

BMSZC Bláthy Ottó Titusz Informatikai Technikum

1032. Budapest, Bécsi út 134.

Szak dolgozat

Pénzügykezelő alkalmazás

| | |
|---|----|
| Bevezetés..... | 4 |
| Felhasználói dokumentáció | 5 |
| A program általános specifikációja..... | 5 |
| Rendszer követelmények..... | 7 |
| Hardver követelmények..... | 7 |
| Szoftver követelmények | 7 |
| Alkalmazás telepítése | 8 |
| A program használatának részletes leírása | 19 |
| Alkalmazás futtatása | 19 |
| Navigációs sáv..... | 20 |
| A “Home” (Kezdő) oldal..... | 21 |
| A “Register” (regisztrációs) aloldal | 22 |
| A “Login” (bejelentkezés) aloldal..... | 23 |
| A “Forgot password” (elfelejtett jelszó) aloldal..... | 24 |
| A “Reset password” (jelszó visszaállítása) aloldal | 24 |
| A “Restore deactivated account” (felhasználói fiók visszaállítása) aloldal | 25 |
| Az “About us” (Rólunk) aloldal | 26 |
| A "Home" főoldal | 26 |
| Az “Incomes” (Bevételek) aloldal | 27 |
| Az “Add Income” (Bevételek hozzáadása) aloldal..... | 27 |
| A “Spending” (Kiadások) aloldal..... | 27 |
| Az “Add Spending” (Kiadások hozzáadása) aloldal | 28 |
| A “Setting” (Beállítások) aloldal | 28 |
| Az “Advanced Statistics” (Haladó Statisztikák) aloldal..... | 28 |
| A “Documentation” (Dokumentáció) aloldal | 29 |
| Fejlesztői dokumentáció..... | 30 |
| Az alkalmazott fejlesztői eszközök..... | 30 |
| Adatmodell leírása (Adatbázis) | 30 |
| Részletes feladat-specifikáció, algoritmusok | 38 |
| Controllerek működése..... | 38 |
| HomeController működése: | 38 |
| Kiegészítő függvények: | 38 |
| FinanceController működése: | 46 |
| Store függvény:..... | 47 |
| IncomeController működése:..... | 47 |
| Kiegészítő függvények: | 47 |
| RestoreAccountController működése: | 50 |
| Függvények: | 50 |
| DocumentationController működése:..... | 51 |
| LoginController működése: | 52 |
| FamilyController működése: | 53 |

| | |
|--|----|
| RegisterController működése: | 54 |
| Kiegészítő függvények: | 54 |
| ForgotPasswordController működése: | 59 |
| AboutUsController működése: | 61 |
| MonthlyController működése: | 61 |
| SettingsController működése: | 62 |
| AdvancedStatisticsController működése: | 66 |
| Views (Nézetek) funkciói | 68 |
| register.blade.php | 69 |
| advancedStatistics.blade.php | 73 |
| Documentation.blade.php | 76 |
| Income.blade.php és spending.blade.php | 77 |
| incomeCreate.blade.php és spendingCreate.blade.php | 78 |
| settings.blade.php | 79 |
| Tesztelési dokumentáció | 80 |
| Összefoglalás | 81 |
| Továbbfejlesztési lehetőségek | 81 |
| Felhasznált irodalom (források) | 81 |
| Konzulens tanár: | 83 |
| Készítette: | 83 |

Bevezetés

A mi kis projektünk egy pénzügyek kezelésére épülő webes alkalmazás. Azért esett erre az ötletre a választásunk, mert szerettünk volna egy olyan projektet létrehozni, amit tudunk használni a jövőben. Amit később egy szerveren futtathatunk, illetve tovább tudjuk fejleszteni amikor szeretnénk. A témaválasztásunk főleg ezen aspektusok mentén jött létre. Másfelől mindig érdekelt minket a pénzügy téma. Ezen felül arra is vágytunk, hogy hozzáadhassunk egy olyan funkciót, amelyet ezelőtt még sehol nem láttunk, ugyanis a pénzügyek kezelésére számos alkalmazás megtalálható, ugyanakkor egyiknél se láttunk olyat, ahol lehetséges lett volna az egész család pénzügyeit kezelni. Meglátásunk szerint azért is lett egyedi és különleges a mi projectünk. Úgy gondoljuk, hogy ezen kis projekt, illetve ezen alkalmazás a jelen állapotban és a továbbfejlesztett változata is képes lehet egy alapot nyújtani nekünk, illetve másoknak a pénzügyeik kezelésében, illetőleg a pénzügyi céljaik megvalósításában. Úgy gondoljuk, hogyha az emberek képesek vezetni kiadásaikat akár csak néhány hónap erejéig, már úgy is jelentős költségcsökkentést lehetséges elérni.

Felhasználói dokumentáció

A program általános specifikációja

A programunk egy modern webes alkalmazás, amelyet kifejezetten a pénzügyek hatékony kezelésére terveztünk. A dokumentáció célja, hogy részletesen bemutassa az alkalmazásunk által kínált szolgáltatásokat és funkciókat, hogy Ön könnyedén el tudja dönteni, hogy megfelel-e az Ön szükségleteinek és elvárásainak.

Funkciók és Jellemzők

Pénzügyi Naplózás: Az alkalmazásunk lehetővé teszi Önnek, hogy precízen naplózza és nyomon kövesse mindennapi pénzügyeit. Egyszerűen rögzítheti bevételeit és kiadásait, ami segít Önnek abban, hogy átlátható képet kapjon a pénzügyi helyzetéről.

Grafikonok és Diagramok: A vizuális elemek rendkívül fontosak a pénzügyi helyzet megértésében. Az alkalmazásunk főoldalán különféle grafikonok és diagramok segítik Önt abban, hogy áttekinthesse havi kiadásait és bevételeit. Emellett egy további diagram segíti a családtagok részesedésének megértését a kiadások és bevételek terén.

Naptári Megjelenítés: A pénzügyek időzítése és tervezése rendkívül fontos. Az alkalmazásunk egy intuitív naptárnézetet kínál, amelyen könnyen áttekintheti a tranzakciók dátumait. Ez segíthet Önnek abban, hogy hatékonyabban tervezze és kezelje pénzügyeit.

Összesítő Kördiagramok: Az "Spending" és "Income" oldalakon található összesítő kördiagramok segítenek Önnek abban, hogy áttekintést kapjon az összes kiadásról és bevételekről. Ez segíthet Önnek abban, hogy hatékonyabban kezelje költségvetését és pénzügyeit.

