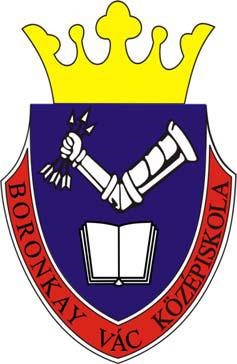
**VÁCI SZAKKÉPZÉSI CENTRUM   
BORONKAY GYÖRGY   
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM**

**SZAKDOLGOZAT**

**Gimesi Deniel, Gimesi Gergely**

**2022.**

**VÁCI SZAKKÉPZÉSI CENTRUM   
BORONKAY GYÖRGY   
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM**



**SZAKDOLGOZAT**

**E-Budget**

|  |  |
| --- | --- |
| Konzulens: Blahut Lóránt | Készítette: Gimesi Deniel 14.P  Gimesi Gergely 14.P |

# **Hallgatói nyilatkozat**

Alulírottak, ezúton kijelentjük, hogy a szakdolgozat saját, önálló munkánk, és korábban még sehol nem került publikálásra.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Gimesi Deniel |  | Gimesi Gergely |

# **Konzultációs lap**

Vizsgázók neve: Gimesi Deniel, Gimesi Gergely

Szakdolgozat címe: E-Budget

Program nyújtotta szolgáltatások:

* A felhasználó költségeinek kezelése
* Tranzakciók hozzáadása, módosítása és törlése
* Bevételi és kiadási kategóriák hozzáadása, módosítása és törlése
* Kiadási alkategóriák hozzáadása, módosítása és törlése
* Tranzakciók szűrése
* Különböző pénznemű tranzakciók átváltása
* Statisztika diagrammokon való megjelenítése

|  |  |  |
| --- | --- | --- |
| Sorszám | A konzultáció időpontja | A konzulens aláírása |
| 1. | 2021.10.01 |  |
| 2. | 2021.12.01 |  |
| 3. | 2022.01.04 |  |
| 4. | 2022.02.01 |  |
| 5. | 2022.03.01 |  |
| 6. | 2022.04.01 |  |

|  |  |  |
| --- | --- | --- |
| A szakdolgozat beadható: |  | A szakdolgozatot átvettem: |
| Vác, 2022. |  | Vác, 2022. |
| Konzulens |  | A szakképzést folytató intézmény felelőse |

**Tartalomjegyzék**

[**Hallgatói nyilatkozat** 3](#_Toc100489597)

[**Konzultációs lap** 4](#_Toc100489598)

[**Tartalomjegyzék** 5](#_Toc100489599)

[**Témaválasztás** 7](#_Toc100489600)

[**1** **Fejlesztői dokumentáció** 8](#_Toc100489601)

[1.1 Feladatspecifikáció 8](#_Toc100489602)

[1.2 Fejlesztői környezet 9](#_Toc100489603)

[1.2.1 SQL, MySQL 9](#_Toc100489604)

[1.2.2 C#, .NET Framework 10](#_Toc100489605)

[1.2.3 PHP 11](#_Toc100489606)

[1.2.4 Git, GitHub 11](#_Toc100489607)

[1.2.5 Fejlesztés során használt programok 12](#_Toc100489608)

[1.3 Technológiák 13](#_Toc100489609)

[1.3.1 MVC 13](#_Toc100489610)

[1.3.2 MVP 14](#_Toc100489611)

[1.4 Adatszerkezet 15](#_Toc100489612)

[1.4.1 A „users” tábla 17](#_Toc100489613)

[1.4.2 A „currencies” tábla 18](#_Toc100489614)

[1.4.3 A „categories\_expense” tábla 19](#_Toc100489615)

[1.4.4 A „categories\_income” tábla 20](#_Toc100489616)

[1.4.5 A „subcategories\_expense” tábla 21](#_Toc100489617)

[1.4.6 A „transactions\_expense” tábla 22](#_Toc100489618)

[1.4.7 A „transactions\_income” tábla 23](#_Toc100489619)

[1.4.8 A „view\_alltransactions” nézet 24](#_Toc100489620)

[1.5 Fontosabb függvények bemutatása 26](#_Toc100489621)

[1.5.1 Regisztráció 26](#_Toc100489622)

[1.5.2 Bejelentkezés 30](#_Toc100489623)

[1.5.3 „Salting” és „Hashing” 33](#_Toc100489624)

[1.5.4 Szűrés 35](#_Toc100489625)

[1.5.5 Pénznem átváltás 37](#_Toc100489626)

[1.6 Tesztelés 39](#_Toc100489627)

[1.6.1 Regisztráció 39](#_Toc100489628)

[1.6.2 Bejelentkezés 41](#_Toc100489629)

[1.6.3 Tranzakció hozzáadása 42](#_Toc100489630)

[1.6.4 Tranzakció törlése 42](#_Toc100489631)

[1.6.5 Felhasználó törlése 43](#_Toc100489632)

[1.7 Fejlesztési lehetőségek 43](#_Toc100489633)

[**2** **Felhasználói dokumentáció** 44](#_Toc100489634)

[2.1 Program leírása 44](#_Toc100489635)

[2.2 Bejelentkezési és Regisztrációs felület 44](#_Toc100489636)

[2.2.1 Bejelentkezés 45](#_Toc100489637)

[2.2.2 Regisztráció 46](#_Toc100489638)

[2.3 Regisztrált és bejelentkezett felhasználói felület 48](#_Toc100489639)

[2.3.1 „Budget” felület 48](#_Toc100489640)

[2.3.2 „Categories” felület 50](#_Toc100489641)

[2.3.3 „Reports” felület 52](#_Toc100489642)

[2.3.4 „Settings” felület 53](#_Toc100489643)

[2.3.5 „Add transaction” felület 55](#_Toc100489644)

[**3** **Irodalomjegyzék** 57](#_Toc100489645)

[**4** **Mellékletek** 58](#_Toc100489646)

# **Témaválasztás**

Szakdolgozatunk témájaként egy költségvetés kezelő programot választottunk, mely asztali és webes felületen is elérhető.

Ez a program hasznos lehet családok, kisebb vállalkozások számára mivel a piacon elérhető költségvetés kezelő alkalmazások többsége nem rendelkezik megfelelő funkciókkal, túlárazott vagy tele van reklámokkal.  
  
Mind az asztali és webes felület sha 256-os hashelést használ, amit az email alapján generált „salt”-al saltol a program. Ennek eredményeként egy nem visszafejthető biztonságosan tárolható hash kulcsot kapunk, amivel biztosítjuk felhasználóink adatainak biztonságát.

A legtöbb piacon fellelhető alkalmazás nem tartalmaz pénznem átváltási funkciót, ennél a szempontnál fogva az E-Budget olyan családok, vállalkozások számára is megfelelő lehet, akiknek több pénznemben is vannak kiadásaik és bevételeik.

A feladatokat asztali és webes felület elkészítésére osztottuk szét.  
Az asztali alkalmazást Gimesi Deniel, a webes alkalmazást Gimesi Gergely készítette.   
A dokumentációt közösen készítettük.

# **Fejlesztői dokumentáció**

## Feladatspecifikáció

Látogatók számára elérhető funkciók:

* Regisztráció
* Bejelentkezés

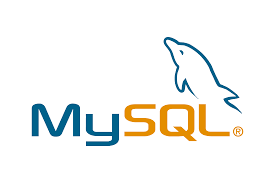
Felhasználók számára elérhető funkciók:

* Tranzakciók hozzáadása
  + módosítása
  + törlése
* Bevételi és Kiadási kategóriák hozzáadása
  + módosítása
  + törlése
* Kiadási alkategóriák hozzáadása
  + módosítása
  + törlése
* Aktuális egyenleg megtekintése
* Tranzakciók szűrése
  + típus
  + kategória
  + alkategória
  + pénznem
  + dátum
* Szűrt tranzakciók bevételi és kiadási összegének megtekintése
* Statisztikák megtekintése
* Felhasználói adatok módosítása
  + név
  + email
  + jelszó
  + pénznem
* Felhasználó törlése
* Kijelentkezés

## Fejlesztői környezet

### SQL, MySQL

Az **SQL**, azaz **Structured Query Language** (strukturált lekérdezőnyelv) relációsadatbázis-kezelők lekérdezési nyelve. Az SQL nyelv segítségével, a különböző típusú adatbázisokat azonos módon kezelhetjük.

****A MySQL egy többfelhasználós, többszálú, SQL-alapú relációs adatbázis-kezelő szerver.   
A MySQL az egyik legelterjedtebb adatbázis-kezelő, aminek egyik oka lehet, hogy a teljesen nyílt forráskódú LAMP (Linux–Apache–MySQL–PHP) összeállítás részeként költséghatékony és egyszerűen beállítható megoldást ad dinamikus webhelyek szolgáltatására.

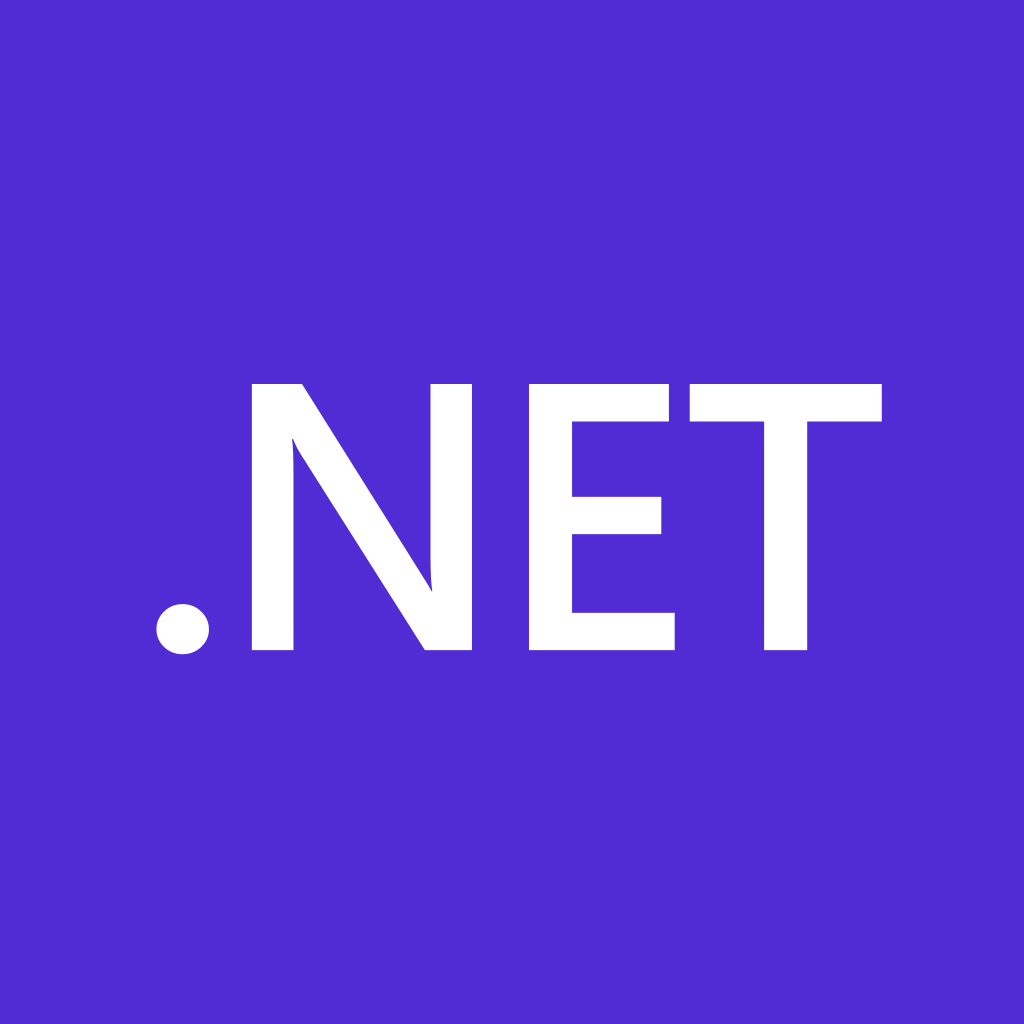
Forrás: https://hu.wikipedia.org/wiki/MySQL  
https://hu.wikipedia.org/wiki/SQL

### C#, .NET Framework

A C# a Microsoft által a .NET keretrendszer részeként kifejlesztett objektumorientált programozási nyelv. A nyelv alapjául a C++ és a Java szolgált.

