|  |  |
| --- | --- |
| Kanyargós út fákkal  Edzés alkalmazás | Szabó Richárd és Híves Sebastian |

Tartalom

[Bevezető 3](#_Toc129686343)

[1. Felhasznált technológiák 5](#_Toc129686344)

[1.1. HTML 5](#_Toc129686345)

[1.1. CSS 5](#_Toc129686346)

[1.2. JavaScript 5](#_Toc129686347)

[1.3. MySQL 6](#_Toc129686348)

[1.4. PHP 6](#_Toc129686349)

[2. Felhasznált szoftverek 7](#_Toc129686350)

[2.1. Visual Studio Code 7](#_Toc129686351)

[2.2. XAMPP 7](#_Toc129686352)

[3. Felhasználói dokumentáció 8](#_Toc129686353)

[3.1. Főoldal 8](#_Toc129686354)

[3.1.1. Navigációs menü 8](#_Toc129686355)

[3.1.2. Slideshow Carousel 8](#_Toc129686356)

[3.1.3. Edzésterv és étrendösszeállítás leírása 8](#_Toc129686357)

[3.1.4. Számláló(counter) 8](#_Toc129686358)

[3.1.5. Lábléc(footer) 8](#_Toc129686359)

[3.2. Regisztrációs oldal 9](#_Toc129686360)

[3.3. Bejelentkezés 9](#_Toc129686361)

[3.4. Kezdőlap a bejelentkezettek számára 9](#_Toc129686362)

[3.5. Saját profil 9](#_Toc129686363)

[3.6. Edzéstervek 9](#_Toc129686364)

[3.7. Chat 10](#_Toc129686365)

[3.8. Kliensek kezelése(edzők számára) 10](#_Toc129686366)

[3.9. Edzők kezelése(kliensek számára) 10](#_Toc129686367)

[4. Fejlesztői dokumentáció 10](#_Toc129686368)

[4.1. Adatbázis kapcsolat 10](#_Toc129686369)

[4.2. Főoldal 11](#_Toc129686370)

[4.2.1. Navigációs menü 11](#_Toc129686371)

[4.2.2. Slideshow Carousel 12](#_Toc129686372)

[4.2.3. Edzésterv és étrendösszeállítás leírása 15](#_Toc129686373)

[4.2.4. Számláló(counter) 20](#_Toc129686374)

[4.2.5. Lábléc(footer) 21](#_Toc129686375)

[4.3. Chat 22](#_Toc129686376)

[5. Összefoglalás 34](#_Toc129686377)

# Bevezető

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

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

Ezért egy olyan alkalmazást készítettünk, amely edzések kivitelezésében nyújt segítséget. Lehetőség van a napi tevékenység rögzítésére, így később bármikor megtekinthető, hogy melyik nap milyen edzést, mozgást végeztünk. Ezen felül előre összeállított edzésprogramok közül válogathatunk erőszint és szükséges eszközök alapján.

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

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

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

# Felhasznált technológiák

## HTML

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

## CSS

A CSS (Cascading Style Sheets) adja a leírókódon belüli elemek megjelenési tulajdonságát. Létrejöttének oka, hogy megnőtt az igény az egyre bonyolultabb elemekre, formázásokra. A CSS segítségével tudjuk a weboldalakat formázni, betűtípusokat, színeket állíthatunk be, margókat, távolságokat, elhelyezni a weboldalon az elemeket.

Egy stíluslapot több oldalhoz is hozzá tudunk kapcsolni. Ha megnyitunk egy stíluslapot a böngészőben, akkor az a cache (gyorsítótár) mappába menti le a stíluslapot. Így tehát ha máskor a weboldalt megnyitjuk, nem kell várni, hogy a CSS fájl betöltsön, mert a gyorsítótár a böngészőben már eltárolta azt. 1996-ban készült el ez a nyelv. Jelenleg a CSS 3-as verzió a legújabb, amely már 1999 óta van forgalomban.

A munkánk során 4 stíluslapot készítettünk, ahol a weboldalak elrendezését flexbox-al és CSS-GRID-el oldottuk meg.

## JavaScript

A JavaScript egy népszerű programozási nyelv, amit a weboldalak dinamikus működtetéséhez használunk. Általában HTML és CSS nyelvekkel együtt használjuk weboldalak elkészítéséhez. 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 és elemzése.

## MySQL

A MySQL egy szerver-oldali adatbázis-kezelő rendszer, amely lehetővé teszi az adatok tárolását, lekérdezését és kezelését. A teljesen nyílt forráskódú LAMP (Linux-Apache-MySQL-PHP) összeállítás részeként költséghatékony, és könnyen beállítható megoldást ad a dinamikus webhelyek szolgáltatására. A MySQL már 1995-től van forgalomban. A legfrissebb verzió már több mint 1 éve, 2021-ben lett forgalomba állítva. 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. Nyílt forráskódú, és az adatok kezelésére is számos lehetőségeket biztosít.

## PHP

A PHP kelti életre a weboldalakat. A PHP segítségével tudjuk összekötni az adatbázist a weboldallal. Képes akár nagyméretű adatbázis alkalmazások működtetésére is. Hogy egy weboldal statikus HTML megjelenítésből dinamikus webes alkalmazássá váljon, a webkiszolgálónak is statikusnak kell lennie, ami a PHP dolga. A nyelvvel dinamikus képeket is létrehozhatunk, olvashatunk fájlokat, adatbázis lekérdezéseket hajthatunk végre. A lehetőségek száma végtelen.