Haladó Statisztikák: Az "Advanced Statistics" oldalon további részletes statisztikákat találhat, amelyek segítenek Önnek jobban megérteni és elemző módon kezelni a pénzügyeit. Ez lehetővé teszi Önnek, hogy mélyebb betekintést nyerjen pénzügyi helyzetébe és hatékonyabban tervezze meg jövőbeli lépéseit.

Reméljük, hogy ez a részletes áttekintés segített abban, hogy teljeskörűen megértse programunkat és annak kínált szolgáltatásait. Ha bármilyen további kérdése van, kérjük,

ne habozzon felvenni velünk a kapcsolatot. Örömmel segítünk további információkkal és támogatással.

Rendszer követelmények

Hardver követelmények

Minimum követelmények:

Operációs rendszer: Windows 10 vagy 11 64 bites operációs rendszer.

Processzor: Intel Core i5 14600.

Memória: 8Gb.

Ajánlott követelmények:

Operációs rendszer: Windows 10 vagy 11 64 bites operációs rendszer.

Processzor: Intel Core i5 14600 vagy nagyobb teljesítményű.

Memória: 16Gb

Szoftver követelmények

Az alkalmazásunk a következő operációsrendszerre(ekre) lett tervezve:

1. Windows 10 vagy 11 64 bites operációs rendszer.

Az alkalmazásunk működéséhez szükséges szoftverek:

1. Bármilyen böngésző, az alkalmazásunk Google Chrome, Microsoft Edge és Vivaldi böngészőben lett tesztelve így ezek az ajánlottak.
2. Visual Studio Code
3. Xampp
4. Composer
5. Node.js

Alkalmazás telepítése

(Bizonyos parancsok és funkciók működését akadályozhatja VPN illetve vírusirtó)

1. Node js letöltése és telepítése. Forrás: <https://nodejs.org/en>
2. Xampp letöltése és telepítése. A xampp magával hozza a php fájljait ezért nem kell külön letölteni a php-t. Forrás: <https://www.apachefriends.org/hu/index.html>
3. Composer letöltése és telepítése. Forrás: <https://getcomposer.org>
4. Telepítse a git-et

(Git install tutorial) <https://www.youtube.com/watch?v=4xqVv2lTo40>)

5. Parancssorban tesztelni, hogy a php és a node js verzióit a windows megtalálja és felismeri-e.
 - `php -v`
 - `node -v`

Amennyiben, nincs telepítve, sérültek a fájlok a telepítés közben vagy nincs helyes elérésiút állítva.

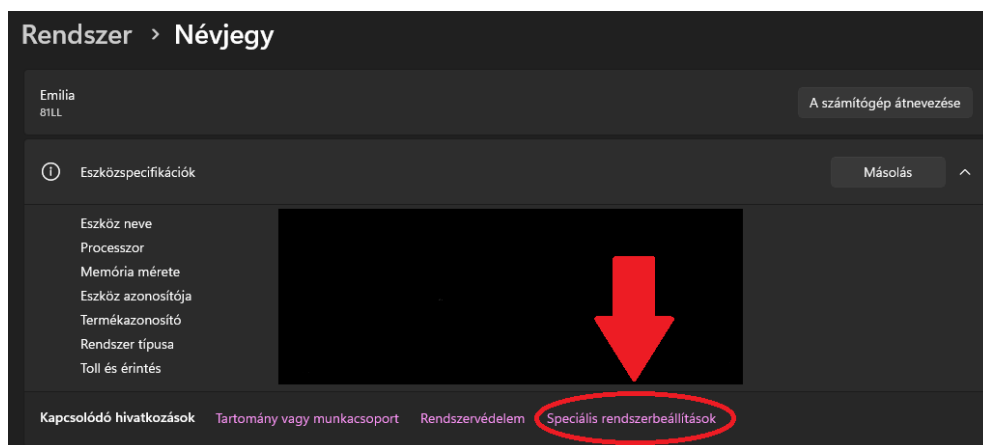
Ezt a következőképpen kell csinálni:

Nyomja meg a “Windows gomb” + “X” gomb kombinációját.

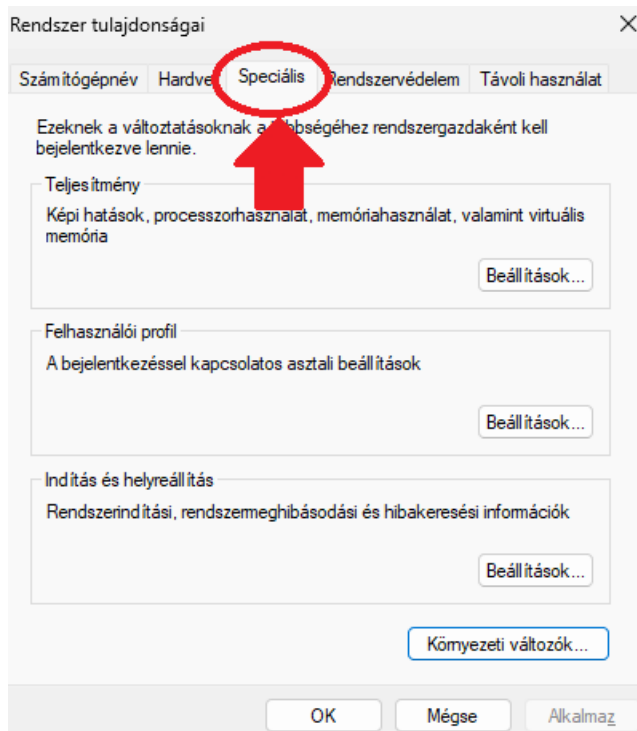
Válassza ki a “Rendszer” menüpontot.



