Szegedi Szakképzési Centrum Vasvári Pál Gazdasági és Informatikai Szakgimnáziuma

Az 54 213 05 számú Szoftverfejlesztő szakképesítés záródolgozata



Készítette:

Rózsa István

Szeged

2020

Tartalom

[**Bevezetés** 3](#_Toc36318347)

[**Fejlesztői Dokumentáció** 4](#_Toc36318348)

[**Adatbázis** 4](#_Toc36318349)

[**Adatbázis ábra** 4](#_Toc36318350)

[**Egyedek és tulajdonságaik** 4](#_Toc36318351)

[**Asztali alkalmazás** 6](#_Toc36318352)

[**Felhasználok** 7](#_Toc36318353)

[**Az alkalmazás** 8](#_Toc36318354)

[**Tesztelés** 11](#_Toc36318355)

[**Web alkalmazás** 12](#_Toc36318356)

[**Index** 12](#_Toc36318357)

[**Csatlakozás az adatbázishoz** 13](#_Toc36318358)

[**Bejelentkezés és regisztráció** 13](#_Toc36318359)

[**Profil** 14](#_Toc36318360)

[**Fejlesztési tervek** 14](#_Toc36318361)

# **Bevezetés**

Egyre nagyobb szerepet játszik a sport, az egészséges életmód az emberek életében.   
Ezért a záródolgozatomba egy olyan alkalmazást hoztam létre, amely a testépítő életmódot támogatja. A mai világban már majdnem minden dolgát az ember online, interneten keresztül intézi. Azonban ez nem igaz az élet minden területére.

Sportolói pályafutásom során számtalan problémát fedeztem fel. Sokan úgy tartják, hogy a testépítés az nem egy sport. Számtalanszor látok most kezdő hétköznapi embereket az edzőteremben, akik füzettel és tollal a kezükben vágnak bele ebbe a sportba, hogy majd edzéstervet és étrendet írnak maguknak. De sokan nem tudják mikor, mennyit és mit egyenek vagy, hogy milyen gyakorlatokat csináljanak a fejlődés érdekében. Ezek a papírok időközben akár el is tűnhetnek, elkeveredhetnek vagy elmaszatolódhat a rajtuk lévő tartalom ebből adódóan már nem is biztos, hogy olvasható így felesleges időtöltés volt ezeknek a megírása. Nem ez a legmegbízhatóbb formája egy nyilvántartásnak.

Úgy gondolom, hogy a mindennapokat meg könnyítené egy olyan alkalmazás, amely útmutatást, irányt adna akár kezdő, újrakezdő vagy már régóta edző ember számára. Mivel már én is közel 5 éve sportolok egy-két kisebb kihagyással, így tudom, hogy egy ilyen szoftverrel könnyebbek lennének a napok. Egy olyan alkalmazással, amely megvalósítja azokat a funkciókat aminek a segítségével minden állandósított adat visszanézhető legyen és nagyobb biztonságot nyújtson mint a papír és társai. Pontosan ezért választottam ezt a témát, hogy minden téren modern legyen ez a sport.

Ezért úgy döntöttem, hogy megvalósítom ezt az alkalmazást, amely képes arra, hogy felvegye a versenyt a korszerűtlen eszközökkel. A **BuildYourBody** egy olyan korszerű és biztonságos alkalmazás, amellyel minden ember napja egyszerűbbé válik. A rendszeren belül megtalálható számos funkcionális követelmény, mint például minden regisztrált felhasználó BMI-t[[1]](#footnote-1)1 tud majd számolni, meg tudja saját magának határozni a testtömege alapján, hogy egy nap hány gramm fehérjére, szénhidrátra, zsírra van szüksége a fejlődés érdekében. Képes lesz naplózni az étkezéseit, edzéseit és a használt táplálékkiegészítőket. A szoftver megvalósításával minden kliens egy felhasználóbarát, letisztult rendszerben tud majd dolgozni.

