?php
session_start();
if(!isset($_SESSION['felh_id'])){
    header('Location: ../../belepes.php');
} else{
    require(\"../kapcsolat.php\");
    mysqli_query($dbconn, \"DELETE FROM felhasznalok WHERE felhasznalo_id = {$_SESSION['felh_id']} \");
    header('Location: ../../index.html');
}
?>
```

Végül JavaScriptben megírtuk a felugró ablakot, ami a törlés megerősítésére szolgál. Itt felhívjuk a felhasználó figyelmét, hogy minden adata törlődni fog az adatbázisból és nem fogja tudni visszaállítani azt.

```
function sajátProfTorles(){
    Swal.fire({
        title: 'Biztosan törölni szeretné a profilját?',
        text: \"Ez a művelet nem visszavonható! Minden adata törlődni fog az alkalmazásból, beleértve az edzésterveit és az üzeneteit is!\",
        icon: 'warning',
        showCancelButton: true,
        confirmButtonColor: '#ff0000',
        cancelButtonColor: '#55be3b',
        confirmButtonText: 'Törlés',
        cancelButtonText: 'Mégsem',
        backdrop: `rgba(120,0,0,0.4)` ,
    }).then((result) => {
```

Ha megerősítette a törlést, elnavigálunk a korábban említett profil törlésére szolgáló php oldalra.

```
if (result.isConfirmed) {
    Swal.fire(
        location.href = \"muveletek/sProfilTorles.php\", )}
```


4.13. Lehetőség a teljes edzésterv szerkesztésére

Az edző, miután felvitt a kliensének egy edzéstervet, mindeddig csak az edzésterv nevének illetve leírásának szerkesztésére volt lehetősége. Ezek mellett most már lehetőség van maga az edzésterv és az étrend szerkesztésére, valamint a napok felcserélésére is. Ehhez az etszerk.php nevű fájlinkat kellett alaposan átírnunk, illetve bővítenünk.

Először ellenőrizzük, hogy létezik-e a getből kiolvasott edzésterv azonosító (\$_GET['etid']), ha nem létezik elnavigálunk egy hiba.html oldalra. Egyébként, ha létezik, kapcsolódunk az adatbázishoz és kiolvassuk a webcíméből a módosítandó terv azonosítóját.

```
if(!isset($_GET['etid'])){  
    header("Location: hiba.html");  
} else{  
    require("kapcsolat.php");  
  
    $etID = $_GET['etid'];
```

A kiolvasás után azonnal ellenőrizzük, hogy a bejelentkezett felhasználóhoz (az edzőhöz) tartozik-e a terv, amit szerkeszteni akar, ugyanis más edzők által írt edzésterveket/étrendeket nem módosíthat. Ezzel próbáljuk kizárni azt a lehetőséget, hogy a felhasználó átírja a webcímbe szereplő azonosítót, ezzel illetéktelenül férjen hozzá és módosítson más edzőkhöz tartozó edzésterveket/étrendeket. Az ellenőrzést egy lekérdezés segítségével valósítjuk meg, amiben lekérjük az összes olyan tervet, ahol a bejelentkezett edzőhöz tartozik a terv. Ezt az edzoklienskapcs táblából tudhatjuk meg, ahol megegyezik a küldő vagy fogadó azonosító a sessionben meghatározott felhasználó azonosítóval.

```
$tervEll = mysqli_query($dbconn, "SELECT terv_id FROM terv INNER JOIN  
edzoklienskapcs ON terv.kapcs_id = edzoklienskapcs.kapcs_id WHERE kuldo_az =  
{$_SESSION['felh_id']} AND terv_id = {$etID} OR fogado_az =  
{$_SESSION['felh_id']} AND terv_id = {$etID}");
```

Egy if szerkezetben a mysqli_num_rows() függvénnyel ellenőrizzük, hogy a lekérdezésünk tartalmaz-e sorokat, tehát ha valóban hozzá tartozik az edzésterv, amit módosítani szeretne.

```
if(mysqli_num_rows($tervEll) != 0){
```

A \$modForm változóban elkezdjük összeállítani a formot, ami az adatok módosítására szolgál majd.

```
$modForm = "<form method=\"post\">";
```

Ahhoz, hogy az input és textarea mezőkben szerepeljenek a korábbi adatok, előbb le kell azokat kérdeznünk az adatbázisból. Először a terv táblából kérdezzük le a tervnek megadott nevet illetve a leírást, majd eltároljuk azokat 1-1 változóban.

```
//terv adatai
$eredmeny = mysqli_query($dbconn, "SELECT neve, leiras FROM terv
WHERE terv_id = { $SetID }");
$sor = mysqli_fetch_assoc($eredmeny);
$SetNeve = $sor['neve'];
$SetLeiras = $sor['leiras'];
```

Miután ezeket lekérdeztük hozzá is fűzzük a \$modForm változóhoz az ezekhez szükséges label, input, és textarea elemeket.