A PHP alkalmazásnak három területe:

1. Grafikus alkalmazás
   * Ez nem a legjobb nyelv ilyen típusú alkalmazások elkészítésére, de ha ismerjük már eléggé a nyelvet, akkor lehetőségünk van ilyen módszerre is a kliens-oldali alkalmazásokban.
2. Parancssori alkalmazás
   * Akár parancsfájlokat is készíthetünk, ilyenkor csak egy PHP értelmezőre van szükségünk. Automatizálásra, vagy szöveg feldolgozására használják.
3. Szerver oldali alkalmazás
   * Ez a PHP leghatékonyabb területe. Három dologra van szükség hozzá: web szerverre, PHP értelmezőre és egy böngészőre.

# Felhasznált szoftverek

## Visual Studio Code

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

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

## XAMPP

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

## GitHub

A GitHub, Inc. egy egyesült államokbeli nemzetközi vállalat, amely a Git segítségével szoftverfejlesztési verziókövetés-szolgáltatást nyújt. 2018-ban a Microsoft leányvállalata lett 7,5 milliárd dollárért. 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 wikiket minden projekthez.

A mi projektünket is a GitHub-on keresztül követtük. Ide commitoltuk az összes változtatást, amit végrevittünk a projektmunkán, és itt tudtuk egymás munkáját nyomon követni. Az ehhez kialakított alkalmazást letöltöttük a weboldalról, így megkönnyítve a változtatások végrehajtását.

# Felhasználói dokumentáció

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

## Főoldal

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

### Navigációs menü

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

### Slideshow Carousel

A weboldalon található egy Slideshow Carousel, ami automatikusan váltogatja a képeket, valamint van az oldalán lapozó is, amivel lapozni tudunk. Itt foglal helyet középen lent a weboldalra látogatók üdvözlésére szolgáló leírás.

### Edzésterv és étrendösszeállítás leírása

Lejjebb lapozva a webalkalmazás ajánlása van a felhasználónak, aki regisztrálni szeretne. Leírjuk, hogy miért érdemes az alkalmazásra regisztrálni, és amit a vendégeinknek biztosítunk. Alatta az étrendösszeállítás leírása foglal helyet, ami azt írja le, hogy az edzésterv mellé, amit a személyedző felvesz a kliensnek, az mellé a legjobb étrendet kapja, amit szintúgy az alkalmazásban a regisztráció után kap meg a kliens egy listában.

### Számláló(counter)

A weboldal főoldalán található egy számláló is, ami az eddigi regisztrációkat, pozitív visszajelzéseket és a Facebook követőket mutatja.

### Lábléc(footer)

A weboldal alján pedig a lábléc található ami a kapcsolatokat tartalmazza, amin keresztül a felhasználó akár személyesen is felveheti a kapcsolatot az edzőkkel. Itt található a weboldal közösségi média oldalainak a címe, valamint a szerzői jog is.

## Regisztrációs oldal

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

## Bejelentkezés

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

## Kezdőlap a bejelentkezettek számára

A regisztráció és bejelentkezés után, ha edzőként regisztráltunk, akkor a kliensek nevét jeleníti meg egy listában, ahol rákattintva kiírja a kliens adatait, valamint itt tudjuk felkérni a klienst a Kliens felkérése gombra kattintva. A Csevegés gombra kattintva a klienssel tudunk privátban kommunikálni. A kliens edzéstervét itt is meg tudjuk tekinteni a Kliens edzéstervének megtekintése gombra kattintva. Ha az edző már felkérte, a felhasználó az oldalpanelen az Edzők kezelése című gombra kattintva megjeleníti az edzőt, akit felkért, és így el tudja fogadni a felkérést, valamint el is tudja utasítani azt.

## Saját profil

A kliens nevére kattintva az edző a profilját tudja megtekinteni, azaz megtudhatja a kliensek teljes nevét és az e-mail címét is, és a profilképét is megtekintheti. Emellett az edző itt tudja felkérni őt, amit majd elfogad vagy elutasít. a csevegésre kattintva megnyílik a chat felület.

## Edzéstervek

Miután elfogadta az edzőtől a felkérést, az Edzéstervek menüpontban az edző fel tudja venni a kliens edzéstervét és étrendjét. 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.

## Chat

A Chat menüpontot megnyitva az kliensek tudnak az edzőkkel kommunikálni közvetlenül az erre kialakított chates webalkalmazásban. A kliens rákattint az edzőre, aki felkérte őt, és így megnyílik egy üzenetmező egy küldés gombbal, ilyenkor pedig az utóbbi üzeneteket is betölti a rendszer, ha van ilyen.

## Kliensek kezelése(edzők számára)

Az utolsó menüpont pedig a Kliensek kezelése menüpont, ahol az edzők látják a klienseket, akiket már felkért. Ha az adott személyedző több klienst is felkért már, akkor lehetősége van arra, hogy a keresőmezőbe beírja az adott kliens nevét, és a listából egyszerűen ki tudja keresni.

## Edzők kezelése(kliensek számára)

Az edzők kezelése pedig a kliensek számára szolgál, mivel edző edzőt nem kezelhet. Ennek értelmében pedig a felület ugyan az, csak edzőkre specifikálva.

# Fejlesztői dokumentáció

## Adatbázis kapcsolat

Külön fájban (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.