# **Fejlesztői Dokumentáció**

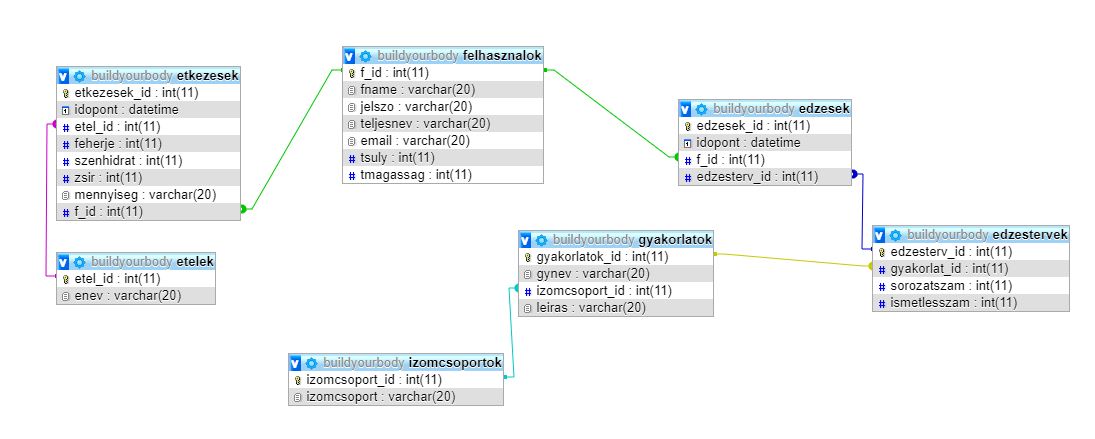
## **Adatbázis**

A rendszer működéséhez szükség van egy olyan adatbázisra amely tárolja mindazon adatokat amelyek előfordulnak egy edzés és életmód során. Az adatbázis a szoftver mindkét részét az asztali és a webalkalmazást is egyaránt kiszolgálja.

Az adatbázis célja, hogy személyre szabottan minden felhasználónak biztosítson bejelentkezést asztali és webes felületen is. **(EZT ITT ÁT KELL ÍRNI)**Valamint biztosítson webes felületen regisztrációt, étrendek, étkezések, edzéstervek és edzések összeállítását esetleges módosítását. Asztali felületen garantálja az étkezések és edzések megtekintését.

### **Adatbázis ábra**

Ez az ábra szemlélteti az egyedek közötti kapcsolatot, a táblák tartalmát és az adattípusokat.



1. ábra Bachmann-ábra

### **Egyedek és tulajdonságaik**

A **felhasználók** táblába kerül eltárolásra a bejelentkezéshez szükséges felhasználónév és jelszó, a felhasználó teljes neve, e-mail címe valamint testsúlya és testmagassága.

Az **edzések** táblába található személyre szabottan az edzések időpontja és maga az edzésterv.

Az **edzéstervek** táblába megtalálható, hogy egy edzéstervben egy gyakorlatból hány sorozatot és hány ismétlést végzünk.

A **gyakorlatok** táblában maga a gyakorlat neve, milyen izomcsoporthoz tartozik és egy rövid leírás található.

Az **izomcsoportok** táblában fellelhető minden izomcsoport.

Az **étkezések** táblában látható egy étkezés időpontja, maga az étel és a mennyisége ami elfogyasztásra kerül és hogy mennyi fehérjét, szénhidrátot és zsírt vittünk be a szervezetbe az étkezéssel.

Az **étel** táblában található minden olyan étel amelyek az étkezések során fogyasztásra kerülnek.

A **felhasználók** tábla tulajdonságai:

* f\_id: Egész szám, elsődleges kulcs, automatikusan növekvő érték amely a felhasználók megkülönböztetésére szolgál.
* fname: Szöveg, a bejelentkezéshez szükséges felhasználónév, egyedi.
* jelszo: Szöveg, a bejelentkezéshez szükséges jelszó.
* teljesnev: Szöveg, a felhasználó neve.
* email: Szöveg, a felhasználó e-mail címe.
* tsuly: Egész szám, a felhasználó testsúlya.
* tmagassag: Egész szám, a felhasználó testmagassága.

Az **edzések** tábla tulajdonságai:

* edzesek\_id: Egész szám, elsődleges kulcs, automatikusan növekvő érték, az edzés azonosítója.
* idopont: Date típusú, az edzés időpontját jelöli, automatikusan növekvő érték (curdate).
* f\_id: Egész szám, idegen kulcs, ebből az értékből tudjuk meg, hogy melyik felhasználóhoz tartozik az adott edzés.
* edzesterv\_id: Egész szám, idegen kulcs, megmutatja, hogy az adott edzések milyen gyakorlatok kerültek elvégzésre.