A Microsoft által készített .NET keretrendszer (a .NET Framework) gyors alkalmazásfejlesztést (RAD), platformfüggetlenséget és hálózati átlátszóságot támogató szoftverfejlesztői platform. A keretrendszert a korábbi platform, a COM leváltására szánták.

Eredetileg a .NET kifejezés nemcsak fejlesztői környezetet jelentett, hanem fejlesztőeszközök, szoftverek, sőt hardvereszközök összességét is. Az évek során a kép kitisztult, így mostanra a .NET alatt a keretrendszert értjük.

A .NET Framework eszköztára a szoftverfejlesztés szinte minden aspektusát (kliens-, illetve szerveroldali megoldások, adatbázisok kezelése, játékfejlesztés stb.) lefedi.

Forrás: https://hu.wikipedia.org/wiki/.NET\_keretrendszer  
https://hu.wikipedia.org/wiki/C\_Sharp

### PHP

A PHP egy általános szerveroldali szkriptnyelv dinamikus weblapok készítésére. Az első szkriptnyelvek egyike, amely külső fájl használata helyett HTML oldalba ágyazható. A kódot a webszerver PHP feldolgozómodulja értelmezi, ezzel dinamikus weboldalakat hozva létre.  
A hagyományos HTML lapokkal ellentétben a kiszolgáló a PHP-kódot nem küldi el az ügyfélnek, hanem a kiszolgáló oldalán a PHP-értelmező motor dolgozza fel azt. A programokban lévő HTML elemek érintetlenül maradnak, de a PHP kódok lefutnak. A kódok végezhetnek adatbázis-lekérdezéseket, létrehozhatnak képeket, fájlokat olvashatnak és írhatnak, kapcsolatot létesíthetnek távoli kiszolgálókkal. A PHP-kódok kimenete a megadott HTML elemekkel együtt kerül az ügyfélhez.



Forrás: https://hu.wikipedia.org/wiki/PHP

### Git, GitHub

A Git egy nyílt forráskódú, elosztott verziókezelő szoftver, vagy másképpen egy szoftverforráskód-kezelő rendszer, amely a sebességre helyezi a hangsúlyt. A Gitet eredetileg Linus Torvalds fejlesztette ki a Linux kernel fejlesztéséhez. Minden Git munkamásolat egy teljes értékű repository teljes verziótörténettel és teljes revíziókövetési lehetőséggel, amely nem függ a hálózat elérésétől vagy központi szervertől.

A GitHub, Inc. egy egyesült államokbeli nemzetközi vállalat, amely a Git segítségével szoftverfejlesztési verziókövetés-szolgáltatást nyújt. 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.

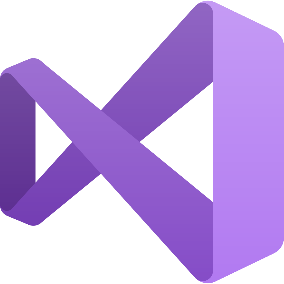


Forrás: https://hu.wikipedia.org/wiki/Git  
https://hu.wikipedia.org/wiki/GitHub

### Fejlesztés során használt programok

Visual Studio 2022

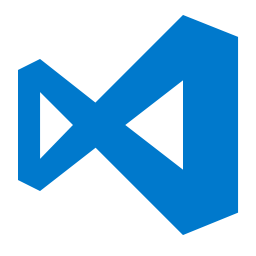
A Visual Studio a Microsoft több programozási nyelvet tartalmazó fejlesztőkörnyezete, amely az évek során egyre több új programnyelvvel bővült. Jelenleg a F#, C++, C# és Visual Basic programozási nyelveket, valamint az XML-t támogatja. A csomag része még a MASM (Microsoft Macro Assembler) is, ami részleges assembly támogatást biztosít.



Forrás: https://hu.wikipedia.org/wiki/Microsoft\_Visual\_Studio

Visual Studio Code

A Visual Studio Code (rövidítve: VSCode vagy VS Code) egy ingyenes, nyílt forráskódú kódszerkesztő, melyet a Microsoft fejleszt Windows, Linux és OS X operációs rendszerekhez. Támogatja a hibakeresőket, valamint beépített Git támogatással rendelkezik, továbbá képes az intelligens kódkezelésre (intelligent code completion) az IntelliSense segítségével. Ezen felül testre szabható, így a felhasználók megváltoztathatják a kinézetet (témát), megváltoztathatják a szerkesztő gyorsbillentyű-kiosztását, az alapértelmezett beállításokat és még sok egyebet.



Forrás: https://hu.wikipedia.org/wiki/Visual\_Studio\_Code

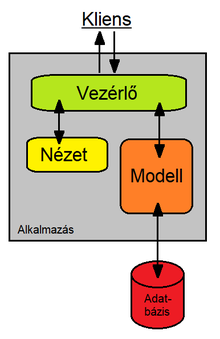
## Technológiák

### MVC

A webes felület MVC technológia segítségével lett felépítve.

A modell-nézet-vezérlő (MNV) (angolul model-view-controller) a szoftvertervezésben használatos programtervezési minta. Összetett, sok adatot a felhasználó elé táró számítógépes alkalmazásokban gyakori fejlesztői kívánalom az adathoz (modell) és a felhasználói felülethez (nézet) tartozó dolgok szétválasztása, hogy a felhasználói felület ne befolyásolja az adatkezelést, és az adatok átszervezhetők legyenek a felhasználói felület változtatása nélkül. A modell-nézet-vezérlő ezt úgy éri el, hogy elkülöníti az adatok elérését és az üzleti logikát az adatok megjelenítésétől és a felhasználói interakciótól egy közbülső összetevő, a vezérlő bevezetésével.

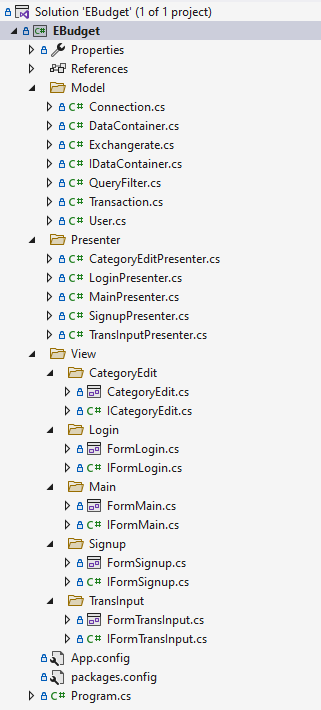
Hagyományosan asztali felhasználói felületekhez használt, de manapság már webalkalmazásokhoz is népszerűvé vált. Népszerű programozási nyelvek, mint a JavaScript, Python, Ruby, PHP, Java, C# és Swift már külön telepítés szükségessége nélkül rendelkeznek MNV keretrendszerekkel web- és mobilalkalmazások fejlesztésére.



Forrás: https://hu.wikipedia.org/wiki/Modell-nézet-vezérlő

### MVP

Az asztali felület MVP technológiát alkalmaz.

A **modell-nézet-prezenter** (angolul: *Modell-View-Presenter*, röviden: MVP) egy szoftvertervezési minta, amely az MVC-mintából alakult ki. Az MVP-modell főleg olyan alkalmazásokban nyújt jelentős előnyöket – hasonlóan az MVC-hez –, ahol komplex adathalmazokon kell műveleteket végezni, és ezek eredményeit a felhasználó elé tárni. A mintát felhasználó alkalmazások a minta sajátosságaiból eredően könnyedén alávethetőek ún. egységteszteknek (angolul *unit test*).

Forrás: https://hu.wikipedia.org/wiki/Modell-nézet-prezenter

## Adatszerkezet

MySQL (MariaDB) kliens verzió: 10.4.20

Adatbázis neve: e-budget

Tároló motor: InnoDB

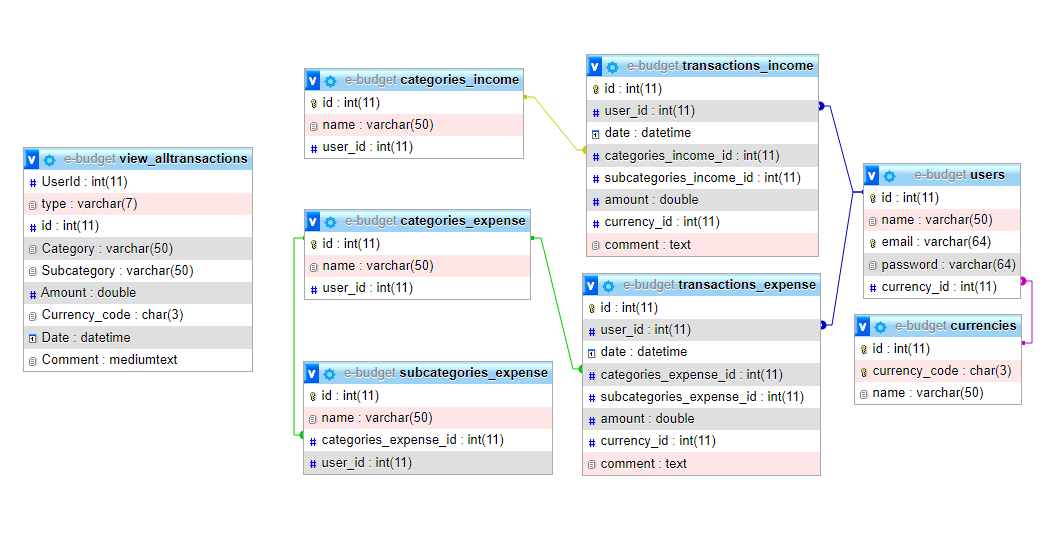
Alapértelmezett illesztés: utf8\_hungarian\_ci

**Adattáblák:**

* users
* currencies
* categories\_expense
* categories\_income
* subcategories\_expense
* transactions\_expense
* transactions\_income

**Nézet:**

* view\_alltransactions

**Adattáblák közötti kapcsolatok:**

### A „users” tábla

Ez a tábla tartalmazza a felhasználói adatokat.

**id:** a felhasználó azonosítója

**name:** a felhasználó neve

**email:** a felhasználó email címe (egyedi)

**password:** a felhasználó jelszava

**currency\_id:** a felhasználó pénznemének id-ja

**Létrehozása:**

CREATE TABLE `users` (

`id` int(11) NOT NULL,

`name` varchar(50) COLLATE utf8\_hungarian\_ci NOT NULL,

`email` varchar(64) COLLATE utf8\_hungarian\_ci NOT NULL,

`password` varchar(64) COLLATE utf8\_hungarian\_ci NOT NULL,

`currency\_id` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_hungarian\_ci;

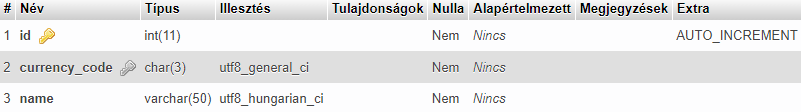
ALTER TABLE `users`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `email` (`email`),

ADD KEY `FK\_users` (`currency\_id`);

### A „currencies” tábla

****Ez a tábla tartalmazza az elérhető pénznemeket.

**id:** a pénznem azonosítója

**currency\_code:** a pénznem kódja

**name:** a pénznem neve

**Létrehozása:**

CREATE TABLE `currencies` (

`id` int(11) NOT NULL,

`currency\_code` char(3) CHARACTER SET utf8 NOT NULL,

`name` varchar(50) COLLATE utf8\_hungarian\_ci NOT NULL

) ENGINE=InnoDB AVG\_ROW\_LENGTH=2340 DEFAULT CHARSET=utf8 COLLATE=utf8\_hungarian\_ci;  
  
ALTER TABLE `currencies`

ADD PRIMARY KEY (`id`),

ADD UNIQUE KEY `currency\_code` (`currency\_code`);

### A „categories\_expense” tábla

Ez a tábla tartalmazza a kiadási kategóriákat.