A header() függvény egyik a PHP beépített függvényei közül, amely HTTP-fejléc küldésére szolgál. Ezek azok a funkciók, amelyek manipulálhatják a webszerver által küldött információkat a kliens vagy böngésző számára.

*header("Content-Type: text/html; charset=utf-8");*

Annak érdekében, hogy megfelelően térjen vissza a tartalom típusa a Content-Type fejlécet használtunk valamint ,hogy ne legyenek problémák az ékezetes betűkkel a karekterkódolást utf-8-ra állítottuk.

A define() függvényt alapvetően konstansok létrehozásához használtuk. Nagyban hasonlítanak a változókhoz, az egyetlen különbség köztük az, hogy a konstansok értékei nem változtathatók meg miután beállítottuk őket a programban.

define("DBHOST", "localhost");

define("DBUSER", "root");

define("DBPASS", "");

define("DBNAME", "edzes\_app");

Konstansoknak adtuk értékül a host name-t, a felhasználónevet, jelszót, valamint az adatbázis nevét.

$dbconn = @mysqli\_connect(DBHOST, DBUSER, DBPASS, DBNAME) or die("Hiba az adatbázis csatlakozáskor");

A mysqli\_connect() függvény képes új kapcsolatot létrehozni a MySQL szerverrel. Ezt a kapcsolatot a $dbconn nevű változóba tároltuk el.

A sikeres kapcsolat ellenőrzéséhez a mysqli\_connect\_error() függvényt használtuk. Ha volt hiba a csatlakozáskor a függvény képes visszaadni a legutóbbi hiba leírását és a die() függvénnyel kilép.

if(mysqli\_connect\_error()){

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

Ha nem volt hiba, akkor pedig kiírjuk, hogy sikeres volt a kapcsolat.

}echo "Sikeres a kapcsolat";

## Főoldal

### Navigációs menü

A navigációs menün az elemek elrendezését flexbox-al oldottuk meg. A címre rákattintva egy <a> tagben lévő href link van, ami a weboldal főoldalára mutat. itt található a hamburger ikon is, ha összehúzzuk kisebb méretűre a weboldalt, akkor megjelenik az ikon, amire rákattintva lenyílnak ugyan úgy a linkek, így reszponzívabbá téve az oldalt. Ezt pedig Javascriptben készítettük el. A jobb oldalon találhatóak a linkek a Főoldalra, bejelentkezésre és a regisztrációra. Ezt számozatlan listában helyeztem el, ugyanúgy egy <a> taggel és href linkkel, amik az adott weboldalra mutatnak.

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>

### Slideshow Carousel

A főoldal carouselje 4 divből áll, amiben helyet foglal a kép, és a szövegek. A kép divje a carousel-img osztálynévre hallgat, míg a szöveg osztálya a text osztálynévre. A köztük lévő slide fade osztályú div elválasztja a carouselben megtalálható képeket, egy áttünéses animációval. A carousel alján gombok vannak, amik a dots osztálynévre hallgatnak, benne pedig span taggel a 3 gombra onclick eseménnyel rálötött function, amit a Javascriptben írtunk meg. Egy <a> tagben megírtuk a gombokat, ami a képernyő két szélén található, hogy kattinthatók legyenek egy onclick eseménnyel, ami ha hozzáad egyet, akkor a következő kép ugrik be, ha pedig elvesz egyet, akkor az előzőt mutatja. A Carousel Javascriptjében pedig létrehoztunk egy változót SlideIndex néven, aminek a kezdőértéke 0, és ehhez fog hozzáadni, vagy elvenni a function, amit erre írtunk meg. A ShowSlides functionön belül azt írtuk meg, hogy a gombok kattintására megjelenjenek a képek, paraméterként pedig n változót adtuk meg. A slides változót kiválogatjuk egy document.getElementsByClassname-el, amiben megadtuk, hogy a slide osztályt válassza ki. 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 képeket, hogy ne mutassa kattintás hatására, style.display = ”none”-al írtuk meg. A második for ciklus pedig arra szolgál, hogy a 3 gomb a kép alján kattintható legyen, így a gomb elérhetővé válik, egyébként pedig nem válik elérhetővé. A for cikluson kívül pedig a slideok stílusát blokk eleműként mutatja, azaz foglalja el az oldal maximum szélességét. A gombokra pedig hozzáadjuk az active osztályt, hogy elérhetők legyenek.

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 megoldottuk, hogy x másodpercenként lapozza a képeket. a for ciklus és az if szerkezet között láthatunk egy SlideIndex++-t, ami azt csinálja, hogy hozzáad egyet a képekhez, tehát lapozza azt. És ezt automatizáltuk a setTimeout metódussal, aminek paraméterként megadtuk az AutoSlide functiont, és hogy 6,5 másodpercenkét váltogassa a képeket.

function AutoSlide() {

let i;

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

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

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

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

}

slideIndex++;

if (slideIndex > slides.length) {

slideIndex = 1;

}

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

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

}

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

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

setTimeout(AutoSlide, 6500);

}

### Edzésterv és étrendösszeállítás leírása

A főoldalon található 2 wrapper és egy section. Az első és második wrapper az oldalon lévő leírásért felelős. Ezeket, hogy a képernyő egyik szélétől a másik széléig legyen, 100%-os szélességgel oldottam meg. A margókat és a paddinget 50pixelesre állítottam, és, hogy auto legyen. Margin-topnak pedig 15pixelt raktam, hogy elváljon a container tetejétől a h1-es címsor. Elhelyeztünk egy border osztálynévre hallgató divet is, ami a container alján található zöld sík. Ez lényegében egy csak egy design. CSS-ben a wrapperekre adtunk egy fehér betűszínt és fekete hátteret.