Az **edzéstervek** tábla tulajdonságai:

* edzesterv\_id: Egész szám, elsődleges kulcs, automatikusan növekvő érték, az edzésterv azonosítója.
* gyakorlat\_id: Egész szám, idegen kulcs, a gyakorlat azonosítója.
* sorozatszam: Egész szám, megmutatja, hogy az adott gyakorlatból hány sorozat kerül elvégzésre.
* ismetlesszam: Egész szám, megmutatja, hogy az adott gyakorlatból hány ismétlés kerül elvégzésre egy sorozaton belül.

A **gyakorlatok** tábla tulajdonságai:

* gyakorlatok\_id: Egész szám, elsődleges kulcs, automatikusan növekvő érték, a gyakorlat azonosítója.
* gynev: Szöveges típusú, a gyakorlat neve.
* izomcsoport\_id: Egész szám, idegen kulcs, az izomcsoport azonosítója amelyhez a gyakorlat tartozik.
* leiras: Szöveges típusú, információ a gyakorlatról.

Az **izomcsoportok** tábla tulajdonságai:

* izomcsoport\_id: Egész szám, elsődleges kulcs, az izomcsoport azonosítója.
* izomcsoport: Szöveges típusú, az izomcsoport megnevezése.

Az **étkezések** tábla tulajdonságai:

* etkezesek\_id: Egész szám, elsődleges kulcs, automatikusan növekvő érték, az étkezés azonosítója.
* idopont: Datetime típusú érték, megmutatja, hogy mikor volt az adott étkezés.
* etel\_id: Egész szám, idegen kulcs, az étel azonosítója, amely fogyasztva volt az étkezésen belül.
* f\_id: Egész szám, idegen kulcs, a felhasználó azonosítója, akihez az étkezés tartozik.

Az **ételek** tábla tulajdonságai:

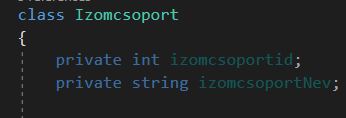
* etel\_id: Egész szám, elsődleges kulcs, automatikusan növekvő érték, az étel azonosítója.
* enev: Szöveges típusú, az étel neve.
* szenhidrat: Egész szám, az étel szénhidrát tartalma.
* feherje: Egész szám, az étel fehérje tartalma.
* zsir: Egész szám, az étel zsír tartalma.
* kaloria: Egész szám, az étel energia tartalma.
* mennyiség: Szöveges típusú, az adott mennyiség az ételből, amelyben megtalálható a fehérje, szénhidrát, zsír és kalória mértéke.

## **Asztali alkalmazás**

Az asztali alkalmazás elkészítéséhez egy olyan programozási nyelvet választottam, amiben már jártas vagyok. Ezért a választás a C#-ra esett. Az iskolai tanulmányaim során is a legtöbb projektet e nyelv segítségével oldottuk meg, így rögtön ezt választottam. Nagyon fontos hogy ezt a programozási nyelvet egy platformon lehet használni, nem platformfüggetlen mint például a Java. A fejlesztői környezet kiválasztása sem okozott gondot, a legoptimálisabb a Microsoft Visual Studio. Mivel egyszerű kezelni, ingyenes, könnyen személyre lehet szabni a felhasználói felületet és rengeteg ingyenes keretrendszer letölthető hozzá. Ennek a programnak a 2019-es verzióját használtam. Viszont ami a legfontosabb, hogy a Visual Studio képes kapcsolatot létesíteni olyan verziókövető rendszerrel, mint például a GitHub, és felhőben is képes eltárolni a program aktuális állapotát. Egy ilyen nagy és hosszadalmas munkánál elengedhetetlen a verziókövető rendszerek használata.

A program fejlesztése során igyekeztem olyan környezetet kialakítani, amely felhasználóbarát és könnyű kezelhetőséget biztosít. A fejlesztés során igyekeztem az objektum orientált programozás mellett maradni, így hát nem újra és újra írtam a kódokat, hanem mindenre metódust készítettem, amiket többször is meg lehet hívni. Így felesleges kódsorokat spóroltam meg.