**id:** a kategória azonosítója

**name:** a kategória neve

**user\_id:** a felhasználó azonosítója

**Létrehozása:**

CREATE TABLE `categories\_expense` (

`id` int(11) NOT NULL,

`name` varchar(50) COLLATE utf8\_hungarian\_ci NOT NULL,

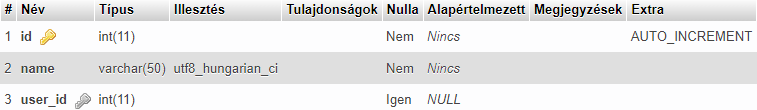
`user\_id` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_hungarian\_ci;  
  
ALTER TABLE `categories\_expense`

ADD PRIMARY KEY (`id`),

ADD KEY `FK\_categories\_expense` (`user\_id`);

### A „categories\_income” tábla

Ez a tábla tartalmazza a bevételi kategóriákat.

**id:** a kategória azonosítója

**name:** a kategória neve

**user\_id:** a felhasználó azonosítója

**Létrehozása:**

CREATE TABLE `categories\_income` (

`id` int(11) NOT NULL,

`name` varchar(50) COLLATE utf8\_hungarian\_ci NOT NULL,

`user\_id` int(11) DEFAULT NULL

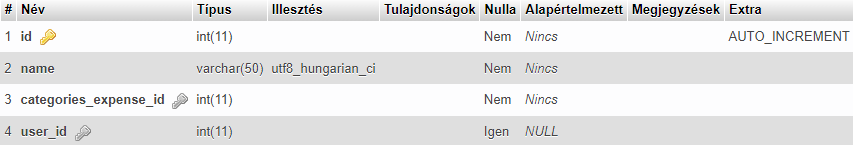
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_hungarian\_ci;

ALTER TABLE `categories\_income`

ADD PRIMARY KEY (`id`),

ADD KEY `FK\_categories\_income` (`user\_id`);

### A „subcategories\_expense” tábla

Ez a tábla tartalmazza a kiadási alkategóriákat.

**id:** az alkategória azonosítója

**name:** az alkategória neve

**categories\_expense\_id:** a kiadási kategória azonosítója

**user\_id:** a felhasználó azonosítója

**Létrehozása:**

CREATE TABLE `subcategories\_expense` (

`id` int(11) NOT NULL,

`name` varchar(50) COLLATE utf8\_hungarian\_ci NOT NULL,

`categories\_expense\_id` int(11) NOT NULL,

`user\_id` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_hungarian\_ci;

ALTER TABLE `subcategories\_expense`

ADD PRIMARY KEY (`id`),

ADD KEY `FK\_subcategories\_e\_cat\_e\_id` (`categories\_expense\_id`),

ADD KEY `user\_id` (`user\_id`);

### A „transactions\_expense” tábla

Ez a tábla tartalmazza a kiadási tranzakciókat.

**id:** a tranzakció azonosítója

**user\_id:** a felhasználó azonosítója

**date:** a tranzakció dátuma

**categories\_expense\_id:** a kiadási kategória azonosítója

**subcategories\_expense\_id:** a kiadási alkategória azonosítója

**amount:** a tranzakció összege

**currency\_id:** a pénznem azonosítója

**comment:** a tranzakció megjegyzése

**Létrehozása:**

CREATE TABLE `transactions\_expense` (

`id` int(11) NOT NULL,

`user\_id` int(11) NOT NULL,

`date` datetime NOT NULL,

`categories\_expense\_id` int(11) NOT NULL,

`subcategories\_expense\_id` int(11) NOT NULL,

`amount` double NOT NULL,

`currency\_id` int(11) NOT NULL,

`comment` text COLLATE utf8\_hungarian\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_hungarian\_ci;

ALTER TABLE `transactions\_expense`

ADD PRIMARY KEY (`id`),

ADD KEY `FK\_transactions\_expense` (`currency\_id`),

ADD KEY `FK\_transactions\_expense\_subcategories\_expense\_id` (`subcategories\_expense\_id`),

ADD KEY `FK\_transactions\_expense\_users\_id` (`user\_id`),

ADD KEY `FK\_transactions\_expense\_categories\_expense\_id` (`categories\_expense\_id`);

### A „transactions\_income” tábla

Ez a tábla tartalmazza a bevételi tranzakciókat.

**id:** a tranzakció azonosítója

**user\_id:** a felhasználó azonosítója

**date:** a tranzakció dátuma

**categories\_income\_id:** a bevételi kategória azonosítója

**subcategories\_income\_id:** a bevételi alkategória azonosítója

**amount:** a tranzakció összege

**currency\_id:** a pénznem azonosítója

**comment:** a tranzakció megjegyzése

**Létrehozása:**

CREATE TABLE `transactions\_income` (

`id` int(11) NOT NULL,

`user\_id` int(11) NOT NULL,

`date` datetime NOT NULL,

`categories\_income\_id` int(11) NOT NULL,

`subcategories\_income\_id` int(11) DEFAULT NULL,

`amount` double NOT NULL,

`currency\_id` int(11) NOT NULL,

`comment` text COLLATE utf8\_hungarian\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_hungarian\_ci;

ALTER TABLE `transactions\_income`

ADD PRIMARY KEY (`id`),

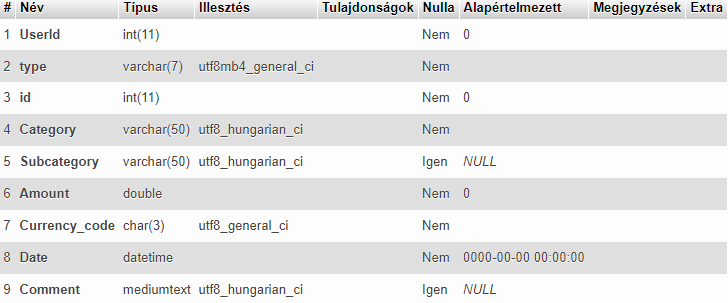
ADD KEY `FK\_transactions\_income` (`currency\_id`),

ADD KEY `FK\_transactions\_income\_users\_id` (`user\_id`),

ADD KEY `categories\_income\_id` (`categories\_income\_id`);

### A „view\_alltransactions” nézet

Ez a nézet tartalmazza a „transactions\_expense” és „transactions\_income” táblák unióját.



**UserId:** a felhasználó azonosítója

**type:** a tranzakció típusa

**id:** a tranzakció azonosítója

**Category:** a kategória neve

**Subcategory:** az alkategória neve

**Amount:** a tranzakció összege

**Currency\_code:** a pénznem kódja

**Date:** a tranzakció dátuma

**Comment:** a tranzakció megjegyzése

**Létrehozása:**

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `view\_alltransactions` AS

SELECT

`t\_i`.`user\_id` AS `UserId`,

'income' AS `type`,

`t\_i`.`id` AS `id`,

`c\_i`.`name` AS `Category`,

NULL AS `Subcategory`,

`t\_i`.`amount` AS `Amount`,

`c`.`currency\_code` AS `Currency\_code`,

`t\_i`.`date` AS `Date`,

`t\_i`.`comment` AS `Comment`

FROM(((`transactions\_income` `t\_i`

JOIN `categories\_income` `c\_i` ON (`t\_i`.`categories\_income\_id` = `c\_i`.`id`))

JOIN `currencies` `c` ON (`t\_i`.`currency\_id` = `c`.`id`))

JOIN `users` `u` ON (`t\_i`.`user\_id` = `u`.`id`))

UNION

SELECT

`t\_e`.`user\_id` AS `UserId`,

'expense' AS `type`,

`t\_e`.`id` AS `id`,

`c\_e`.`name` AS `Category`,

`s\_e`.`name` AS `Subcategory`,

`t\_e`.`amount` AS `Amount`,

`c`.`currency\_code` AS `Currency\_code`,

`t\_e`.`date` AS `Date`,

`t\_e`.`comment` AS `Comment`

FROM((((`transactions\_expense` `t\_e`

JOIN `categories\_expense` `c\_e` ON (`t\_e`.`categories\_expense\_id` = `c\_e`.`id`))

JOIN `subcategories\_expense` `s\_e` ON (`t\_e`.`subcategories\_expense\_id` = `s\_e`.`id`))

JOIN `currencies` `c` ON (`t\_e`.`currency\_id` = `c`.`id`))

JOIN `users` `u` ON (`t\_e`.`user\_id` = `u`.`id`));

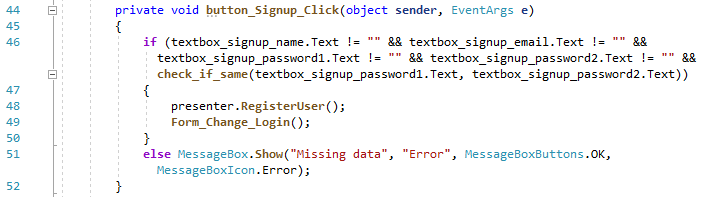
## Fontosabb függvények bemutatása

### Regisztráció

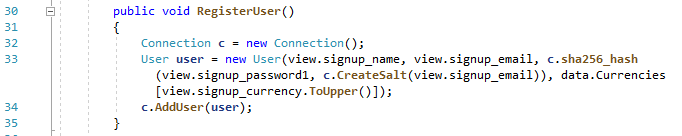
* Ellenőrzi, hogy:
  + az email cím megfelelő formátumú-e
  + az adott email elérhető-e
  + a jelszó megfelel-e a követelményeknek
  + a két jelszó megegyezik-e
* Amennyiben helyesen töltötte ki az adatokat, a regisztráció sikeresen végbemegy, az adatok elmentésre kerülnek az adatbázisban.

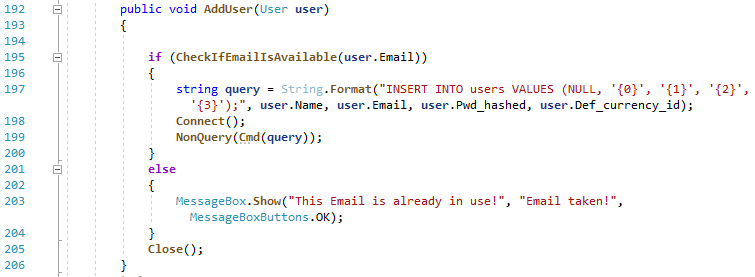
**C#:**

* **View => Signup => FormSignup.cs** 
  + **44.sor button\_Signup\_Click()**: Ellenőrzi a felhasználó által megadott adatokat és meghívja a presenter **RegisterUser()** metódusát.



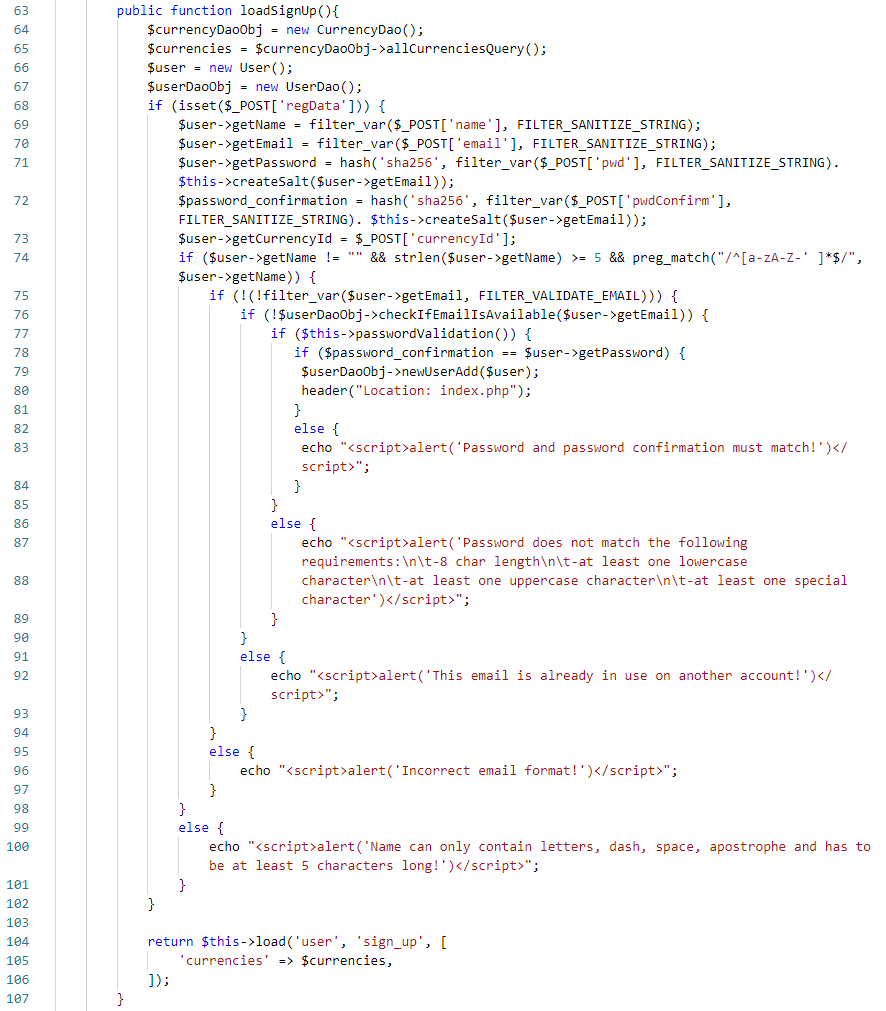
* **Presenter => SignupPresenter.cs**
  + **30.sor RegisterUser():** Rögzíti a felhasználó adatait egy User típusú object-be, majd meghívja a Connection class **AddUser()** metódusát.