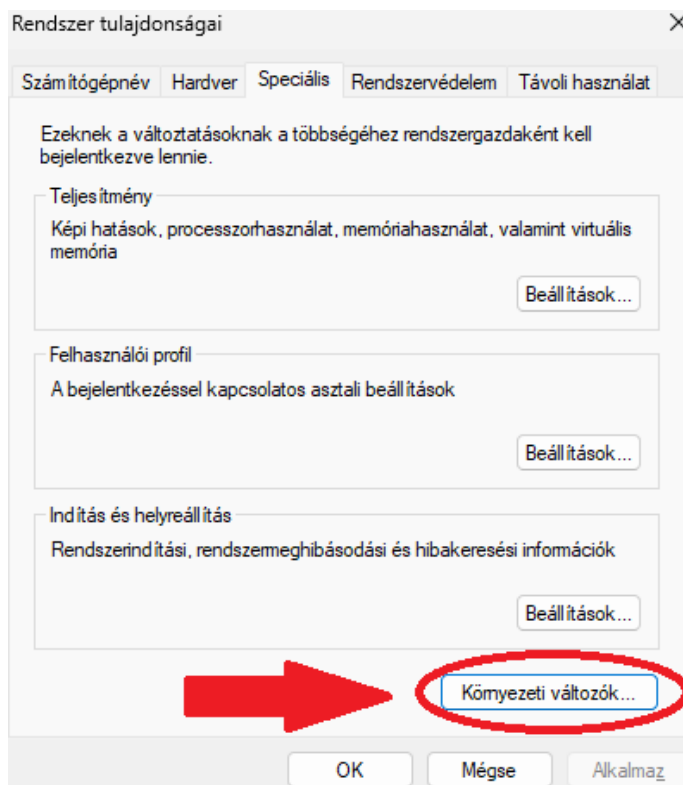
Válassza ki a “Speciális rendszer beállítások” menüpontot.



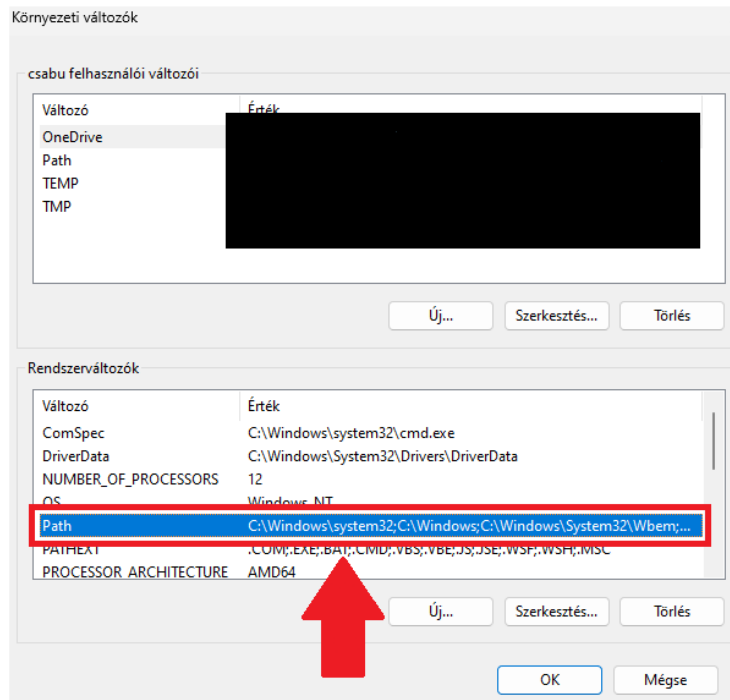
Válassza ki a “Speciális” menüpontot.



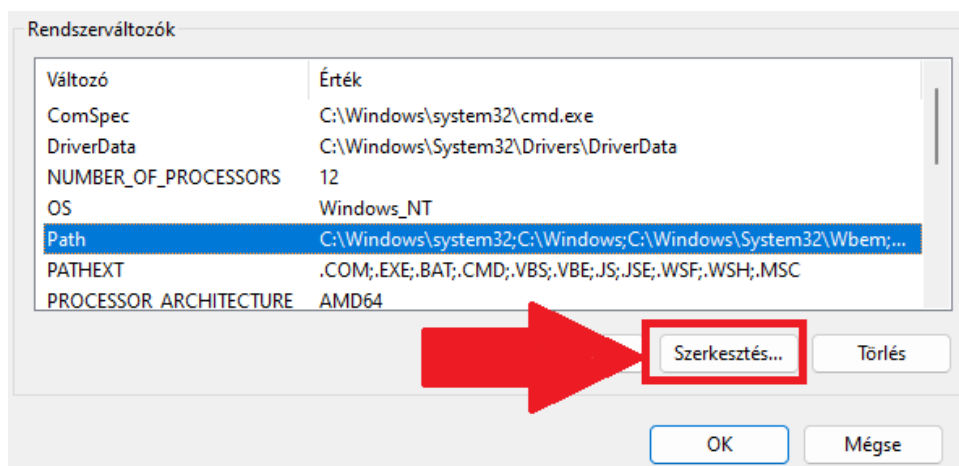
Válassza ki a “Környezeti változók” menüpontot.



Válassza ki a “Rendszerváltozók” menüből a “Path” menüpontot.



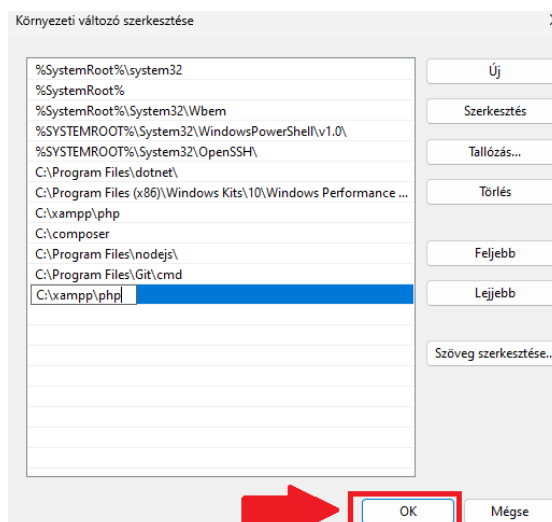
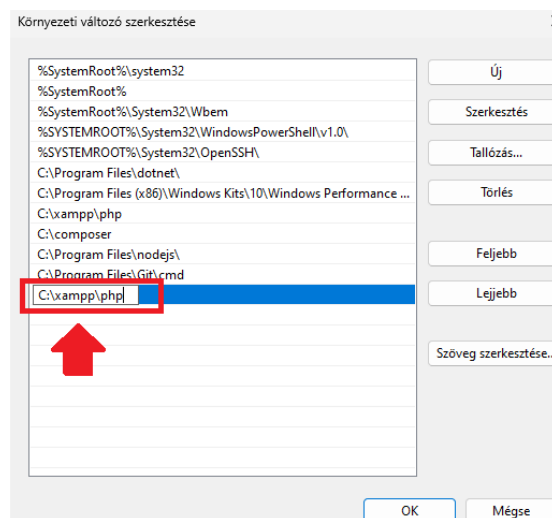
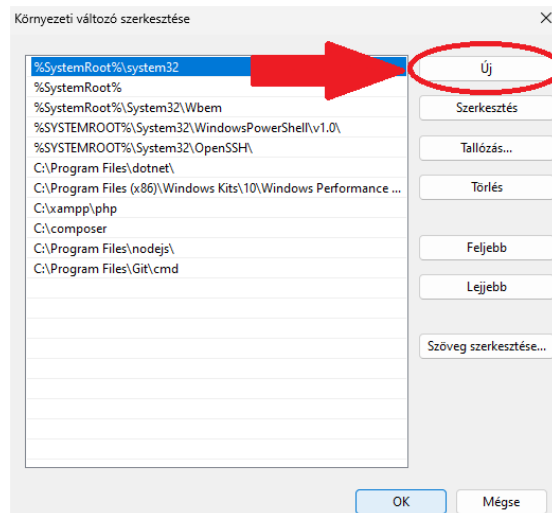
Nyomjon rá a “szerkesztés” gombot.