Az objektum orientált programozás olyan módszereket nyújt nekünk fejlesztőknek, amelyek lehetővé teszik a programok bonyolultságának csökkentését, a megbízhatóság és a hatékonyság növelését. Az OOP objektumokból építi fel a programot. Az objektum adatokat tárol, és kérésre tevékenységet végez.

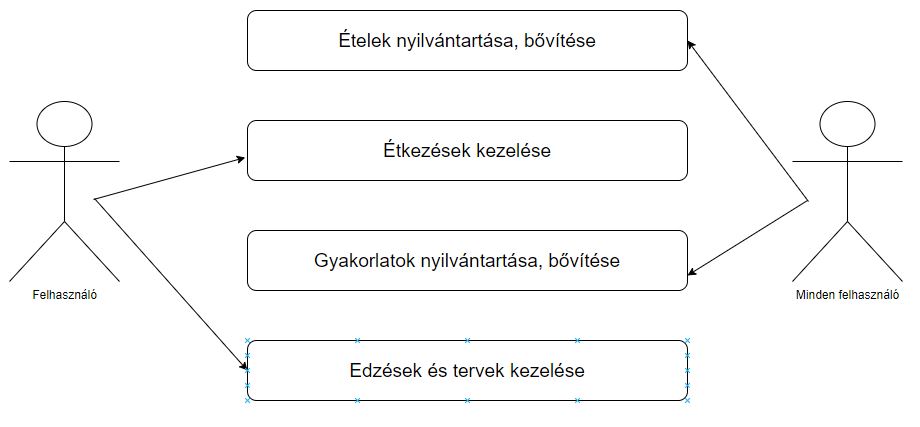
**

2. ábra Objektum a hozzátartozó attribútumokkal

A felhasználói felület minőségét befolyásolja a felhasznált színek száma és fajtája valamint az olvashatóság, így igyekeztem olyan ablakokat létrehozni ahová színes elemeket tettem fel, így sokkal áttekinthetőbb és vonzóbb lett. Ugyanakkor nem használtam túl világos színeket mivel a túl világos színek egy idő után nem tesznek jót az ember szemének.

### **Felhasználok**

A felhasználók feladatainak szemléltetésére a USE-CASE diagramot használjuk. A használat eset diagram egy olyan diagram, amely bemutatja a fejlesztő által elkészített program felhasználóinak a feladatköreit. A mi esetünkben ez a következőképpen néz ki:



. ábra USE-CASE diagram

Ahogy az ábra is szemlélteti, vannak olyan feladatkörök, amit minden felhasználó egyaránt el tud végezni. Azonban van olyan feladat, amit csak egy felhasználó tud elvégezni. Például Józsi szeretné látni egy másik felhasználó étkezéseit. Azonban erre nincs lehetőség ugyanis az étkezés funkció személyre szabottan van elkészítve.

### **Az alkalmazás**

* Bejelentkezés:

Mivel egy olyan rendszerről beszélünk, amely személyre szabottan működik, így elengedhetetlen az azonosítás, ahogy a legtöbb program esetében is szükséges, hogy a felhasználó jogosult-e a program használatára. Erre egy olyan ablakot készítettem, ahol a felhasználónév és jelszó megadása után kap egy visszaigazolást a rendszertől, hogy van e jogosultsága bejelentkezni vagy nincs.

Amint a felhasználó rákattint a bejelentkezés gombra, azonnal lefut egy kérés az adatbázis felé, amely a Login osztályon belül található, ez a kérés ellenőrzi, hogy létezik-e a megadott felhasználónév és jelszó. Ha nem létezik a felhasználónév vagy valamelyik adatot rosszul adott meg a kliens, akkor egy hibaüzenet jelenik meg és a rendszer nem lép. Azonban ha a kliens mindent helyesen adott meg akkor a ,,Sikeres Bejelentkezés” üzenet után a program átirányítja a fő menüre.

* Bejelentkezés után

Amennyiben sikeres volt a bejelentkezés azt követően a felhasználó előtt megjelenik egy úgynevezett üdvözlő form. Itt lehetőség nyílik a testtömegindex kiszámítására valamint a menüsávból egy kattintással elérhetővé válnak az ételek vagy az étkezéseink.

Ha szeretnénk megtekinteni az ételeket akkor válasszuk az ételek menüpontot majd megjelenik egy új ablak ami után elénk tárulnak egy ComboBoxban az ételek. Az étel kiválasztása után egy DataGridView-ban láthatóvá válnak az étel adatai.