A görgetősáv stílusát is megváltoztattuk a weboldal alap színére, ami főleg a zöld és fekete.

body::-webkit-scrollbar{

width: 10px;

background-color: var(--fekete);

}

body::-webkit-scrollbar-track{

border-radius: 10px;

}

body::-webkit-scrollbar-thumb{

background-color: var(--zold);

border-radius: 10px;

}

body::-webkit-scrollbar-thumb:hover{

background-color: #3d8a2a;

}

HTML:

<div class="wrapper1">

<div class="container1">

<h1>Edzéstervek</h1>

<p>

Edzéstervek a személyedzők ajánlásával. <b><a href="reg.html" class="regLink">REGISZTRÁLJON MOST</a></b>, hogy elkezdje minél előbb az edzést a kitűzött cél eléréséhez.

</p>

<p>A webalkalmazásban a kliensek a legjobb edzéstervet kapják a személyedzőktől.</p>

<h2>Amit kínálunk:</h2>

<ul>

<li>Megbízhatóság</li>

<li>Baráti társaság</li>

<li>Hatékony edzésterv</li>

</ul>

</div>

<div class="border"></div>

</div>

<div class="wrapper2">

<div class="container2">

<h1>Étrendösszeállítás</h1>

<p>

A legjobb étrend az edzésterv mellé, amit a személyedző az edzésterv mellé ajánl.

</p>

</div>

<div class="border"></div>

</div>

CSS:

.wrapper1 {

margin: 50px auto;

margin-top: 15px;

background-color: var(--fekete);

color: var(--feher);

width: 100%;

padding: 50px;

}

.wrapper2{

margin: 50px auto;

margin-top: 15px;

background-color: var(--fekete);

color: var(--feher);

width: 100%;

padding: 50px;

}

.container2{

text-align: center;

}

.container1{

text-align: center;

}

.container1 ul{

list-style-type: none;

color: var(--zold);

font-size: 25px;

}

.container1 h2{

margin-top: 50px;

}

.wrapper1 .container1 .regLink{

color: var(--zold);

text-decoration: none;

}

.wrapper1 .container1 .regLink:hover{

opacity: 0.6;

}

Egy link is látható az első containerben, ami a regisztrációra mutat. A CSS-ben megadtuk neki, hogy ne legyen rajta semmilyen alap formázás, és ráraktunk egy zöld betűszínt. A kurzort, ha rámutatjuk a linkre, akkor elsötétedik a betűszíne. Ezt egy 0.6-os opacityvel oldottam meg. A containerek elemei, hogy középre legyenek helyezve, text-align: centerrel lett megvalósítva. A container1-en lévő listánál, hogy ne jelenjenek meg a szimbólumok előtte, egy list-style-type: none-t adtam meg. A betűszíne zöld, és 25pixeles betűméret.

### Számláló(counter)

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

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

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

let speed = 100;

counters.forEach(counter => {

let updateCount = () => {

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

let count =+ counter.innerText;

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

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

counter.innerText = count + inc;

setTimeout(updateCount, 8);

}else{

count.innerText = target;

}

}

updateCount();

});

### Lábléc(footer)

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

<footer>

<div class="footer-container">

<h2>Kapcsolat</h2>

<div class="listak">

<ul class="lista">

<li class="fa fa-envelope"> E-mail: shinegymandfit@gmail.com</li>

<li class="fa fa-phone"> Tel.: 0620000000</li>

</ul>

<ul class="lista">

<li class="fa fa-facebook"> Facebook: ShineGym&Fit Official Page</li>

<li class="fa fa-instagram"> Instagram: ShineGym&Fit</li>

</ul>

<ul class="lista">

<li class="fa fa-youtube-play"> YouTube: ShineGym&Fit</li>

<li class="fa fa-twitter"> Twitter: ShineGym&Fit</li>

</ul>

</div>

<p><em class="fa fa-copyright"> Copyright ShineGym&Fit 2023 - All Rights Reserved</em></p>

<hr class="border-footer">

</div>

</footer>

## Chat

A chat.php chat felületének HTML részében először is külön PHP fájlokban írtuk meg az oldalsó navigációs menüt és a felső navigációs menüt, és itt a HTML-en belül hivatkoztunk a fájl nevére require-el, és kiírattuk.

<?php require("leker/SidebarNavbar.php"); ?>

És innentől már a chat felület kinézete található. megy main tagen belül, aminek az osztálya a ChatMain, azon belül 2 div, amelyek a ChatWrapper, left\_side\_pannel osztályneveket kapták. A left\_side\_pannel, azaz a felhasználólista tetején, a keresőmező felett található a weboldal címe, ami egy <a> tagben foglal helyet, mcim osztálynévvel. Továbbá van egy befoglaló div contact osztálynévvel, ezen belül maga a lista, és a keresőmező foglal helyet. A lista tartalmát a PHP segítségével a PHP print funkcióval kiírattuk. A keresőmezőt post metódus segítségével készítettük el, mivel ezzel a metódussal a bodyban utazik az adat, és nem az URL-ben szeretnénk megjeleníteni az adatokat, hanem magán a panelen. Van egy keresés típusú input mezőnk is kifejezesChat névvel, ugyanerre a névre szóló azonosítóval. Mellette pedig egy button, kereses azonosítóval, benne egy keresés ikonnal.