Hozzon létre új elérési utat az “Új” gombbal, írja be a xampp elérési útját.

Xampp esetében: C:/xampp/php (a telepítés helye alapján eltérő lehet).

Node js esetében: C:\Program Files\nodejs\

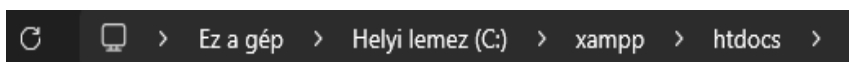
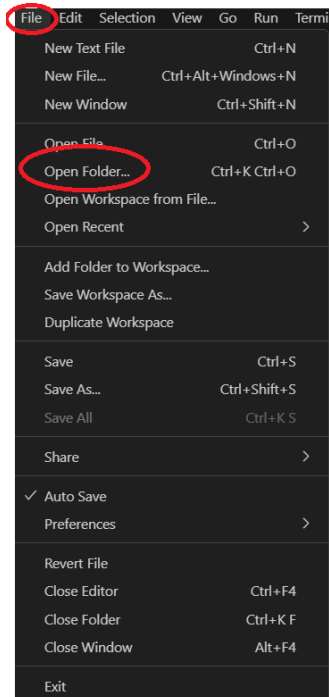


6. Visual studio code letöltése majd telepítése. Forrás:

<https://code.visualstudio.com>

7. A projekt letöltése és telepítése github-ról.

- Nyissa meg a “Visual Studio Code” nevű alkalmazást.
- A felső menüsorban nyomjon rá a “File” gombra, majd nyomjon rá az ”Open folder” gombra, válassza ki a “xampp” mappán belüli “htdocs” nevű mappát. (Ez a gép -> Helyilemez (C:, vagy ahova a xampp telepítve lett) -> xammpp -> htdocs)



- Nyisson egy új terminált a “Ctrl+Shift+ö” billentyű kombinációval vagy a felső menüsorban lévő “Terminal” gomra kattintva majd a “New terminal” gombra nyomva.

Két féle képpen tudja letölteni a projektünket:

(Lehetséges hogy 7zip vagy más csomagkezelő alkalmazásra van hozzá szükség:

<https://www.7-zip.org/download.html>)

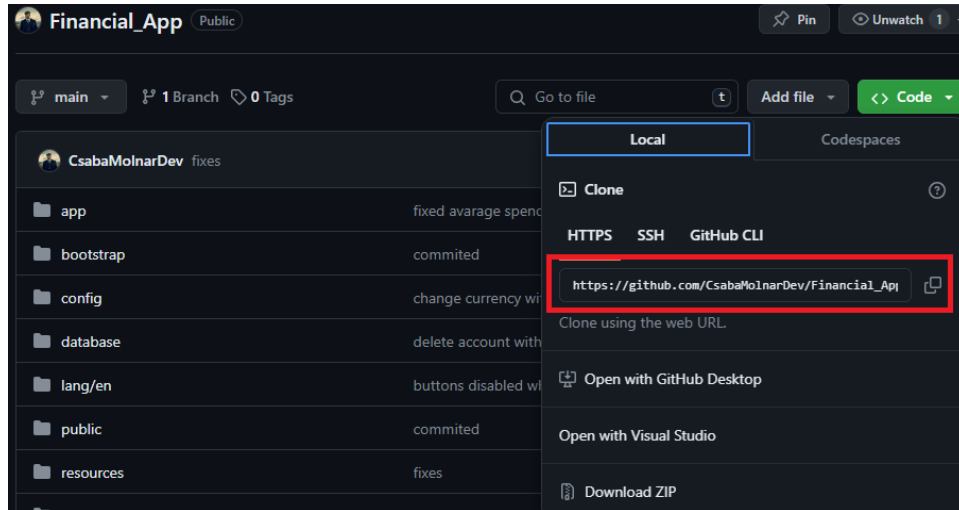
A. Futtasa a következő parancsot:

1. Hozzon létre új mappát “FinancialApp” néven, a következő helyen:

Ez a gép -> Helyilemez (C:, vagy ahova a xampp telepítve lett) -> xampp -> htdocs

2. Nyissa meg ezt a mappát a Visual Studio Codeban
3. Futtassa a következő parancsokat a Visual Studio Code termináljában:

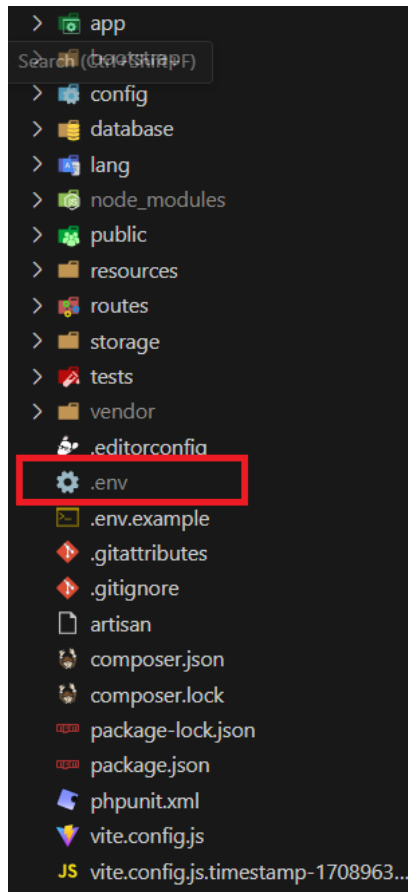
- git clone (url)



(Ezt az URL-t a képen láthatja)

(Amennyiben az URL-t nem találja, a mi GitHub repository-nk url-je a következő:
https://github.com/CsabaMolnarDev/Financial_App.git)

- composer install
4. Hozzon létre új fájlt “.env” néven.