Az étel hozzáadása gombra kattintva új ételt lehet hozzáadni az adatbázishoz. Ha minden mező helyesen került kitöltésre akkor megtörténik a sikeres hozzáadás.

if (!ujEtel.validate())

{

return;

}

else

{

//Beszúrás listába

r.addEtelToList(ujEtel);

//Beszúrás adatbázisba

r.insertEtelToDatabase(ujEtel);

//Frissít DataGridView

beallitDataGridView();

FormSucces fs = new FormSucces("Étel mentése sikeres!");

DialogResult result = fs.ShowDialog();

if (result == DialogResult.OK)

{

panelEtel.Visible = false;

fs.Hide();

feltoltComboboxEtelek();

}

Amennyiben törölni szeretnénk az ételek közül úgy az étel törlés gomb kiválasztása után ellenőrzésre kerül, hogy a DataGridView-ban van e kiválasztva sor. Ha van akkor a program megkérdezi a klienst, hogy biztosan szeretne e törölni.

FormMessage fm = new FormMessage("Biztos törölni szeretnél?");

DialogResult result = fm.ShowDialog();

if (result == DialogResult.OK)

{

fm.Hide();

try

{

//Törlés adatbázisból

r.deleteEtelFromDataBase(Convert.ToInt32(dataGridViewEtelek.SelectedRows[0].Cells[0].Value));

//Törlés listából

r.deleteEtelFromList(Convert.ToInt32(dataGridViewEtelek.SelectedRows[0].Cells[0].Value));

}

catch(RepositoryException ex)

{

Debug.WriteLine(ex.Message);

FormError fe = new FormError("Sikertelen törlés, az étel tagja egy étkezésnek!");

DialogResult dr = fe.ShowDialog();

if (dr == DialogResult.OK)

fe.Hide();

}

//DataGridView frissítés

feltoltComboboxEtelek();

beallitDataGridView();

Amennyiben a törölni kívánt ételt egy étkezés tartalmazza ez jelzésre kerül a felhasználó fele és ebben az esetben az étel nem törölhető.

Egy ételt hozzáadhatunk étkezéshez a megfelelő menü kiválasztásával. Ha megtörtént erre a gombra a kattintás megjelenik egy új panel amin ki kell válasszuk az étkezés időpontját. Amint sikerült kiválasztani az időpontot úgy az étel adatai kiolvasásra kerülnek az adattáblából.

Az étkezések a kezdő formon az étrend gombra kattintás után válnak elérhetővé. Egy ComboBoxból kell kiválasztanunk az ott megjelenített dátumot, amelyek az ablak betöltődésekor a konstruktorban kerülnek feltöltésre. A dátum kiválasztása után válnak láthatóvá az étkezéseink. A bal alsó sarokban kiszámításra kerül a napi makro tápanyag és kalória bevitelünk.

public int getOsszFeherje(string idopont)

{

List<EtkezesView> etkezesview = etkezesekviewn.FindAll(x => x.Idopont == idopont);

int totalfeherje = 0;

foreach(EtkezesView evv in etkezesview)

{

totalfeherje = totalfeherje + evv.Feherje;

}

return totalfeherje;

}

public int getOsszKaloria(string idopont)

{

List<EtkezesView> etkezesview = etkezesekviewn.FindAll(x => x.Idopont == idopont);

int totalkaloria = 0;

foreach (EtkezesView evv in etkezesview)

{

totalkaloria = totalkaloria + evv.Kaloria;

}

return totalkaloria;

}

Ezen a formon is elérhető a törlés funkció. Ha van kiválasztva sor akkor felugrik egy ablak amelyen megerősíthető a törlés.