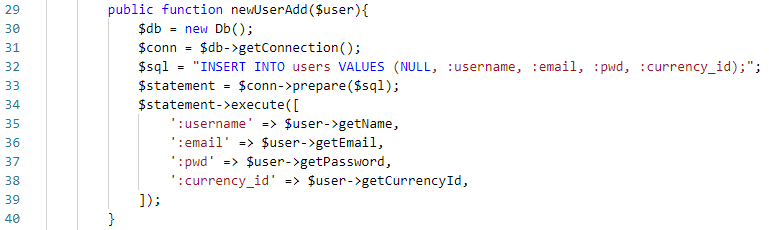
* **Model => Connection.cs**
  + **192.sor AddUser():** Ellenőrzi, az email cím elérhetőségét majd feltölti a felhasználó adatait az adatbázisba.

**PHP:**

* **app => controller => TransactionController.php**
  + **63.sor loadSignUp():** A felhasználó által megadott adatokat elmenti egy User object-be majd ellenőrzi, hogy a megadott adatok megfelelő formátumúak-e majd ellenőrzi, hogy az email elérhető-e. Amennyiben a felhasználó minden adatot helyesen adott meg, az adatok bekerülnek az adatbázisba.



* **app => model => UserDao.php**
  + **29.sor newUserAss():** A kapott User object adatait feltölti az adatbázisba.

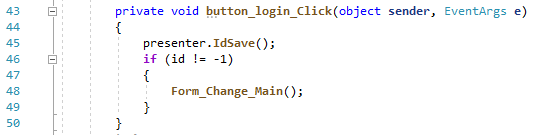


### Bejelentkezés

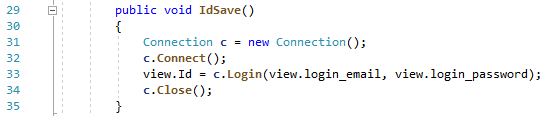
* Ellenőrzi, hogy létezik-e az email.
* Az adatbázisban tárolt hashelt jelszót összehasonlítja a megadott jelszóval.
* Amennyiben minden adat helyes, bejelentkezteti az alkalmazás.

**C#:**

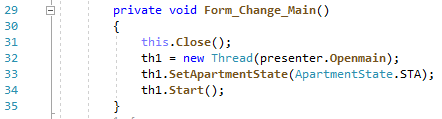
* **View => Login => FormLogin.cs** 
  + **43.sor button\_login\_Click()**: Meghívja a presenter **IdSave()** metódusát.



* **Presenter => LoginPresenter.cs**
  + **29.sor IdSave():** Meghívja a Connection class **Login()** metódusát a felhasználó által megadott bejelentkezési adatokkal majd a visszatérési értéket átadja a view Id tulajdonságának.

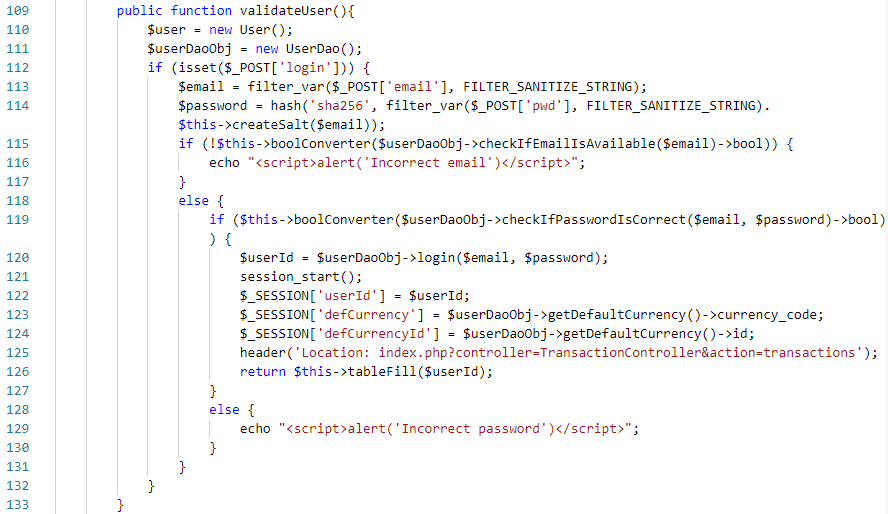


* **Model => Connection.cs**
  + **47.sor Login():** Hasheli a jelszót és lekéri az adott bejelentkezési adatokhoz tartozó felhasználói id-t.
* **View => Login => FormLogin.cs** 
  + **29.sor Form\_Change\_Main()**: Amennyiben a felhasználói id nem egyenlő -1-gyel a **FormLogin.cs 48.sorában** meghívásra kerül a **Form\_Change\_Main()** metódus, mely zárja a bejelentkezési formot és egy új Threaden példányosítja a Main formot.

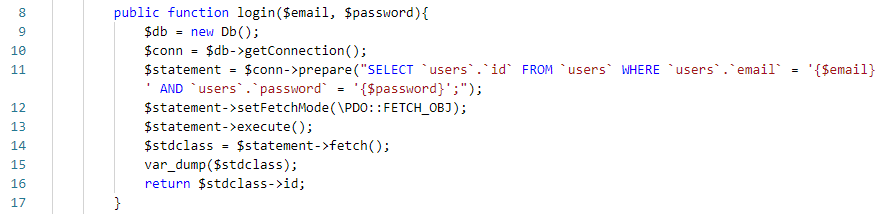


**PHP:**

* **app => controller => TransactionController.php**
  + **109.sor validateUser():** Hasheli a megadott jelszót majd ellenőrzi, hogy a megadott email létezik-e az adatbázisban. Amennyiben igen, összeveti a hashelt jelszót az adatbázisban tárolt jelszóval. Ha a két hashelt jelszó egyezik meghívja a **UserDao.login()** metódust, ezzel bejelentkezteti a felhasználót és a **TransactionController.tableFill()** metódust mellyel feltölti az oldalt a felhasználó adataival.



* **app => model => UserDao.php**
  + **8.sor login():** Lekéri a felhasználói adatokhoz tartozó felhasználói Id-t.

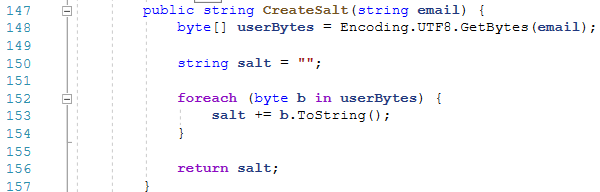


### „Salting” és „Hashing”

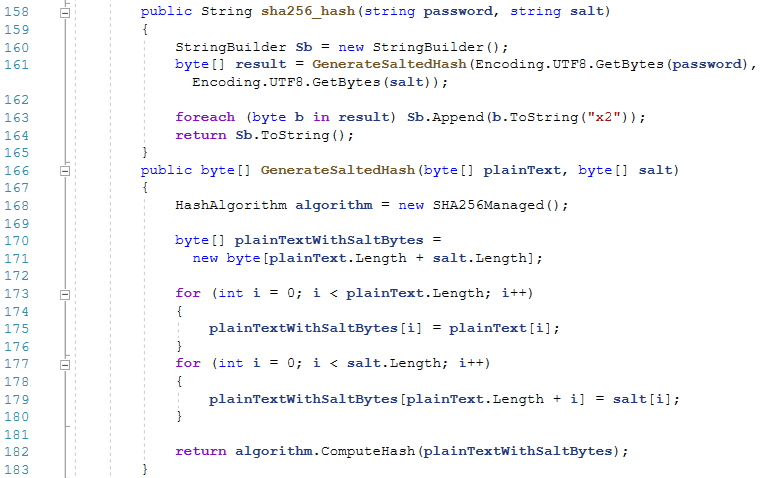
* sha256-os hashelést használva titkosítja a felhasználó jelszavát, az adatbázisban csak egy vissza nem fejthető hash kulcsot tárolunk el.
* A „salt”-ot a felhasználó emailcíme alapján állítjuk elő, mivel nincs két egyforma emailcím az adatbázisban így minden hash kulcs egyedi lesz, ezzel védelmet nyújtunk a „rainbow dictionary” támadások ellen.

**C#:**

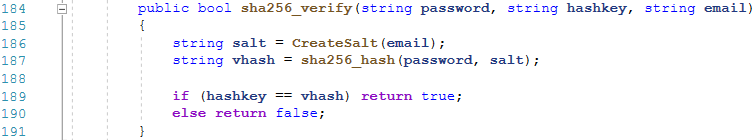
* **Model => Connection.cs**
  + **147.sor CreateSalt():** Bemenetként megkapja az email címet, amit byte halmazzá alakít. A byte halmaz minden elemét hozzáfűzi a salt stringhez. majd visszaadja az eredményt.



* + **158.sor sha256\_hash():** Meghívja aGenerateSaltedHash() metódust, mely összefűzi a jelszó és a salt byte halmazait, majd az eredményt sha256-os hasheléssel titkosítja.

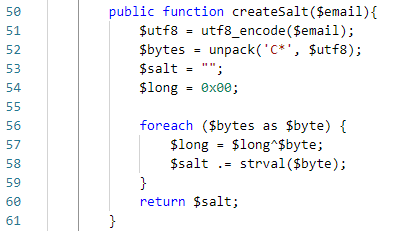


* + **184.sor sha256\_verify():** Megkapja a bejelentkezési jelszót és email-t, valamint a hashelt jelszót az adatbázisból. Hasheli a bejelentkezési adatokat, a hash eredményét összeveti az adatbázisban tárolt hashelt jelszóval.

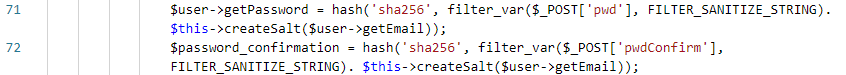


**PHP:**

* **app => controller => TransactionController.php**
  + **50.sor createSalt():** Bemenetként megkapja az email címet, amit byte halmazzá alakít. A byte halmaz minden elemét hozzáfűzi a salt stringhez. majd visszaadja az eredményt.



* + **71-72.sor loadSignUp():** Hasheli a megadott jelszót és jelszó megerősítést.



* + **114.sor validateUser():** Hasheli a megadott jelszót, ez a hashelt jelszó kerül összehasonlításra az adatbázisban tárolt hashelt jelszóval.



### Szűrés

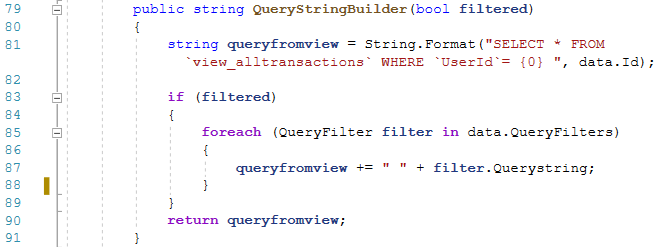
* Az aktív szűrőket hozzáfűzi az SQL parancshoz.