5. .env example-ben lévő összes adatot másolja át a .env fájlba

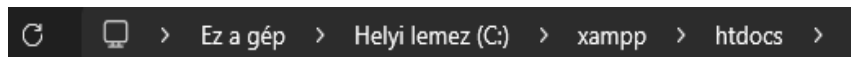
6. Futtassa a következő parancsokat a Visual Studio Code termináljában:

- npm install

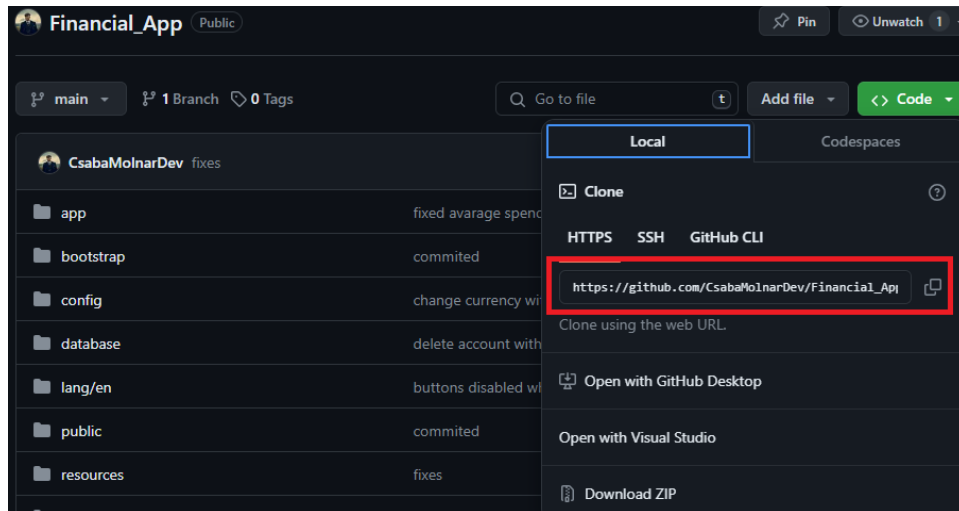
B. Futtassa a következő parancsot

composer create-project laravel/laravel FinancialApp

- A felső menüsorban nyomjon rá a “File” gombra, majd nyomjon rá az ”Open folder” gombra, válassza ki a “FinancialApp” mappát.



- Nyisson egy terminált és futtassa le a következő parancsokat:
 1. git init
 2. git remote add origin {githubról kimásolt URL}



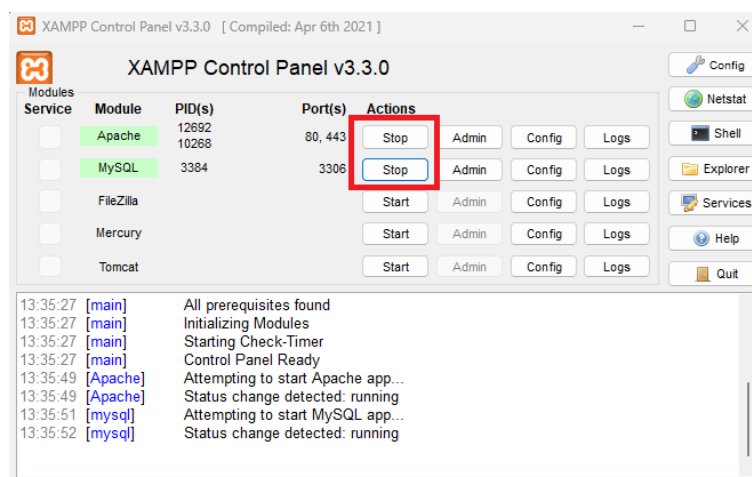
(Ezt az URL-t a képen láthatja)

(Amennyiben az URL-t nem találja, a mi GitHub repository-nk url-je a következő:
https://github.com/CsabaMolnarDev/Financial_App.git)

3. git pull

8. Hozzon létre adatbázist a következő néven: Financial

- Indítsa el a “Xampp” alkalmazást.
- Nyomjon rá a “Start” gombra a(z) “Apache” és “MySQL” module mellett.

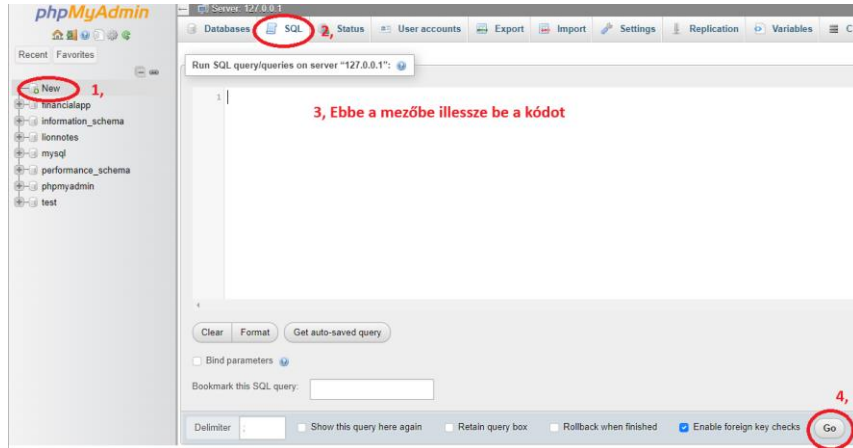


(Akkor jó, ha zöld lesz mind a kettő, lásd a fenti ábrán)

- Nyomjon rá az “Admin” gombra a MySQL modul mellett
- Kattintson a “new” gombra majd az “SQL” gombra.
- Írja be az alábbi kódot:


```
CREATE DATABASE Financial CHARACTER SET utf8 COLLATE
utf8_hungarian_ci;
```

- Ezt követően nyomja meg a “Go” gombot.



- A visual studio code termináljában futtassa le a következő parancsot, hogy az adatbázis fel legyen töltve: `php artisan migrate:fresh --seed`, illetve, ha így nem működne, akkor először `php artisan migrate:fresh`, és utána a `php artisan migrate:fresh --seed` parancsot.

(Erre azért van szükség mert a Laravel adatbázis feltöltését használjuk)

- Futtassa a következő kódokat a Visual Studio Code-ban

- `php artisan migrate:fresh`
- `php artisan migrate:fresh --seed`

- Mappák létrehozása a képeknek, iconoknak és videóknak.