<main class="chatMain">

<div class="chatWrapper">

<div class="left\_side\_pannel">

<a class="mcim">ShineGym&Fit</a>

<div class="contact">

<form method="post">

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

<button id="kereses"><i class="fa fa-search"></i></button>

</form>

<?php print $chatLista;?>

</div>

</div>

Ezek után található a felületen a jobb oldali része a chatnek, azaz a right\_side\_pannel osztálynévre hallgató div. Ezen belül a header és a prof divek osztályneve. A prof osztályon van egy onclick esemény is, ami arra a felhasználói profilra vezet át, akivel mi jelen pillanatban chatelünk. Ezt a chat jobb oldali fejlécében helyeztük el, és szintén php print segítségével írattuk ki.

<div class="header">

<div class="prof" onclick="location.href='sProfil.php';">

<?php

isset($\_GET['chat']) ? print "<div class=\"pkep\"><img src=\"../pics/profile/{$Vkep}\"></div><p>{$Vvnev} {$Vknev}</p>" : '';

?>

</div>

</div>

A másik kliens profil kiíratását egy conditional operátorral oldottuk meg. Ezzel az operátorral 3 értéket vár a program. Az első az a feltételezés. Ez esetben az isset azt a feladatot hajtja végre, hogy a változó deklarálva legyen. A $\_GET metódussal pedig az URL-ben utazik az adat, és így oldjuk meg azt, hogy az adott személlyel tudjunk chatelni az ID-je szerint. Ha az ID-je be lett olvasva, akkor az adott kliens profilképét, vezetéknevét és keresztnevét írja ki. Ezt elválasztjuk egy kettősponttal, és a kettőspont után az else ág foglal helyet. Jelenleg ennek a kikötésnek else ága nincs, ezért ezt nem kell megadnunk, beírunk egy üres stringet. Ez után következik a chat felület containere, ezen belül pedig maga az üzenetek divje chatUzenetek osztálynévvel, ami egy üres div, mivel ide íratjuk ki majd PHP-ből az üzenet menetét. Ezek alatt ugyanúgy egy form tagben post metódussal megadva egy textarea típusú szövegmező, ahová a chat szövegét tudjuk begépelni. Erre a textarea-ra a szoveg nevet, és ugyanezt az azonosítónevet írtuk meg. Át is formáztuk inline style-al. Ráraktunk egy betűtípust, és egy 13 pixeles padding-topot, hogy ne ragadjon hozzá a textarea tetejéhez a szöveg, amit placeholderként írtunk meg. Mellette pedig egy submit típusú buttont helyeztünk el, aminek a neve és id-je szintén ugyan az, chatUzenet névvel. A buttonön belül pedig található az ikon.

<div class="container">

<div class="chatUzenetek">

<!-- Üzenetek kiírása -->

</div>

<form method="post" class="chat-szoveg-kuldes">

<textarea type="text" name="szoveg" id="szoveg" placeholder="Ide írja a szöveget..." style="font-family: 'Nunito', sans-serif; padding-top: 13px;"></textarea>

<button type="submit" name="ChatUzenet" id="ChatUzenet" title="Küldés"><i class="fa fa-paper-plane" aria-hidden="true"></i></button>

</form>

</div>

</div>

</div>

</main>

A PHP-t úgy kezdtük el, hogy a kapcsolat meg legyen az adatbázissal. Ha a felhasználó nincs bejelentkezve, akkor egy header functionnel megadtuk, hogy vezesse őt vissza a belepes.php-re, egyébként ha be van jelentkezve, akkor hozza létre a kapcsolatot és kérje le a saját profilunk adatait.

session\_start();

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

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

exit();

} else {

require("kapcsolat.php");

//Saját profil adatainak lekérése

require("leker/sajatProfil.php");

Ezután jön az oldalsó lista a kliensekről. Ha van keresett kifejezés, ezesetben a név, akkor a keresett kifejezésre hasonlító találatokat jelenítse meg a teljes lista helyett. Ezt conditional operátor segítségével hoztuk létre.

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

Következő a lekérdezés. A $felhChat változóban adjuk meg, hogy milyen típusú profilokat akarunk lekérdezni.

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

$felhChat = "kliens";

}else{

$felhChat = "edző";

}

Ez azt takarja, hogy ha a profil típusa, amivel bejelentkeztünk, ha az edző, akkor a kliensek listáját írja ki, de ha pedig a profil típusa kliens, akkor az edzőket listázza ki.

Ez után jön MySQL segítségével az adatok lekérése az adatbázisból. Az adatokat eltároltuk az $fosszesChat nevű változóban. Mysqli\_query segítségével az adatokat lekérdeztük. Először is az adatbázis kapcsolatát, ami a $dbconn nevű változóra hallgat, meghívtuk, aztán következik a lekérdezés. Minden adatot kiszelektáltunk a felhasznalok táblából, ahol a profil típusa megegyezik a klienssel, vagy az edzővel, amiket eltároltunk a $felhChat nevű változóban. A legvégén pedig összekonkatenáltuk a vezetéknevet a keresztnévvel a $kifejezesChat változó adatai alapján.

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

Ha ez meg van, akkor összeállítjuk az összes edző típusú felhasználó listáját a $chatLista változóba.

