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

Tartalom

[Bevezető 3](#_Toc129588542)

[1. Felhasznált technológiák 5](#_Toc129588543)

[1.1. HTML 5](#_Toc129588544)

[1.1. CSS 5](#_Toc129588545)

[1.2. JavaScript 5](#_Toc129588546)

[1.3. MySQL 6](#_Toc129588547)

[1.4. PHP 6](#_Toc129588548)

[2. Felhasznált szoftverek 7](#_Toc129588549)

[2.1. Visual Studio Code 7](#_Toc129588550)

[2.2. XAMPP 7](#_Toc129588551)

[3. Felhasználói dokumentáció 8](#_Toc129588552)

[3.1. Főoldal 8](#_Toc129588553)

[3.1.1. Navigációs menü 8](#_Toc129588554)

[3.1.2. Slideshow Carousel 8](#_Toc129588555)

[3.1.3. Edzésterv és étrendösszeállítás leírása 8](#_Toc129588556)

[3.1.4. Számláló(counter) 8](#_Toc129588557)

[3.1.5. Lábléc(footer) 8](#_Toc129588558)

[3.2. Regisztrációs oldal 9](#_Toc129588559)

[3.3. Bejelentkezés 9](#_Toc129588560)

[3.4. Kezdőlap a bejelentkezettek számára 9](#_Toc129588561)

[3.5. Saját profil 9](#_Toc129588562)

[3.6. Edzéstervek 9](#_Toc129588563)

[3.7. Chat 10](#_Toc129588564)

[3.8. Kliensek kezelése(edzők számára) 10](#_Toc129588565)

[3.9. Edzők kezelése(kliensek számára) 10](#_Toc129588566)

[4. Fejlesztői dokumentáció 10](#_Toc129588567)

[4.1. Adatbázis kapcsolat 10](#_Toc129588568)

[Összefoglalás 12](#_Toc129588569)

# 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.

# 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 letöltéseket, 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 a kedve szerint megadhatja, ha akarja, vagy nem. A többi mező megadása kötelező. Például a Vezetéknév, Keresztnév, E-mail, Jelszó, amit meg is kell erősíteni, Képzettség, Tapasztalat és a Telefonszám. 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 kliensnek 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, a kliens edzéstervének megtekintése pedig az erre létrehozott gomb szolgál.

## Edzéstervek

Miután elfogadta az edzőtől a felkérést, az Edzéstervek menüpontban az edző fel tudja venni a kliens edzéstervét és étrendjét. 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/edző 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) hoztam 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 használtam valamint ,hogy ne legyenek problémák az ékezetes betűkkel a karekterkódolást utf-8-ra állítottuk.

A define() függvényt alapvetően konstansok létrehozásához használjuk. 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 adtam 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 tároltam el.

A sikeres kapcsolat ellenőrzéséhez a mysqli\_connect\_error() függvényt használtam. 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);

}

### Főoldal tartalma

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 width-el 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.

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 oldottuk 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 *counters* osztályhívónévvel ellátott div tagben egy 0 van kezdőértékként, hogy a számláló majd a nullától induljon. A data-target pedig azt takarja, hogy a számlálás megfelelően fusson le.

### Lábléc(footer)

# 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 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.