```
$modForm .= "<div class=\"mezo\">
    <label for=\"etneve\">Terv neve:</label>
    <input type=\"text\" value=\"{ $SetNeve }\" name=\"etneve\"
id=\"etneve\">
</div>
<div class=\"mezo\">
    <label for=\"etleiras\">Leírás:</label>
    <textarea name=\"etleiras\" id=\"etleiras\">{ $SetLeiras }</textarea>
</div>";
//-----
```

Egy \$napok tömbben eltároljuk a hét napjait és a minden alkalomra lehetőséget, amiből majd kiválaszthatja az edző, hogy melyik napra szeretné módosítani az adott edzéstervet vagy étrendet.

```
$napok = array("Minden alkalomra", "Hétfő", "Kedd", "Szerda", "Csütörtök",
"Péntek", "Szombat", "Vasárnap");
```

Lekérdezzük a tervhez tartozó összes edzéstervet, majd ismét hozzáfűzzük a \$modForm változóhoz a szükséges mezőket.

```
//edzésterv
$edzesterv = mysqli_query($dbconn, "SELECT nap, edzesterv FROM
edzestervek WHERE terv_id = { $SetID }");
```

A \$dbEterv változóban eltároljuk, hogy hány edzésterv tartozik a tervhez, majd ellenőrizzük, hogy biztosan tartozik hozzá legalább egy edzésterv.

```
$dbEterv = mysqli_num_rows($edzesterv);
if($dbEterv != 0){
```

Létrehozuk az \$i változót, ami az edzésnapok elkülönítésére szolgál majd.

Aztán while ciklus segítségével végigmegyünk a lekérdezett eredményeken és ismét a \$modForm változóhoz fűzzük a form további tartalmát.

```
$i = 1;
$modForm .= "<h2>Edzésterv</h2>";
while($eterv = mysqli_fetch_assoc($edzesterv)){
    $modForm .= "<div class=\"mezo\">
```

```

        <input type="hidden" name="edznapdb" id="edznapdb"
value="\{$dbEterv}\">
        <select name="edznap{$i}" id="edznap{$i}">;

```

Közben foreach ciklus használatával fűzzük hozzá a választható opciókat a kimenethez a korábban meghatározott \$napok tömbből. Az if-else szerkezet arra szolgál, hogy meg tudjuk határozni, melyik opció volt korábban kiválasztva. Ezt a lekérdezett \$eterv['nap'] segítségével tudjuk ellenőrizni, ha megegyezik az éppen kiírandó adattal, az option mezőt kiválasztottra állítjuk.

```

        foreach ($napok as $nap) {
            if($nap == $eterv['nap']){
                $modForm .= "<option value="\{$nap}\\"
selected>{$nap}</option>";
            }
            else{
                $modForm .= "<option value="\{$nap}\">{$nap}</option>";
            }
        }
        $modForm .= "</select>
        <textarea name="edzterv{$i}"
id="edzterv{$i}">{$eterv['edzesterv']}</textarea>
        </div>";

```

A while ciklus végén, amíg végigmegy az eredményeken a függvény, mindig hozzáadunk egyet a \$i változóhoz, hogy megfelelő nevet és azonosítót kapjanak a létrehozott mezők.

```

        $i++;
    }
}
//---

```

Az edzéstervekhez hasonlóan fűzzük hozzá a \$modForm változóhoz az étrendek form kimenetét is. Lekérdezzük az összes olyan étrendet, ahol a terv azonosítója megegyezik a \$etID értékével.

```

        $etrend = mysqli_query($dbconn, "SELECT nap, etrend FROM etrendek
WHERE terv_id = {$etID}");
        $dbEtrend = mysqli_num_rows($etrend);

```

Ha a lekérdezés tartalmaz sorokat, azaz legalább egy étrend tartozik az adott tervhez, akkor szintén while ciklussal végigmegyünk az eredményeken és hozzáfűzzük a kimenethez az étrend szerkesztéséhez szükséges mezőket is.

```

        if($dbEtrend != 0){
            $j = 1;
            $modForm .= "<h2>Étrend</h2>";
            while($etr = mysqli_fetch_assoc($etrend)){
                $modForm .= "<div class="mezo">

```

```

        <input type="hidden" name="etrenddb" id="etrenddb"
value="{ $dbEtrend} ">
        <select name="etnap{ $j} " id="etnap{ $j} ">";

```

Közben itt is foreach-el fűzzük hozzá a select elemhez az option elemeket.