Ezután egy while ciklussal létrehoztunk egy $felhasznalo tömböt, ahol a mysqli\_fetch\_assoc segítségével az adatokat kiválogattuk ebbe az asszociatív tömbbe, amit az $fosszesChat nevű változóban már megadtunk. Ha nincs több sor a tömbben, akkor false értékkel tér vissza.

A cikluson belül létrehoztunk egy $felhEllenorzes nevű változót, ahol ismét egy MySQL lekérdezést írtunk meg. Mindent kiszelektáltunk az ekkapcs táblából, ahol a fogadónak az azonosítója megegyezik a felhasználó azonosítójával. A küldő azonosítója pedig megegyezik a tömbben eltárolt felhasználó azonosítójának számával. Vagy fordítva, ha a küldő azonosítója egyezik meg a felhasználó azonosítójával, és a fogadó azonosítója a tömbben eltárolt felhasználó azonosítójával. Miután ezt megírtuk, létrehoztuk a kapcsolatot a mysqli\_query segítségével, amit eltároltunk egy $felhEllEredmeny nevű változóban. Ezt egy asszociatív tömbben szintén eltároltuk, amelynek a neve a $felhEllSor. A $chatLista változóhoz hozzá konkatenáltuk azt , amit majd meg kell jelenítenie a jobb oldali listában a weboldalon, tehát a felhasználó profilképét, vezetéknevét és keresztnevét. A while cikluson kívül pedig lezártuk a konkatenálást egy üres stringgel.

$chatLista = "";

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

$felhEllenorzes = "SELECT \* FROM edzoklienskapcs

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

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

$felhEllEredmeny = mysqli\_query($dbconn, $felhEllenorzes);

$felhEllSor = mysqli\_fetch\_assoc($felhEllEredmeny);

$chatLista .= "<a href=\"?chat={$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>";

}

$chatLista .= "";

A következő egy if szerkezet. Ha az URL-ben lévő chatprofil id-je megegyezik a választott felhasználó id-jével, akkor a mysqli\_query-ben megadott lekérdezés alapján, tehát a profilképét, vezetéknevét és keresztnevét írja ki a fejlécbe. A lekérdezést eltároltuk egy $sqlValP nevű változóban, ami a választott profil adataira utal. a mysqli\_fetch\_assoc-al pedig eltároltuk egy tömbbe az adatokat. A tömb neve pedig $ValP. Külön változókba eltároltuk a vezetéknevét, keresztnevét, és profilképét.

Megadtuk azt is, hogy a chat azonosító legyen egyenlő a $\_GET metódussal lekért URL-ben lévő chat azonosítóval. Ezen felül a $\_GET metódusban felvitt adatot eltároltuk egy $fogadoAz nevű változóba. Ha a $\_POST metódussal lekért chatUzenet és a szoveg nem egyenlő üres string,, akkor írja ki az adatokat. A nem egyenlő üres string arra a célra szolgál, hogy ha a felhasználó nem írt semmilyen szöveget a szövegmezőbe, akkor ne lehessen felvinni az adatbázisba adatot, tehát nem fogja elküldeni a chatet. Ha pedig van szöveg írva, akkor az adatbázisba vigye fel az alábbi adatokat:

* Az üzenet elküldésének dátumát a $mikor változóban eltárolva.
* Magát az üzenet szövegét $\_POST metódussal a szoveg azonosítójú textarea-ból.
* Beszúrás az adatbázisba MySQL-el. A változó neve: $sqlBeszur.

Az SQL lekérdezés így történt: Először is a kapcsolatot létrehoztuk a $dbconn változó segítségével, aztán beszúrtuk az uzenet táblába INSERT INTO-val a felhasználó id-jét(felh\_id), a $fogadoAz változót, kimeno\_id-t, bejovo\_id-t, az üzenet dátumát(mikor), és magát az üzenetet(uzenet). Aztán pedig az adatok értékeit. És végül pedig elkészült a csevegő alkalmazásunk HTML és PHP része.

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

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

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

$Vvnev = $ValP['vnev'];

$Vknev = $ValP['knev'];

$Vkep = $ValP['kep'];

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

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

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

$mikor = date("Y-m-d H:i:s");

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

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

}

}

}

### Chat üzenetek kiírása

Először is elindítjuk a munkamenetet a session\_start() funkcióval.

session\_start();

Létrehoztuk a kapcsolatot az adatbázissal require-el. Alatta egy if-es szerkezetben megadtuk egy $\_SESSION globális változóban a chataz nevet. Majd ezt egy változóban, a $fogadoAz nevű változóban tároltuk el. Ez után az $sqlKim változóban az adatbázis lekérdezést hoztuk létre. A lekérdezésben az uzenetek táblából kiválasztjuk a kimenő üzenet id-jét, a bejövő üzenet id-jét és magát az üzenetet, ahol a kimeno\_id és a felh\_id megegyezik és a bejovo\_id megegyezik a $fogadoAz nevű változóval, amit már feljebb megadtunk. Vagy fordítva, ha a bejovo\_id egyezik meg a felh\_id-vel, azaz a felhasználóval, és a kimeno\_id egyezik meg a $fogadoAz nevű változóval. Létrehoztunk egy $kiiras nevű változót, ami segítségével majd konkatenáljuk az üzenet szövegét.