**C#:**

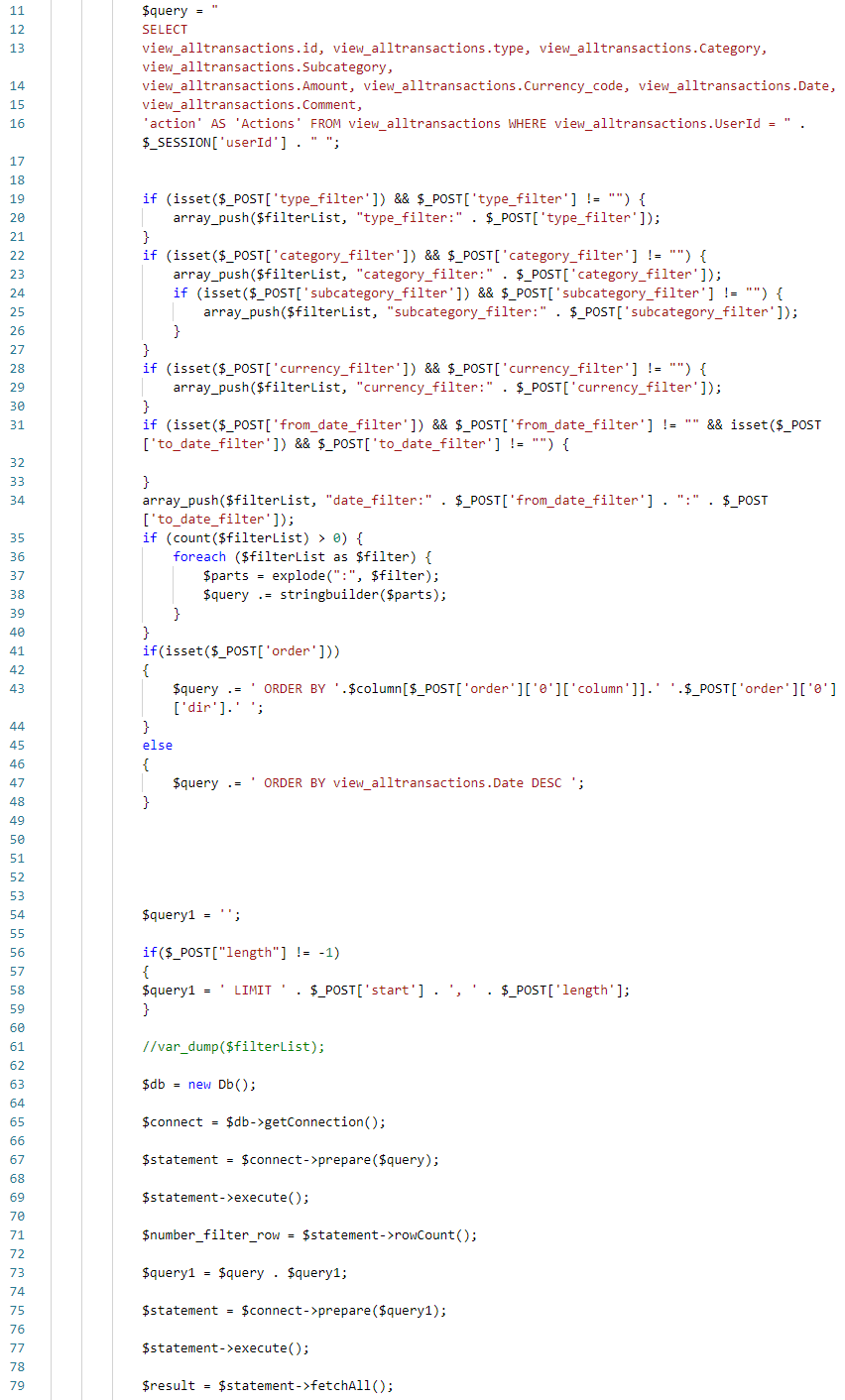
* **Presenter => MainPresenter.cs**
  + **109.sor DeleteCreateAddFilter():** Megkapja a szűrő bemeneti adatait, melyek alapján létrehoz egy filter objectet majd hozzáadja a szűrő listához.

****

* + **79.sor QueryStringBuilder():** Megkapja, hogy van-e aktív szűrő, amennyiben van aktív szűrő a metódus végig lépteti a szűrő listát és minden szűrő query stringjét hozzáfűzi a lekérdezéshez, majd visszaadja a teljes lekérdezés stringet.



**PHP:**

* **filter.php**
  + **11.sor:** A filter.php feldolgozza az aktív szűrőket. Amennyiben egy szűrő benne van a POST szuperglobális tömbben, hozzáadásra kerül az sql stringhez.

### Pénznem átváltás

* Átváltja a tranzakció pénznemét a felhasználó alapértelmezett pénznemére

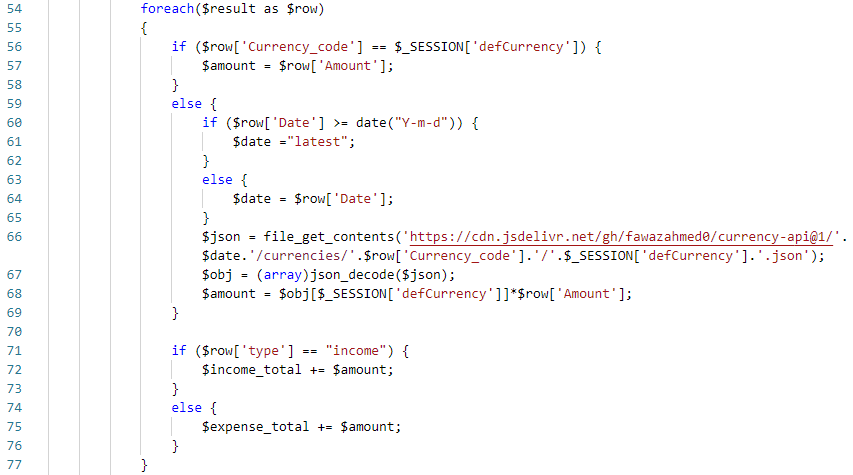
**C#:**

* **Model => Exchangerate.cs**
  + **11.sor CurrencyExchangeRate():** Bemeneti értékként megkapja az alap és cél pénznem kódját, valamint az átváltás dátumát. A megkapott adatokat beilleszti az url-be majd lekéri az api JSON válaszát, amennyiben a válasz nem megfelelő formátumú a kapott dátum helyett a „latest” kulcsszót használva újrapróbálkozik. A kapott JSON választ Dictionary-vé alakítja a **JavaScriptSerialiser.Deserialise()** metódus segítségével. A kapott Dictionary-ből kiolvassa és visszaadja az átváltási értéket.



**PHP:**

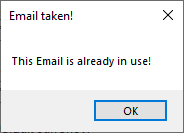
* **balance.php**
  + **54.sor:** A Ballance.php soronként feldolgozza az lekérdezett tranzakciós adatokat, amelyik sorban nem az alapértelmezett pénznem van megadva, ott meghívja a currency-api-t a sor adataival. Amennyiben a tranzakcióhoz tartozó dátum tulajdonság a mai dátummal megegyezik vagy későbbi, a dátum helyett a „latest” kulcsszót használjuk.

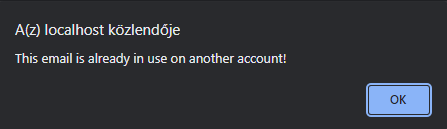


## Tesztelés

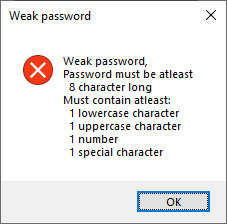
### Regisztráció

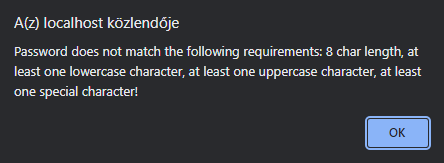
* Ha a felhasználó által meadott email cím foglalt.

**Asztali:**

**Web:**

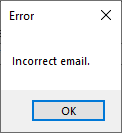
* Ha a jelszó nem megfelelő erősségű.

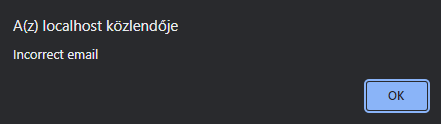
**Asztali:**

**Web:**

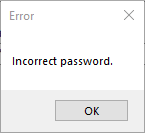
### Bejelentkezés

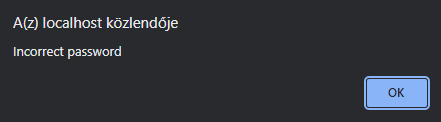
* Amennyiben a felhasználó nem létező email címet ad meg.

**Asztali:**

**Web:**

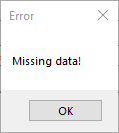
* Amennyiben a felhasználó nem megfelelő jelszót ad meg.

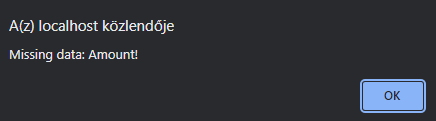
**Asztali:**

**Web:**

### Tranzakció hozzáadása

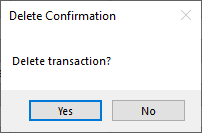
* Amennyiben a felhasználó nem adja meg az összes adatot.

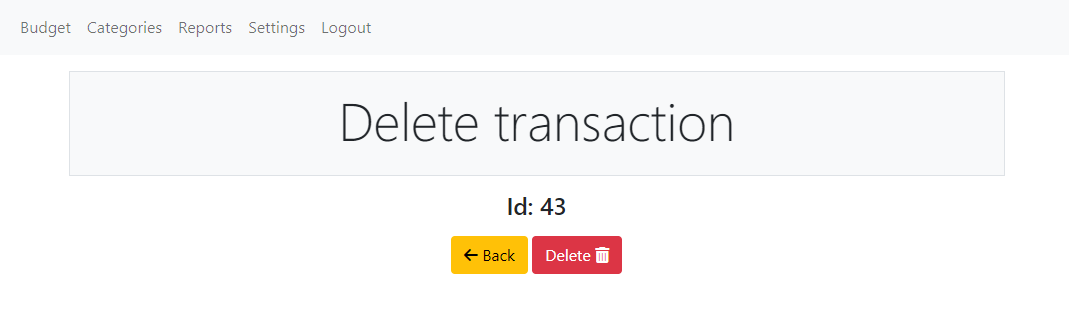
**Asztali:**

**Web:**

### Tranzakció törlése

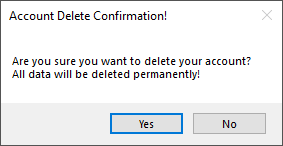
* Tranzakció törlés megerősítése.

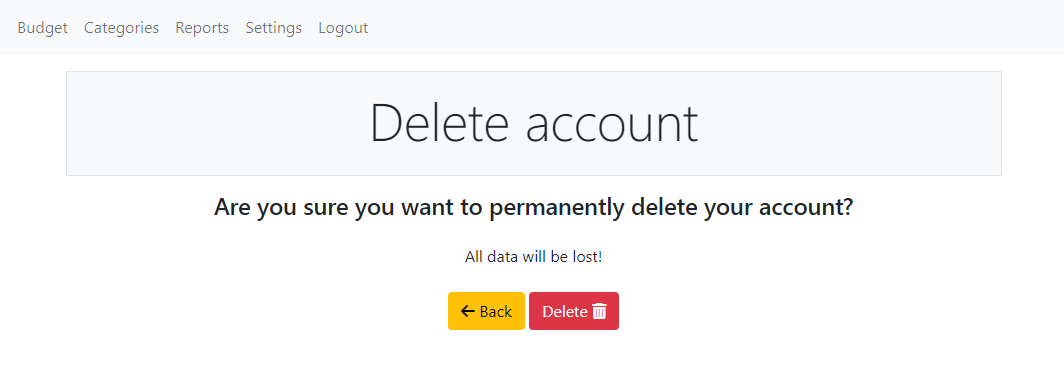
**Asztali:**

**Web:**

### Felhasználó törlése

* Felhasználó törlésének megerősítése.

**Asztali:**

**Web:**

## Fejlesztési lehetőségek

* „subcategories\_income” tábla hozzáadása.
* Költségvetési kategóriák szerinti tervezett/aktuális költség rendszer kialakítása.
* Havi, féléves, éves ismétlődő tranzakciók ütemezése.