### **Tesztelés**

A szoftverfejlesztés során elengedhetetlen az egységtesztek készítése. Ezekkel igazolható, hogy a programunk jól működik és megfelel a specifikációnak valamint eleget tesz a megrendelőnek. A teszttel nem csak az elkészített kódot lehet leellenőrizni, hanem fel is lehet használni például input adatok ellenőrzése során. Az alkalmazásban nem mindenhez készült teszt vagy ellenőrzés de az étel hozzáadása során a mentés gomb megnyomásakor ellenőrzésre kerülnek a bemeneti adatok. Ha valamilyen hiba lépne fel a program kivételt dob az adott problémát megjelölve és egy errorProvider segítségével jelzésre kerül az adott mező mellett. Az első betű ellenőrzésére mutatnék egy tesztet példaképpen:

[TestMethod()]

public void validateTestEnevFirstLetterUpperCase()

{

try

{

Etel etel = new Etel("Csirkemell",2,3,4,5,"3g");

if (etel.validate())

{

}

}

catch (Exception e)

{

Assert.Fail("Az étel első betűje nagybetű mégis hibát dob");

}

}

## **Web alkalmazás**

A web alkalmazás elkészítése során törekedtem arra, hogy a kezdő oldal minél átláthatóbb legyen ezáltal nem használtam sok színt és kevés szöveget helyeztem el az oldalon. A fejlesztés során alkalmaztam programozási nyelveket és jelölőnyelvet is egyaránt. Az oldal vázszerkezetét HTML jelölőnyelv segítségével készítettem el. Emellett a dizájnhoz igénybe vettem a Bootstrap 4- es verzióját valamint CSS stíluslapokat is. Igyekeztem reszponzívvá alakítani, hogy minden digitális eszközön jól olvasható legyen a tartalom. Valamint a front-end oldalon használtam még jquery és javascript függvényeket.

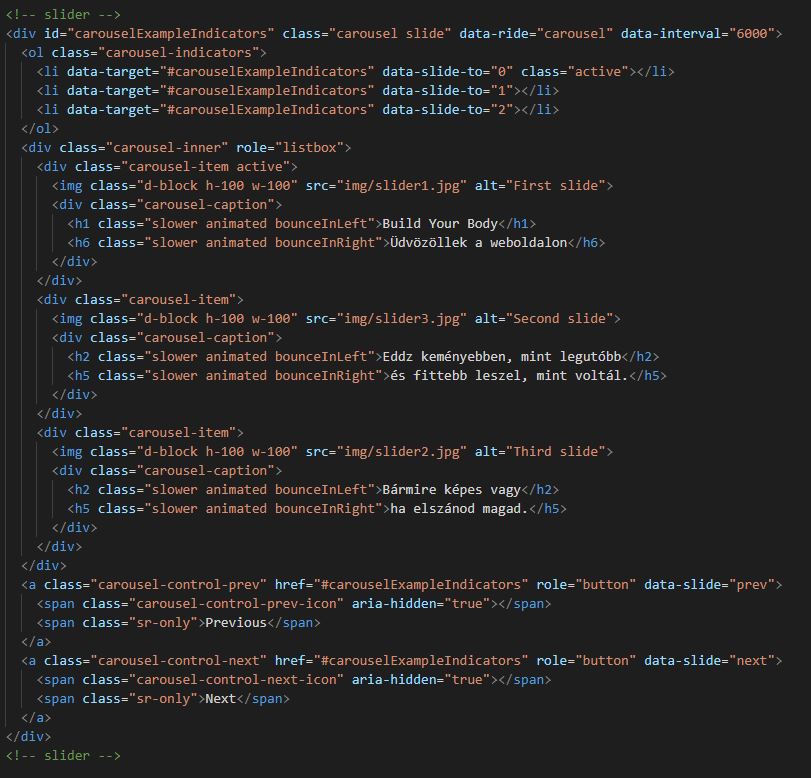
A backend oldalon a PHP programozási nyelvet választottam. Ezt a nyelvet használtam a legtöbbet a tanulmányaim során így tehát ez nem is volt kérdés mivel dolgozzak a backenden.

Ahhoz, hogy minél élvezhetőbbé tegyem a weboldalt igénybe vettem még az ajax technológiát is. Ezáltal lehetővé válik a szerverrel való kapcsolattartás a teljes oldal újratöltése nélkül. Növelve a honlap sebességét és használhatóságát.

A fejlesztői környezet kiválasztása sem okozott problémát. Itt a Visual Studio Code nevezetű IDE-re esett a választás. Ugyanis számos olyan funkcióval rendelkezik amelyek segítették a munkámat a kódolás során. Vegyük példaként a formatter funkcióját amely egy gombkombinációra tagolja a több száz soros kódot. Ez a fejlesztés végéhez közeledve nagyon sokat segített, hogy minél jobban át lássam az általam megírt forráskódot.