```

        foreach ($napok as $nap) {
            if($nap == $etr['nap']){
                $modForm .= " <option value='{ $nap} '"
selected>{ $nap} </option>";
            } else{
                $modForm .= " <option value='{ $nap} '">{ $nap} </option>";
            }
        }
        $modForm .= "</select>
        <textarea name='etrend{ $j} '"
id="etrend{ $j} '">{ $etr['etrend']} </textarea>
        </div>";
        $j++;
    }
}

```

Miután az edzésterv és étrend szerkesztéséhez szükséges mezőket is hozzáfűztük a \$modForm változóhoz, leírjuk még a Mentés gomb submit típusú input mezőjét is, és lezárjuk a formot.

```

        $modForm .= " <input type='submit' name='kuldes' id='kuldes'
value='Mentés' ">
        </form>";

```

Ha létezik a \$_POST['kuldes'], tehát ha rányomtunk a mentés gombra és elküldtük a form tartalmát, akkor létrehozunk egy-egy változót a terv nevének és leírásának, majd eltároljuk benne azokat.

```

        if(isset($_POST['kuldes'])){
            $Neve = $_POST['etneve'];
            $Leirasa = $_POST['etleiras'];

```

Ellenőrizzük, hogy a név biztosan nem üres-e. Ha üres, akkor egy hibaüzenetet állítunk össze. Egyébként végre is hajtjuk a mysql parancsot, ami frissíti az adatbázisban a terv nevét illetve annak leírását.

```

        if($Neve == ""){
            $nevhiba = "<p style='color: red; font-weight: bold; font-size: 18px'>A
név mező nem lehet üres!</p>";
        } else{
            $update = mysqli_query($dbconn, "UPDATE terv SET neve =
'{ $Neve}', leiras = '{ $Leirasa}' WHERE terv_id = { $etID}");

```

Ez után töröljük az ezen tervhez tartozó összes edzéstervet az adatbázisból. Ezek után ellenőrizzük, hogy melyik post metódusban küldött adat létezik. Szóval, ha létezik egy valamely napra megadott edzésterv, annak adatait beszúrjuk az edzéstervek táblába.

```
$edzDelete = mysqli_query($dbconn, "DELETE FROM edzestervek
WHERE ter_v_id = {$SetID}");
```

```
isset($_POST['edznap1'])    &&    isset($_POST['edzter_v1'])    ?
mysqli_query($dbconn, "INSERT INTO edzestervek (nap, edzesterv, ter_v_id) VALUES
('$_POST['edznap1']','$_POST['edzter_v1']','{$SetID}'))" : "";
isset($_POST['edznap2'])    &&    isset($_POST['edzter_v2'])    ?
mysqli_query($dbconn, "INSERT INTO edzestervek (nap, edzesterv, ter_v_id) VALUES
('$_POST['edznap2']','$_POST['edzter_v2']','{$SetID}'))" : "";
isset($_POST['edznap3'])    &&    isset($_POST['edzter_v3'])    ?
mysqli_query($dbconn, "INSERT INTO edzestervek (nap, edzesterv, ter_v_id) VALUES
('$_POST['edznap3']','$_POST['edzter_v3']','{$SetID}'))" : "";
isset($_POST['edznap4'])    &&    isset($_POST['edzter_v4'])    ?
mysqli_query($dbconn, "INSERT INTO edzestervek (nap, edzesterv, ter_v_id) VALUES
('$_POST['edznap4']','$_POST['edzter_v4']','{$SetID}'))" : "";
isset($_POST['edznap5'])    &&    isset($_POST['edzter_v5'])    ?
mysqli_query($dbconn, "INSERT INTO edzestervek (nap, edzesterv, ter_v_id) VALUES
('$_POST['edznap5']','$_POST['edzter_v5']','{$SetID}'))" : "";
isset($_POST['edznap6'])    &&    isset($_POST['edzter_v6'])    ?
mysqli_query($dbconn, "INSERT INTO edzestervek (nap, edzesterv, ter_v_id) VALUES
('$_POST['edznap6']','$_POST['edzter_v6']','{$SetID}'))" : "";
isset($_POST['edznap7'])    &&    isset($_POST['edzter_v7'])    ?
mysqli_query($dbconn, "INSERT INTO edzestervek (nap, edzesterv, ter_v_id) VALUES
('$_POST['edznap7']','$_POST['edzter_v7']','{$SetID}'))" : "";
```

(Miután az edzéstervekkel végeztünk, az étrendekkel is ugyanígy járunk el.)

Létrehozunk egy session globális változót, amit majd egy megerősítő üzenethez használunk fel. Aztán elnavigáljuk a felhasználót egy oldalra, ahol a már módosított edzéstervet tekintheti át teljes formájában.

```
$_SESSION['sikeresMod'] = "<p>Sikeres módosítás!</p>";
header("Location: teljeset.php?edzesterv={$SetID}");
}
}
```

Ha az ellenőrzés nem tartalmaz sorokat, tehát a szerkeszteni kívánt edzésterv nem hozzánk tartozik, akkor elnavigáljuk a felhasználót a hiba.html nevű oldalra.