- Link létrehozása a storage mappához, hogy a tárolt fájlokat elérjük, a következő paranccsal (a visual studio code termináljában):

```
php artisan storage:link
```

- A médiafájlok letöltése majd a mappák helyére tétele. Forrás:

[https://drive.google.com/drive/folders/1Ql60tXd04ir7gKhv2CBnWQzJP-DKtNmS?usp=drive link](https://drive.google.com/drive/folders/1Ql60tXd04ir7gKhv2CBnWQzJP-DKtNmS?usp=drive_link),

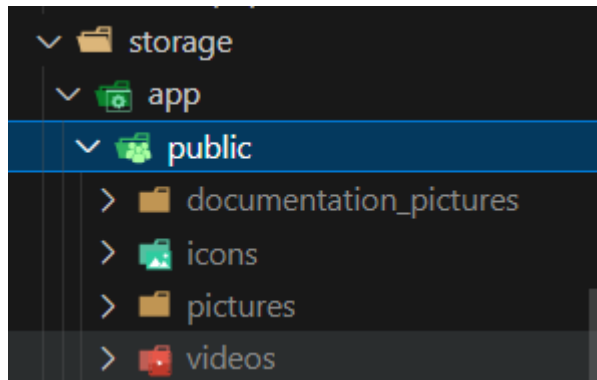
[https://drive.google.com/drive/folders/14584tiiwnkDk_xU2k5hYM10kMDEbCm4v?usp=drive link](https://drive.google.com/drive/folders/14584tiiwnkDk_xU2k5hYM10kMDEbCm4v?usp=drive_link),

[https://drive.google.com/drive/folders/11p6evOCWW7ulNtSxvdT-sFnBGte6I0df?usp=drive link](https://drive.google.com/drive/folders/11p6evOCWW7ulNtSxvdT-sFnBGte6I0df?usp=drive_link),

https://drive.google.com/drive/folders/1hY40HB5RTcUiJD3cQUh6XSFs3Xj2zZtd?usp=drive_link

A mappákat a következő helyre tegye: Ez a gép -> Helyilemez (C:, vagy ahova a xampp telepítve lett) -> xampp -> htdocs -> FinancialApp -> storage -> app -> public

Az alábbi mappa struktúrát követve:



A program használatának részletes leírása

Alkalmazás futtatása

(Bizonyos parancsok és funkciók működését akadályozhatja VPN illetve vírusírtó)

Futtassa az adatbázist.

1. A Visual Studio Code alkalmazásban a projekt mappájának megnyitása után, nyisson két terminált.
2. Futtassa a következő parancsokat eltérő terminálokban:
 - `php artisan serve`
 - `npm run dev`
3. nyissa meg a termináljában megjelenő linket vagy írja be egy böngészőbe hogy `localhost:8000`

A bejelentkezéshez az adatbázisban szereplő bármely tesztként létrehozott felhasználót használhatja, illetve regisztrálhat új felhasználót.

Navigációs sáv

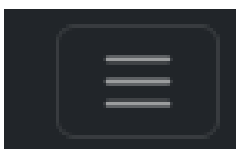


Itt található minden gomb, aminek a segítségével más aloldalakat tekinthetünk meg.

- A ház alakú ikon (icon) a navigációs sáv bal oldalán, 2 funkciót lát el. Az egyik az, ha az adott felhasználó be van jelentkezve akkor az alap kezdőoldalra irányítja, illetve, ha be vagyunk jelentkezve, akkor a főoldalra irányítja a felhasználókat.

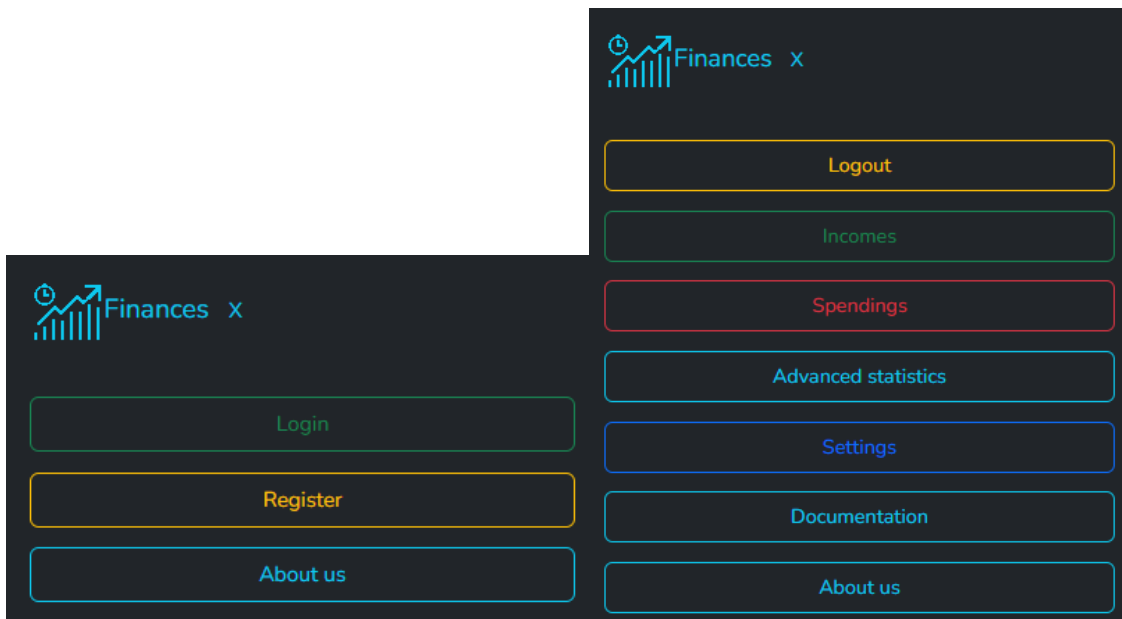


- A jobb oldalon található lenyíló menüben gombok találhatóak, amik a navigálásért felelnek.



- Az “X” gombra kattintva a lenyíló menüt a felhasználó be tudja zárni.
- A “Login” gombra kattintva a bejelentkezési aloldalt tudja a felhasználó megtekinteni.
- A “Register” gombra kattintva a regisztrációs aloldalt tudja a felhasználó megtekinteni.
- Az “About us” gombra kattintva a rólunk lévő információs aloldalt tudja a felhasználó megtekinteni.
- Az “Income” gombra kattintva a hozzáadott bevételeknek lévő aloldalt tudja a felhasználó megtekinteni. Ez a gomb csak akkor jelenik meg, ha a felhasználó bejelentkezett.
- A “Spending” gombra kattintva a hozzáadott kiadásoknak lévő aloldalt tudja a felhasználó megtekinteni. Ez a gomb csak akkor jelenik meg, ha a felhasználó bejelentkezett.
- A “Settings” gombra kattintva a beállítások aloldalt tudja a felhasználó megtekinteni. Ez a gomb csak akkor jelenik meg, ha a felhasználó bejelentkezett.