# **Felhasználói dokumentáció**

## Program leírása

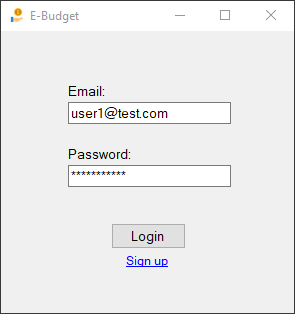
Az általunk készített alkalmazás, felhasználók költségeinek napról napra való követésére szolgál. Az alkalmazás szűrési és kategóriakezelési funkciói által segíti a tranzakciók lekérdezését és rendszerezését.

## Bejelentkezési és Regisztrációs felület

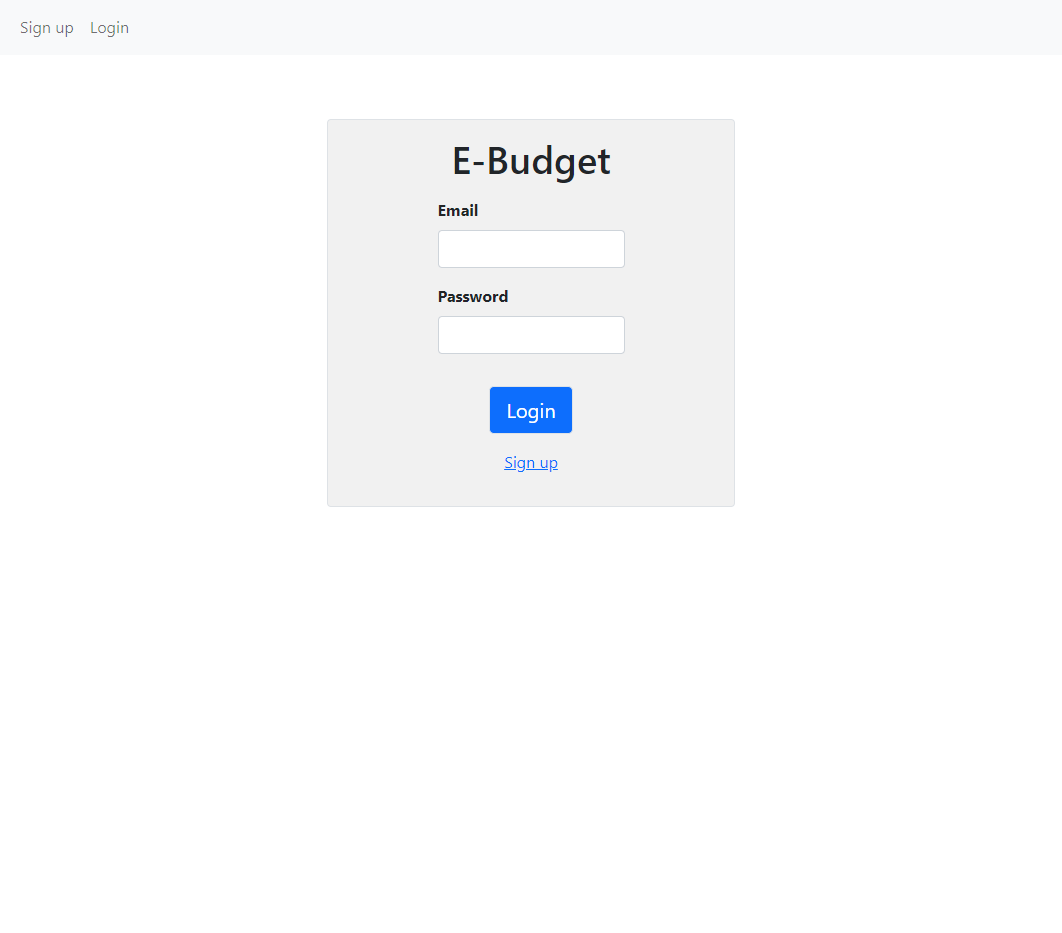
A még nem regisztrált és be nem jelentkezett felhasználók számára csak a bejelentkezés és regisztrációs felületek érhetőek el.

### Bejelentkezés

* **Sign up:** A gombra kattintva a program átirányítja a felhasználót a regisztrációs felületre.
* **Login:** A gombra kattintva a program bejelentkezteti a felhasználót, amennyiben minden adatot helyesen adott meg.

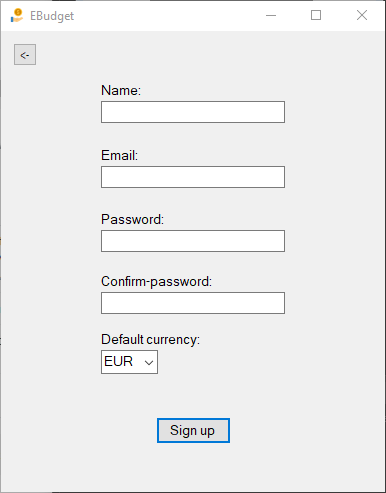
**Asztali:**

**Web:**

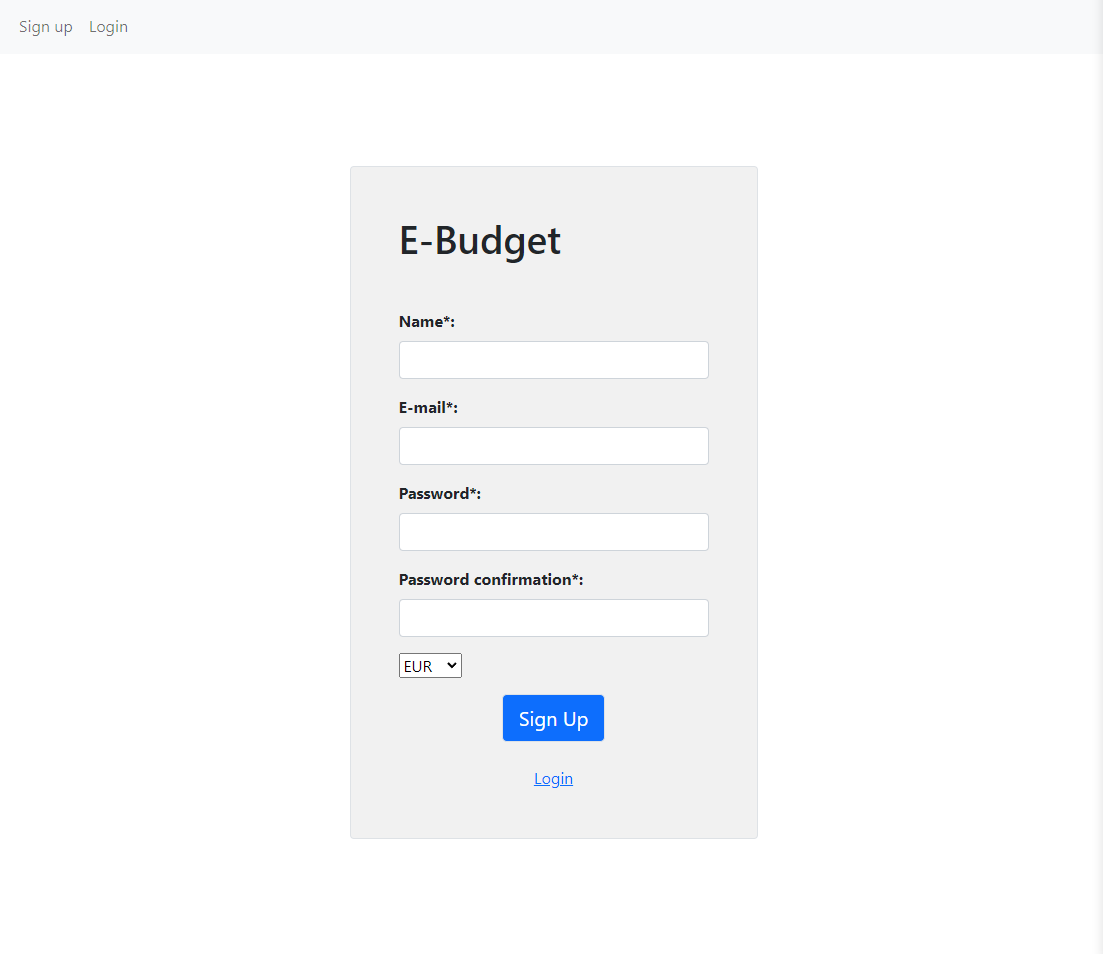


### Regisztráció

* **Visszanyíl/Login:** A gombra kattintva a program átirányítja a felhasználót a bejelentkezési felületre.
* **Sign up:** A gombra kattintva a program regisztrálja a felhasználót, amennyiben minden adatot helyesen adott meg.

**Asztali:**

**Web:**



## Regisztrált és bejelentkezett felhasználói felület

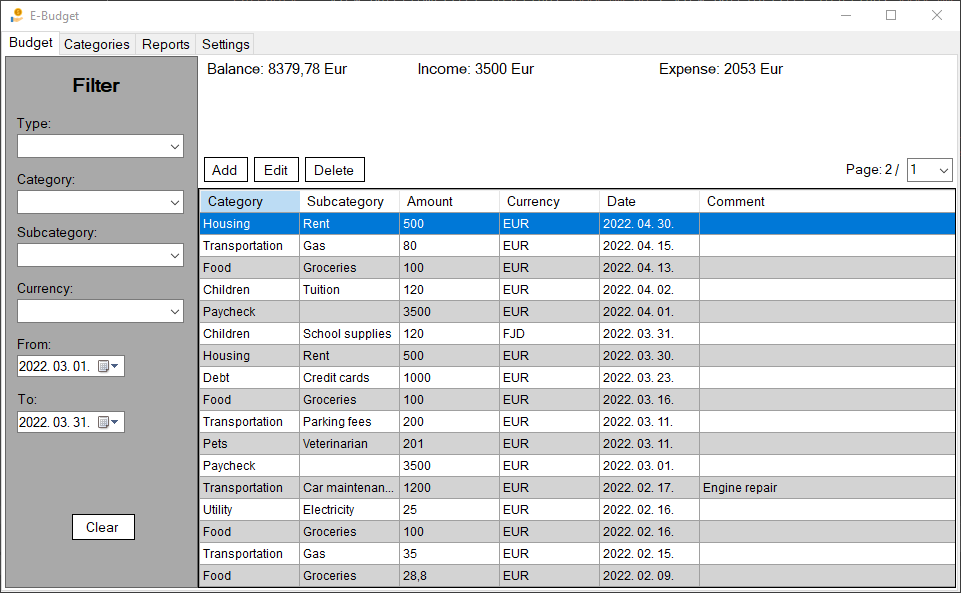
A bejelentkezett felhasználói felület.