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

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}");

$kiiras = "";

Majd egy while ciklusban, ahol $sorKim változóval megadtunk egy asszociatív tömböt a mysqli\_fetch\_assoc() segítségével, paraméterként megadtuk neki a $sqlKim változót, amelyikben lekérdeztük az adatokat az adatbázisból. Ha a $sorKim változó egyenlő a felh\_id-vel, akkor írassa ki a kimenő üzenetet, tehát amit mi írunk a szövegmezőbe és elküldünk. Ez fog megjelenni fekete hátérrel az alkalmazás jobb oldalán. Egyébként pedig bejövő üzenet legyen. Ez a bal oldalon jelenik meg, fehér háttérrel. Ezt írta a szövegmezőbe a másik fél, akivel írunk. A legvégén pedig print-el kiprinteljük a $kiiras változót, hogy megjelenjenek az üzenetek.

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

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

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

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

</div>";

}

else{

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

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

</div>";

}

}

print $kiiras;

}

### Chat felület JavaScript része

A Javascriptet is felhasználtuk a chat alkalmazásunk elkészítéséhez. Először is létrehoztunk 4 konstanst form, inputField, sendBtn és chatBox névvel. a form konstanson belül querySelectorral kiszelektáltuk a chat-szoveg-kuldes osztályt, inputField-en belül a szoveg id-t, a sendBtn-en belül a ChatUzenet id-t, a chatBox-on belül pedig a chatUzenetek osztályt.

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 úgy oldottuk meg, hogy ha az értéke nem egyenlő üres stringgel, tehát ha nem üres, akkor a küldés gomb legyen elérhető. Egyébként pedig ha üres a mező, akkor ne legyen elérhető.

inputField.focus();

inputField.onkeyup = ()=>{

if (inputField.value != "") {

sendBtn.disabled = false;

}else{

sendBtn.disabled = true;

}

}

Ezek után megírtuk azt, hogy ha a chatBox változóban megadott mezőre, 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 majd eltároljuk az URL-ben lévő adatokat. Ha nincs megnyitva egy chat ablak, akkor pedig a küldés gomb és a szövegmező ne legyen látható. Ha egy chat ablakot megnyitunk egy felhasználóval, akkor megjelenik a küldés gomb és a szövegmező is. If szerkezettel oldottuk meg azt, hogy ha az URL sáv tartalmazza a chat= kifejezést, akkor jelenítse meg a küldés gombot és a szövegmezőt. És ha az URL tartalmazza a chat= kifejezést, akkor az AJAX-ot is hozza létre.

let url = window.location.href;

sendBtn.style.display = "none";

inputField.style.display = "none";

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, ha a chatoldalon létrejött. Tehát, ha új üzenetet akarunk írni egy adott felhasználónak, akkor ne kelljen folyton frissíteni az oldalra a felhasználónak, hogy elküldje az üzenetet, hanem a rendszer automatikusan küldje el. Ezt egy setInterval-ban adtuk meg. Ezen belül létrehoztunk egy funkciót, amiben lefut maga az AJAX. Megírtunk egy változót xhr néven, ami hatására létrehozzuk az XMLHttpRequest-et. Ez az a megoldás, hogy az oldalt ne kelljen folyton frissíteni a felhasználónak. Aztán xhr.open-el lekérjük a szerverről az adatot GET metódussal, a chatleker.php fájlból. Ha pedig betöltötte a lekérést, és ha jól lefutott, tehát a 200-asra fut a program, akkor létrehoztunk egy változót data néven, és ezt az data-t írjuk ki a HTML-ben. Megadtuk azt is, ha a chatBox nem tartalmazza az active osztályt, akkor görgessen le a chat aljára. Tehát, ha új üzenetet küldünk és az első üzenetek között vagyunk, akkor ne kelljen le görgetni folyamatosan, hanem automatikusan leugrik a jelenlegi üzenetekhez. Ezt pedig xhr.send-el elküldjük 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();

}, 500)

}

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

function scrollToBottom(){

chatBox.scrollTop = chatBox.scrollHeight;

}

Arra is figyeltünk, hogy ha az enter-t lenyomjuk, küldje el az üzenetet. Az inputField-re kötöttünk egy addEventListener-el egy funkciót, hogy gombnyomásra, ha a billentyű, amit lenyomtunk, az enter, akkor az event.preventDefault-al akadályozza meg a weboldal alap működését és a küldés gomb reagáljon úgy az enter lenyomására, mintha egérrel kattintottunk volna rá, tehát az enter lenyomására küldi el az üzenetet.

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

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

event.preventDefault();

sendBtn.click();

}

})

## Kezdőlap

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

session\_start();

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

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

exit();

} else {

require("kapcsolat.php");

//Saját profil adatainak lekérése

require("leker/sajatProfil.php");

//felhasználók listája

require('leker/felhLista.php');

}

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

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

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

### Keresőmező

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

<div class="felh-lista scrollbar">

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

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

<form method="post">

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

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

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

<?php

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

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

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

?>

</form>

### Lista

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

<?php echo $kimenet ?>

</div>

### Extra funkció kliens típusú felhasználóknak

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

<?php

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

require("leker/tevekenysegek.php");

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

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

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

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

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

Felület összeállítása $tevTeljes változóba

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

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

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

$sTev['leiras']

."</div>

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

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

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

<div class=\"tevTeljGombok\">

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

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

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

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

</div>

</div>";

Miután mindent összefűztünk a $tevTeljes változóba kiírjuk azt.