- A “Documentation” gombra kattintva a dokumentációs aloldalt lehet megtekinteni. Ez a gomb csak akkor jelenik meg, ha a felhasználó bejelentkezett.
- A “Advanced statistics” gombra kattintva a haladó statisztikák aloldalt lehet megtekinteni. Ez a gomb csak akkor jelenik meg, ha a felhasználó bejelentkezett.
- A “Log out” gombra kattintva a felhasználó ki tud jelentkezni. Ez a gomb csak akkor jelenik meg, ha a felhasználó bejelentkezett.



A “Home” (Kezdő) oldal

Itt egy szlogen, illetve egy gomb található. Ez a gomb “Getting started” (magyarul “Kezdés”) a regisztrációs aloldalra irányítja a felhasználót

A “Register” (regisztrációs) aloldal

Ezen aloldalon hat beviteli mező, egy gomb, egy hivatkozás és egy lenyíló menü található.

The screenshot shows a 'Register' form with the following fields and options:

- Fullname**: A text input field.
- Username**: A text input field.
- Phone (optional)**: A text input field with a dropdown menu for country code. The selected option is 'United Arab Emirates +971'.
- Currency**: A dropdown menu with 'Unit' selected. Other visible options include 'United Arab Emirates +971'.
- Email Address**: A text input field with a dropdown menu for country code. The selected option is 'United Kingdom +44'.
- Password**: A text input field with a dropdown menu for country code. The selected option is 'United States +1'.
- Confirm Password**: A text input field.
- Register**: A yellow button.
- I already have an account. Log in!**: A blue link.

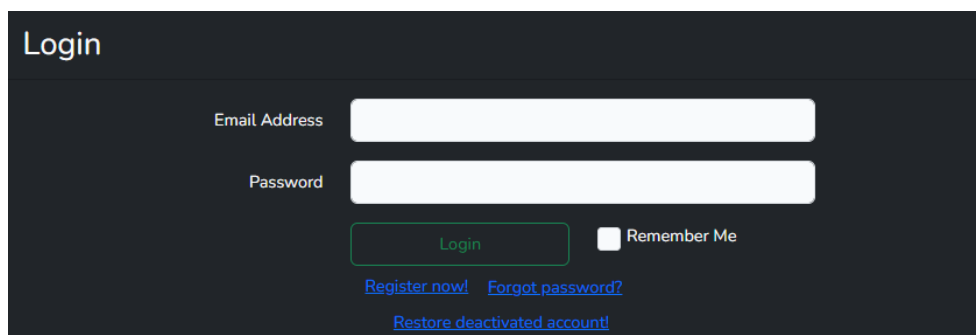
A regisztrációhoz a következő adatokat a következő formátumban kell megadni:

- Teljes név. A “Fullname” szöveg melletti beviteli mezőbe. Minimum 5 karakter.
- Felhasználó név. A “Username” szöveg melletti beviteli mezőbe. Minimum 5 karakter, egyedinek kell lennie.
- Telefonszám (ez opcionális). A “Phone (optional)” szöveg melletti beviteli mezőbe. A felhasználónak ki kell választania az országát, és ennek az országnak a telefonszám formátumának megfelelően kell megadnia a telefonszámát.
- Pénznem. A “Currency” szöveg melletti lenyíló menüben ki kell választania a felhasználónak egy preferált pénznemet.
- Levelezőcím (email). Az “Email Address” szöveg melletti beviteli mezőbe. Maximum 255 karakter, egyedinek kell lennie.
- Jelszó. A “Password” szöveg melletti beviteli mezőbe. Minimum 8 karakter. Javasolt speciális karakterek számok, kis- és nagy betűk használata.
- Jelszó megerősítés. A “Confirm password” szöveg melletti beviteli mezőbe. Meg kell egyeznie a jelszóval.

Az “I already have an account. Log in!” hivatkozás a bejelentkezés aloldalra irányítja a felhasználót.

A “Login” (bejelentkezés) aloldal

Ezen aloldalon két beviteli mező, egy gomb három hivatkozás és egy jelölőnégyzet található.



Ha a felhasználó be szeretne lépni akkor az “Email Address” szöveg melletti beviteli mezőbe a felhasználó által regisztrációkor megadott “email” (levelezési) címet kell megadnia, illetve a “Password” szöveg melletti beviteli mezőbe a regisztrációkor megadott jelszót. Ez után az oldalon látható “Login” gombra kattintva amennyiben helyes levelezési cím és jelszó párost adott meg a felhasználó, be lesz jelentkeztetve, és a főoldal fog a felhasználónak megjelenni Amennyiben a jelölőnégyzetet a felhasználó bepipálja (rákattint majd egy pipa jelenik meg a négyzetben), akkor a program elmenti a bejelentkezési adatait és ezután nem kell megadnia a bejelentkezési adatait.

A “Register now!” hivatkozásra kattintva gyorsan és könnyedén tudja a felhasználó a regisztrációs aloldalt elérni.

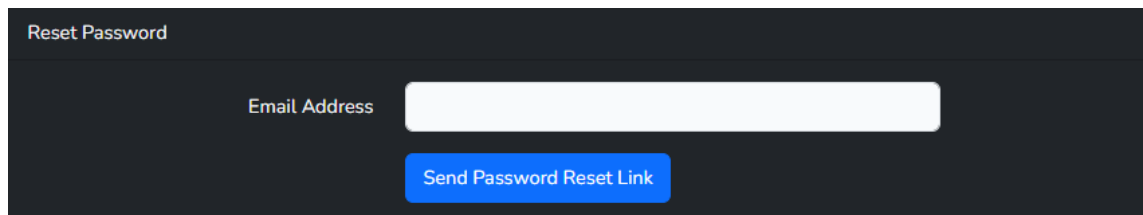
A “Forgot password?” hivatkozás egy olyan aloldalra irányítja a felhasználót, ahol az elfelejtett jelszavát vissza tudja állítani.

A “Restore deactivated account” hivatkozás egy olyan aloldalra irányítja a felhasználót, ahol az inaktívvá tett felhasználói fiókot vissza tudja állítani.