```
} else{
    header("Location: hiba.html");
}
}
```

5 Tesztelés

A weboldalaink HTML kódrészleteit a <https://validator.w3.org/> oldalon 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/> oldalon 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 változóban, 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> oldalon is ellenőriztük. Habár egyrészt, a Visual Studio Code, ha hibát észlel a PHP kódrészletekben, akkor 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 a weboldalon szintén azonnal megjelenik a hibaüzenet, 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 problémát 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 továbbfejleszteni 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 a mi munkánknak 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 szlovákiai magyar állampolgár, így mindenképpen érdemes lenne kihasználnunk a szlovák nyelv tudását is. Mivel mindketten tanultunk angolul, így az lenne a harmadik választható nyelv majd az oldalunkon.

Az elkészített projektmunkánk elérhetősége a GitHubon:

https://github.com/SzaboRichard01/2023_Zaroprojekt

A Nethely ingyenes tárhelyen:

<http://shinegymandfit.nhely.hu/index.html>

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 CodingLab: Responsive Side Navigation Bar in HTML CSS & JavaScript
2021.04.14. Link: <https://www.codinglabweb.com/2021/04/responsive-side-navigation-bar-in-html.html>
- 4 Paritosh Pandey: How to create Pagination with PHP and MySql 2022.04.05.
Link: <https://www.myprogrammingtutorials.com/create-pagination-with-php-and-mysql.html>
- 5 Redactor: <http://redactorjs.com/>
- 6 StackOverflow: Check if the array contain multiple same values in php Link:
<https://stackoverflow.com/questions/45182758/check-if-the-array-contain-multiple-same-values-in-php>
- 7 StackOverflow: How to reload current page without losing any form data? Link:
<https://stackoverflow.com/questions/17591447/how-to-reload-current-page-without-losing-any-form-data>
- 8 StackOverflow: How to use PHP's password_hash to hash and verify passwords
Link: <https://stackoverflow.com/questions/30279321/how-to-use-phps-password-hash-to-hash-and-verify-passwords>
- 9 StackOverflow: Short Text, PHP Link:
<https://stackoverflow.com/questions/7348103/short-text-php>
- 10 StackOverflow: Textarea editor Redactor. Insert value with jQuery Link:
<https://stackoverflow.com/questions/19590297/textarea-editor-redactor-insert-value-with-jquery>
- 11 SweetAlert v.11.7.3 2023. Alerts Link: <https://sweetalert2.github.io/>
- 12 W3School : How TO - Trigger Button Click on Enter Link:
https://www.w3schools.com/howto/howto_js_trigger_button_enter.asp
- 13 W3School: How TO - Slideshow Link:
https://www.w3schools.com/howto/howto_js_slideshow.asp
- 14 WEBiskola: Mi az a PHP? A PHP fogalma és bemutatása Link:
<https://webiskola.hu/php-ismeretek/mi-az-a-php-fogalma-bemutatasa/>

- 15 Wikipedia: CSS Link: <https://hu.wikipedia.org/wiki/CSS>
- 16 Wikipedia: GitHub Link: <https://hu.wikipedia.org/wiki/GitHub>
- 17 Wikipedia: HTML Link: <https://hu.wikipedia.org/wiki/HTML>
- 18 Wikipedia: MySQL Link: <https://hu.wikipedia.org/wiki/MySQL>
- 19 Wikipedia: PHP Link: <https://hu.wikipedia.org/wiki/PHP>
- 20 Wikipedia: Visual Studio Code Link:
https://hu.wikipedia.org/wiki/Visual_Studio_Code
- 21 Wikipedia: XAMPP Link: <https://hu.wikipedia.org/wiki/XAMPP>

Ábrajegyzék

1)	ábra Főoldal	9
2)	ábra Regisztrációs felület	11
3)	ábra Bejelentkezés felülete	11
4)	ábra Kezdőlap	12
5)	ábra Felhasználó adatlapja	12
6)	ábra Saját profil	13
7)	ábra Edzésterv oldal (edző profil)	13
8)	ábra Edzésterv felvitele	14
9)	ábra Étrend felvitele	15
10)	ábra Edzésterv oldal (kliens profil)	15
11)	ábra Chat	16
12)	ábra Kliensek kezelése	16
13)	ábra Edzők kezelése	17
14)	ábra Felhasználók tábla	22
15)	ábra Üzenetek tábla	23
16)	ábra Feljegyzések tábla	24
17)	ábra Edző-Kliens kapcsolatok tábla	24
18)	ábra Terv tábla	25
19)	ábra Edzéstervek tábla	26
20)	ábra Étrendek tábla	26
21)	ábra Adatbázis diagram	27
22)	ábra Keresés minta	39