### „Budget” felület

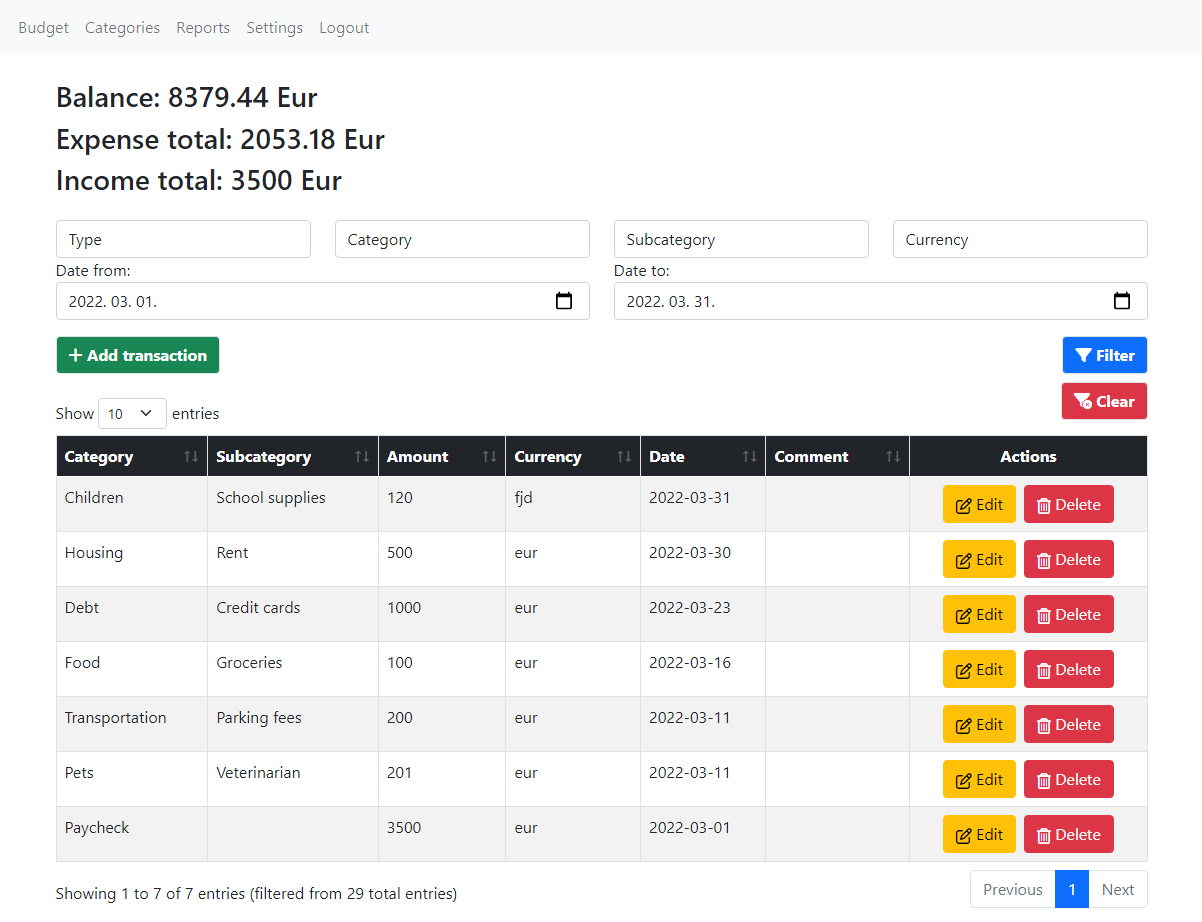
A felhasználó ezen az oldalon kezelheti a tranzakcióit.

* **Add/Add transaction:** Megnyitja az „Add Transaction” felugró ablakot.
* **Edit:** Megnyitja az „Add Transaction” felugró ablakot a kiválasztott sor adataival betöltve.
* **Delete:** Törli az adott sort egy felugró ablakos megerősítés után.
* **Clear:** Törli a szűrési paramétereket.
* **Filter:** Szűri a tranzakciókat az aktív szűrők alapján.

**Asztali:**



**Web:**

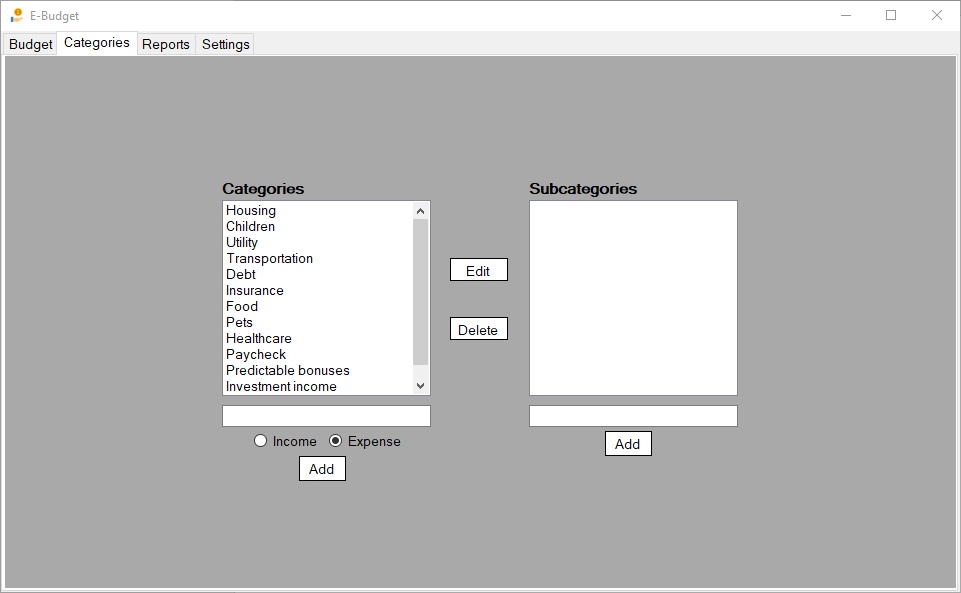


### „Categories” felület

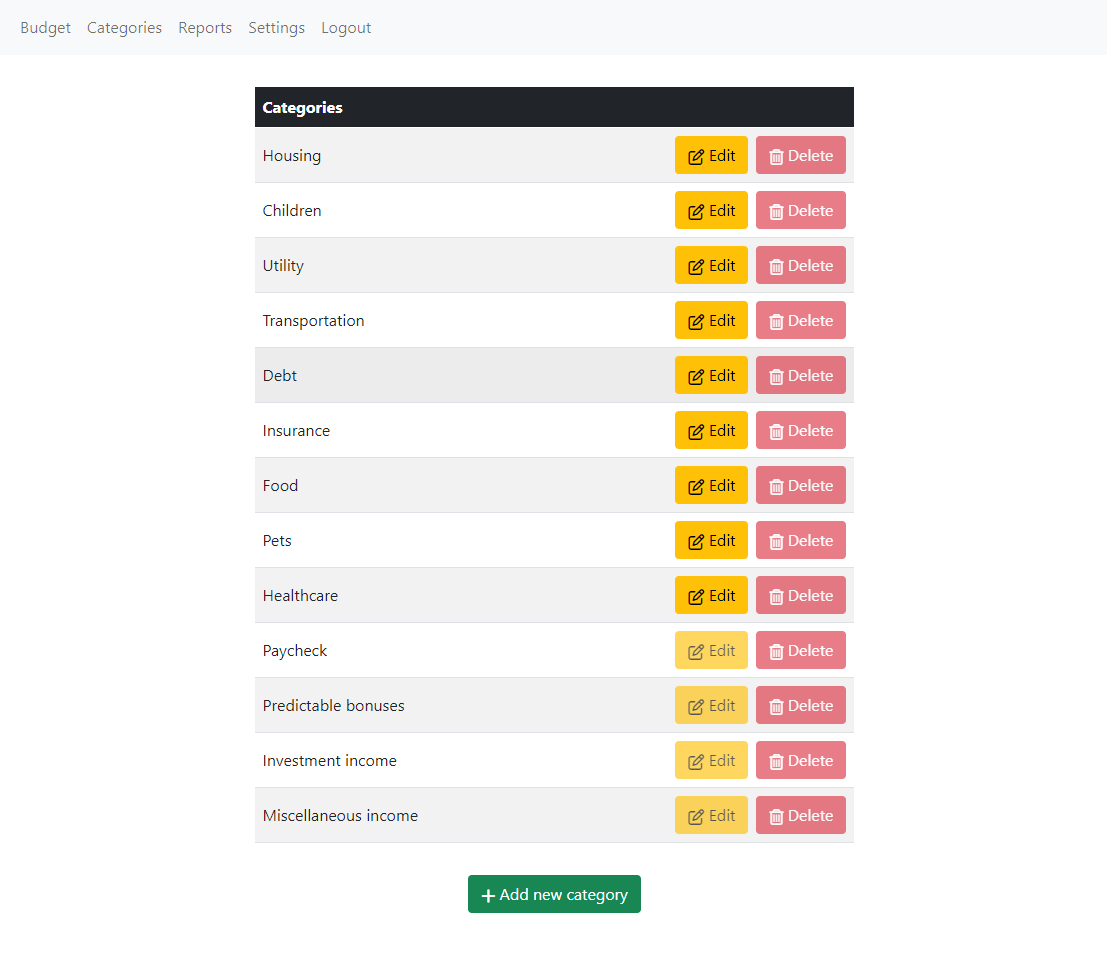
A felhasználó ezen az oldalon kezelheti a kategóriákat.

* **Add:** Hozzáadja a Kategóriát/Alkategóriát.
* **Edit:** Módosítja a kijelölt Kategóriát/Alkategóriát.
* **Delete:** Törli a kijelölt Kategóriát/Alkategóriát.

**Asztali:**



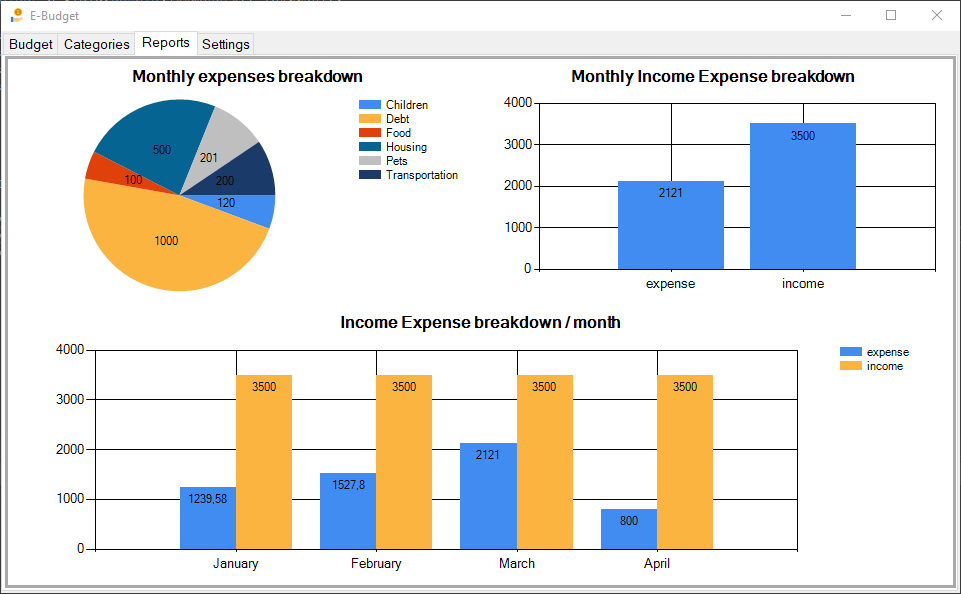
**Web:**

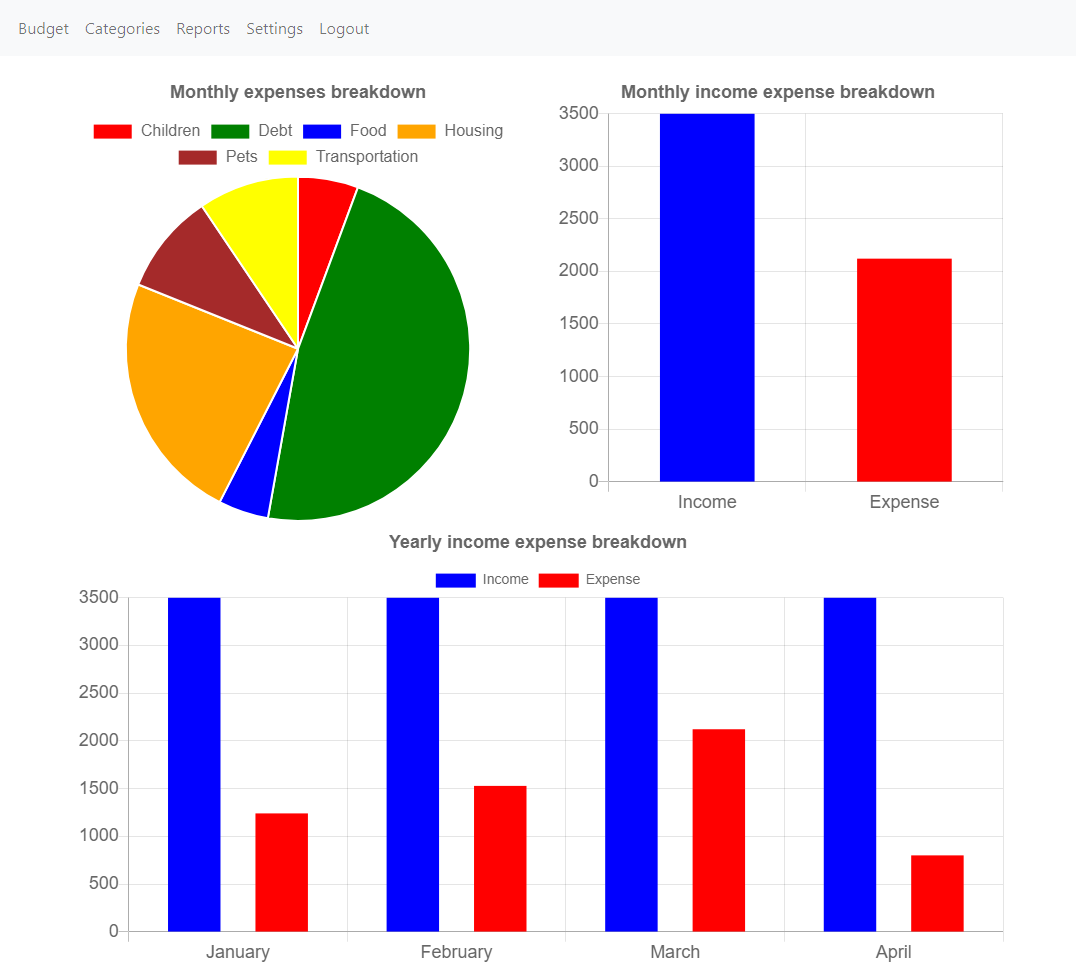


### „Reports” felület

A felhasználó ezen az oldalon tekintheti meg a kimutatásokat.