print $tevTeljes;

}

}

?>

### 4.4.4 tevTeljTorles() függvény

function tevTejlTorles(datum, tevazon){

Swal.fire({

title: 'Biztosan törölni szeretné?',

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

icon: 'warning',

showCancelButton: true,

confirmButtonColor: '#55be3b',

cancelButtonColor: '#080B0C',

confirmButtonText: 'Törlés',

cancelButtonText: 'Mégsem'

}).then((result) => {

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

if (result.isConfirmed) {

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

}

})

}

## sajatProfil.php – Saját profil adatai

### Lekérdezés

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

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

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

FROM felhasznalok

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

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

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

$vnev = $sor['vnev'];

$knev = $sor['knev'];

$email = $sor['email'];

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

$kep = $sor['kep'];

$nem = $sor['nem'];

$bemutatkozo = $sor['bemutatkozo'];

$telefon = $sor['telefon'];

### Kimenet és profilkép összeállítása

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

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

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

$kimenet = "

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

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

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

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

</div>

Profilkép és táblázatos rész:

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

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

<p>Profilkép</p>

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

</div>

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

<table>

<tr>

<th>Vezetéknév:</th>

<td>{$vnev}</td>

</tr>

<tr>

<th>Keresztnév:</th>

<td>{$knev}</td>

</tr>

<tr>

<th>E-mail:</th>

<td>{$email}</td>

</tr>

<tr>

<th>Profil típusa:</th>

<td>{$profilTipus}</td>

</tr>

<tr>

<th>Nem:</th>

<td>{$nem}</td>

</tr>";

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

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

if($profilTipus == "kliens"){

$kimenet .= "<tr>

<th>Telefon:</th>";

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

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

$kimenet .= "</tr>

</table>

</div>

<div class=\"bemutatkozo\">

<h2>Bemutatkozó:</h2>";

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

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

$kimenet .= "</div>";

}

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

else{

$kimenet .= "<tr>

<th>Telefon:</th>

<td>{$telefon}</td>

</tr></table>

</div>

<div class=\"bemutatkozo\">

<h2>Bemutatkozó:</h2>

<p>{$bemutatkozo}</p>

</div>";

}

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

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

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

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

if($\_SESSION['p\_tipus'] == "edző"){

$felhTipus = "kliens";

}

else{

$felhTipus = "edző";

}

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

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

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

$kimenet = "";

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

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

$fEll = "SELECT elfogadva FROM edzoklienskapcs

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

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

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

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

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

$fvan = 0;

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

$fvan = 1;

$felfogadva = $fEllSor['elfogadva'];

}

Elkezdjük a kimenet változóban összeállítani a listát.

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

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

<div class=\"felh\">

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

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

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

if($fvan == 1){

if($felfogadva == 1){

if($\_SESSION['p\_tipus'] == "edző"){

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

}

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

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

}

}

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

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

else{

$kimenet .= "(Felkérve)";

}

}

$kimenet .= "</p>

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

}

## tevekenysegek.php - Tevékenységek lekérése

### Lapozó

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

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

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

} else {

$lap = 1;

}

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

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

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

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

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

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

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

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

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

### Shorter függvény

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

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

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

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

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

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

}

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

else{

return $text;

}

}

### Tevékenységek lekérdezése és kimenet összeállítása

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

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

FROM tevekenysegek

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

ORDER BY datum DESC

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

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

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

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

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

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

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

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

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

$tDatum = $sorTev['datum'];

$tLeiras = $sorTev['leiras'];

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

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

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

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

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

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

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

</div>";

}

}

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

else{

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

}

$tevKimenet .= "</div>";

## tevModositas.php – Tevékenység módosítása/szerkesztése

A modID változóban eltároljuk a webcímből kiolvasott azonosítót.

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

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

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

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

$kDatum = $sor['datum'];

$kLeiras = $sor['leiras'];

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

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

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

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

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

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

}

## tevTorlese.php – Tevékenység törlése

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

A $torlendo változóban eltároljuk a webcímből kiolvasott tevékenység azonosítóját.

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

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

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

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

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

## tevRogzitese.php – Tevékenység rögzítése

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

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

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

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

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

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

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

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

header("Location: kezdolap.php");

}

Megjelenítés a főoldalon (kezdolap.php):

<div class="sikeres">

<?php

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

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

print $\_SESSION['tevrogz'];

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

}

?>

</div>

# Összefoglalás

Tovább fejlesztési lehetőségek: kalóriaszámláló,

Tervezzük az oldal keretrendszerbe való áthelyezését, CakePHP –be vagy Laravelbe, mivel ott könnyedén megvalósíthatónak tűnik a többnyelvűsítés is. Ugyanis felvetettük azt az ötletet is, mivel Sebastian szlovák, így mindenképpen érdemes kihasználnunk a szlovák nyelv tudását is. Mivel mindketten tanultunk angolul, így az lenne a harmadik választható nyelv majd az oldalunkon. Tervezünk hozzá mobil applikációt is írni, de ehhez még folytatnunk kell a tanulást.

fizetési rendszer pl. paypalon keresztül, ahol úgy oldanánk meg, hogy amikor a vendég fizet az edzőnek, akkor abból 5% jutalékot számolnánk fel, amelyből az oldal működtetésének költségeit fedezni tudjuk, mint szerver bérlet, és a saját munkadíjunk az oldal karbantartására.

## Források