### **Index**

A kezdőoldalt másnéven indexnek nevezzük. Egy oldal betöltődésekor ez lesz az amit automatikusan keresni fog a böngésző a fájlok között. A kezdőoldalt fontosnak tartottam, mert úgymond egy képet ad magáról az egész alkalmazásról. Amikor betöltődik az oldal a felhasználó egy Image slider-el találja magát szemben, amin elhelyeztem három darab idézetet. Az oldal tetején található még egy navigációs sáv, amelyen ha kattintunk egy menüre úgy mond oda scrollol a megfelelő tartalomhoz.

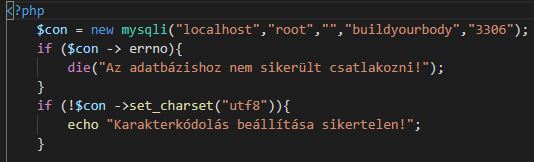


. ábra Image-slider az index.php-n

Maga az oldal több section-re van felosztva így könnyítve az átláthatóságot.

### **Csatlakozás az adatbázishoz**

Az adatbázishoz való csatlakozást a connect.php nevezetű fájlba oldottam meg amely a php mappán belül található. Később becsatolásra kerül az init.php fájlba amiben indításra kerül egy session amely elengedhetetlen például a bejelentkezés során.



5. ábra Csatlakozás az adatbázishoz

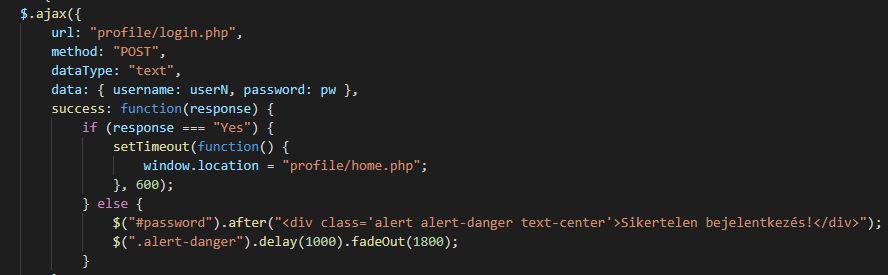
### **Bejelentkezés és regisztráció**

* Regisztráció

A regisztráció az egész rendszer alapja. Mivel az asztali alkalmazáson erre nincs lehetőség így a weboldalon keresztül a látogatók feltudnak regisztrálni és később a szoftverhez is hozzáférhetnek. Fontosnak tartom, hogy a regisztráció során a kliens valós és helyes adatokat adjon meg. Például az email cím formátum érvényessége. Ezek ellenőrzése megtörténik kliens oldalon jQuery és RegExp segítségével. Az adatokat utána Ajaxal küldöm át szerver oldalra a reg.php nevű állományba, majd ezen keresztül viszem fel az adatbázisba.

* Bejelentkezés

A bejelentkezést úgy oldottam meg, hogy a menüsávban a belépés gombra kattintva egy modal ablak jelenik meg. Az első mezőbe a felhasználónév a másodikba a jelszó kerül. Ha helyesen van kitöltve akkor egy ajax kéréssel fut le, majd átirányításra kerül az oldal a home.php-ra. Ha a bejelentkezés sikertelen nem töltődik újra az oldal hanem a kliens oldalról hiba üzenetet jelenitek meg. Íme néhány kódrészlet:



6. ábra Ajax kérés belépéskor



. ábra Login.php

### **Profil**

## **Fejlesztési tervek**

Sajnos rengeteg olyan funkció nem szerepel a szoftverben amely hasznos lenne. Ezek az idő hiánya miatt nem készültek el. Itt ezeket a funkciókat sorolom fel, hogy az alkalmazás egy következő életciklusba lépjen és egy kisebb célközönség használatba is vegye. A közeljövőben megvalósítandó tervek:

* Kommunikáció a felhasználók között (chat funkció)
* Személyi edzők menüpont létrehozása. Edzők hozzáadása az applikációhoz.
* Edzők a vendégeik számára edzéstervet és étrendet készíthetnek.
* Telefonos alkalmazás készítése

1. 1 testtömeg-index [↑](#footnote-ref-1)