**Asztali:**



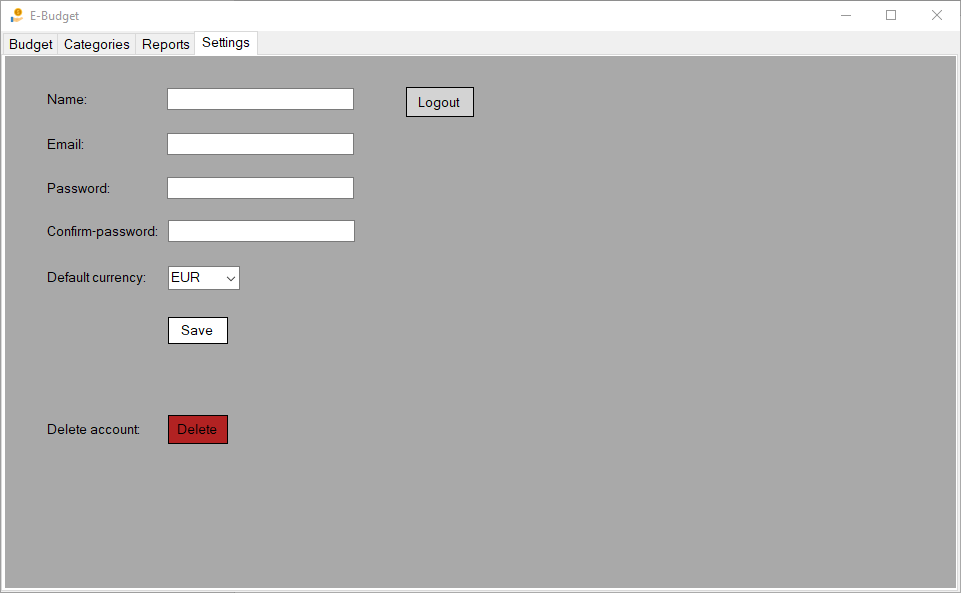
**Web:**

### „Settings” felület

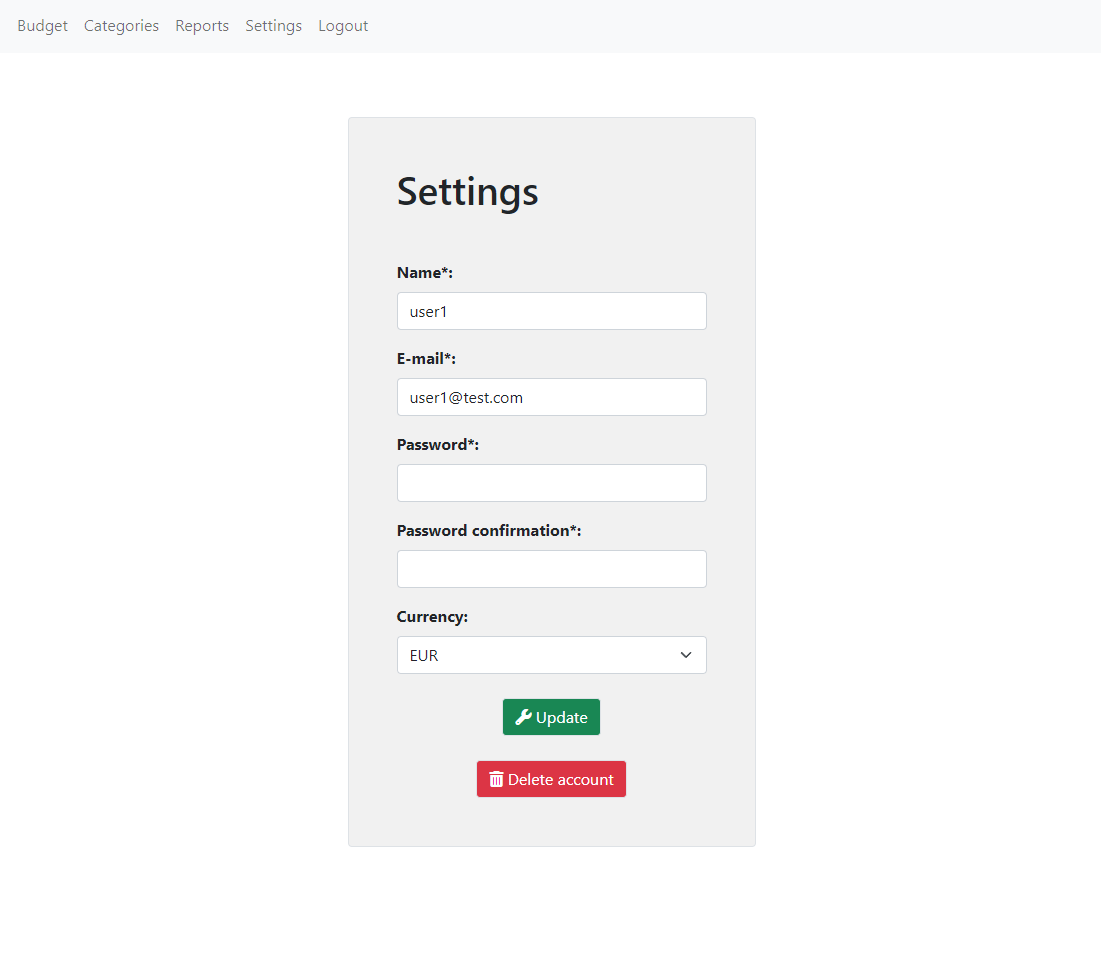
A felhasználó ezen az oldalon kezelheti az adatokat.

* **Save/Update:** Menti a felhasználó adatait.
* **Logout:** Kijelentkezteti a felhasználót.
* **Delete:** Törli a felhasználó minden adatát az adatbázisból.

**Asztali:**



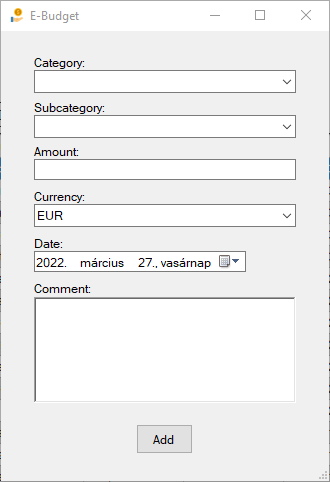
**Web:**

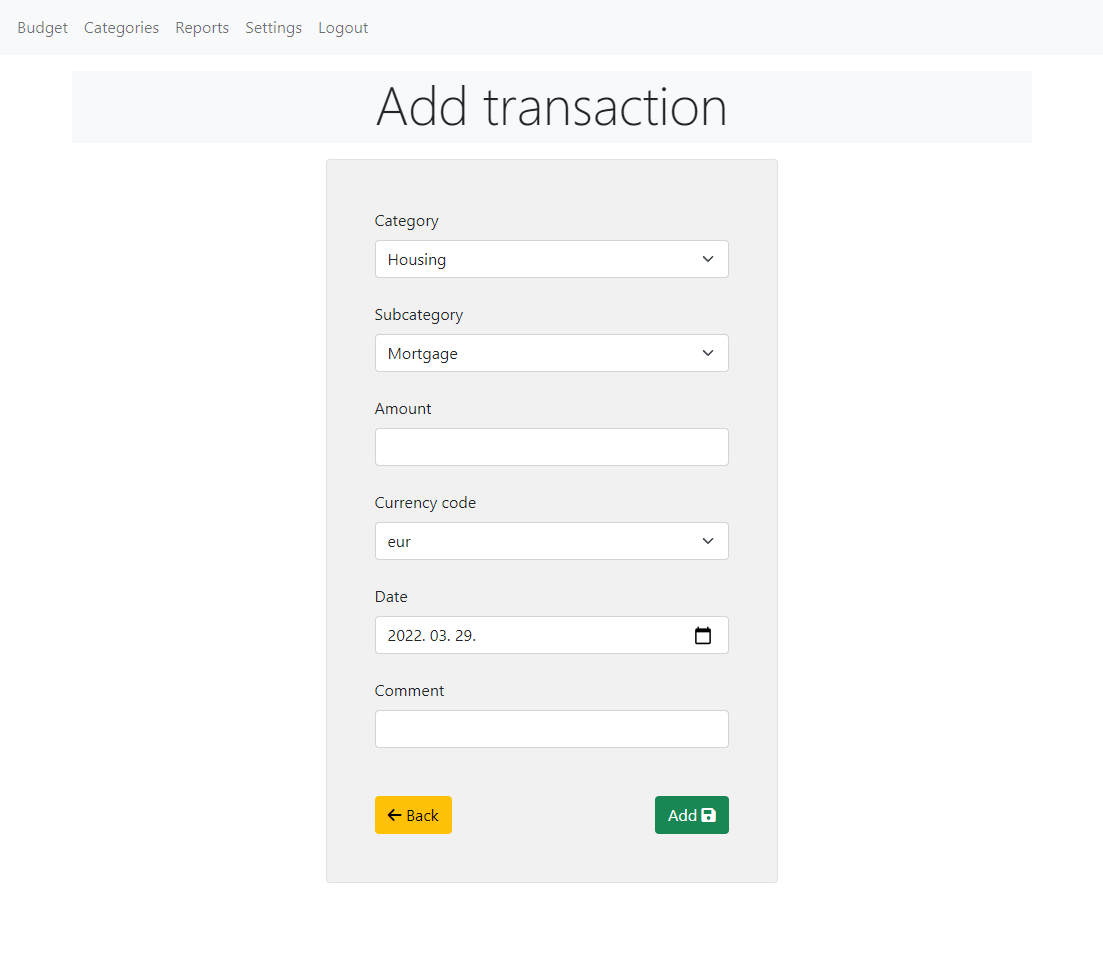


### „Add transaction” felület

A felhasználó ezen a felületen tud új tranzakciót hozzáadni.

**Asztali:**

****

**Web:**

# **Irodalomjegyzék**

**Online források:**

* https://hu.wikipedia.org/wiki/MySQL
* https://hu.wikipedia.org/wiki/SQL
* https://hu.wikipedia.org/wiki/.NET\_keretrendszer
* https://hu.wikipedia.org/wiki/C\_Sharp
* https://hu.wikipedia.org/wiki/PHP
* https://hu.wikipedia.org/wiki/Git
* https://hu.wikipedia.org/wiki/GitHub
* https://hu.wikipedia.org/wiki/Microsoft\_Visual\_Studio
* https://hu.wikipedia.org/wiki/Visual\_Studio\_Code
* https://github.com/fawazahmed0/currency-api#readme
* https://www.webslesson.info/2018/09/add-server-side-datatables-custom-filter-using-php-with-ajax.html
* https://datatables.net/
* https://www.aspsnippets.com/Articles/Create-Pie-Chart-in-Windows-Forms-Application-using-C-and-VBNet.aspx

# **Mellékletek**

Név Formátum Tartalom

Database könyvtár Adatbázis

Documentation könyvtár Dokumentáció elemek

PHP könyvtár Webes felület

Winform könyvtár Asztali felület

c67b00adc43….ico .ico Alkalmazás ikon

testuser.txt .txt Teszt felhasználó adatai

Dokumentáció.pdf .pdf Dokumentáció