A “Forgot password” (elfelejtett jelszó) aloldal

Ezen az aloldalon egy bevitelmezőt, illetve egy gombot talál a felhasználó.

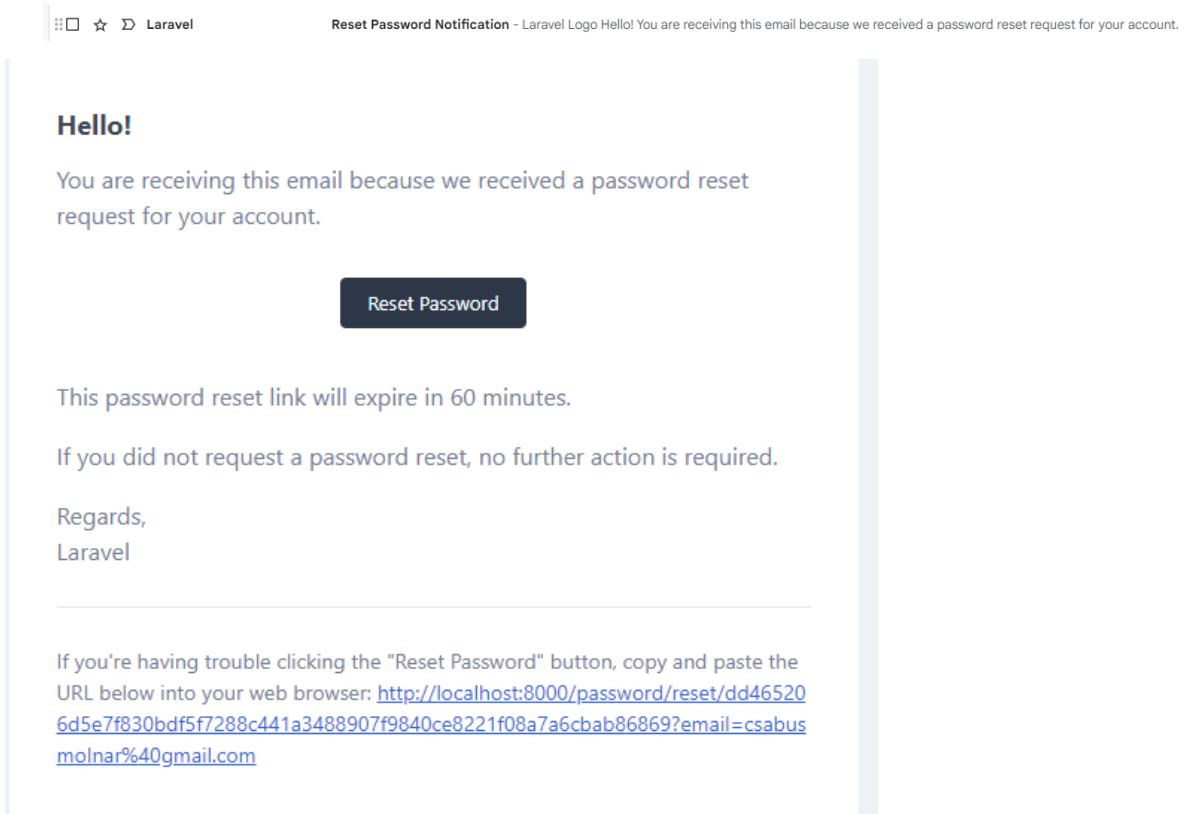
Ahhoz, hogy visszaállítsa régi felhasználói fiókját írja be az ahhoz a fiókhoz tartozó levelezési címét (email címét). Majd nyomja meg a “Send Password Reset Link” gombot. A gomb megnyomása után kiküldünk a felhasználónak egy levelet (emailt) a felhasználó által megadott levelezési címre és az ebben szereplő “Reset password” gombra kattintva, a “Reset password” (jelszó visszaállítása) aloldalra kerül a felhasználó.



A “Reset password” (jelszó visszaállítása) aloldal

Itt a felhasználó a bevitelmezők és a gomb használatával új jelszót tud állítani.

Az új jelszó létrehozásához a felhasználónak az “Email Address” (levelezési cím) mezőt a levelezési címével, a “password” (jelszó) és a “Confirm password” (jelszó megerősítés) mezőket egy minimum 8 karakter hosszú azonos karakter láncsal (kis- és nagy betű, szám és speciális karakter) kell kitöltenie



Reset Password

Email Address

Password

Confirm Password

Reset Password

A “Restore deactivated account” (felhasználói fiók visszaállítása) aloldal

Ahoz, hogy inaktívvá tett felhasználói fiókját újra aktívvá tehesse a felhasználó, az “Email Address” (levelezési cím) beviteli mezőbe meg kell adnia a felhasználói fiókhoz tartozó levelezési címet (emailt) a felhasználónak. Ezt követően a felhasználónak rá kell nyomnia a “Reactivate Account” (felhasználói fiók visszaállítása) gombra. Ezután a felhasználó felhasználói fiókját a program visszaállítja, majd visszairányítja a felhasználót a bejelentkezési felületre.

Reactivate Account

Email Address

Reactivate Account


Az "About us" (Rólunk) aloldal

Itt a projektünk, alkalmazásunk történetét, négy reklámozott funkciót, illetve a csapatunk elérhetőségét találhatja meg a felhasználó.

A reklám kártyák automatikusan mozognak, viszont, ha a felhasználó felé viszi az egerét, akkor megáll az animáció. Amennyiben manuálisan szeretné a felhasználó lapozgatni erre is van lehetősége. Kattintson az egyik kártyára bal egér gombbal és tartsa lenyomva, majd húzza az egerét a jobbra vagy balra.


The story of our project in short

Our little project is a web application designed for financial management. We chose this idea because we wanted to create a project that we could both use in the future by running it on a server and continue to develop as we please. The choice of topic mainly stemmed from these aspects. On the other hand, we have always been interested in finance. We believe that this small project, as well as this application, even in its current state, could provide a foundation for us and others in managing their finances and achieving their financial goals. We believe that if people are able to track their expenses even for a few months, significant cost savings can be achieved.




Access anywhere

You can access our product from anywhere on the globe



Remember everything

With the help of our product, you can easily track and manage your finances



Family system

You can create a group with your family, and you can manage the finances together.

Contact us here

- 24/7 on email: laravelmybeloved@gmail.com

A

"Home"

főoldal

A Főoldal tartalmazza a bevételek és kiadások mind személyre mind családra vonatkozó összefoglalását kör ábrán. Alatta(uk) található a naptár, amelyben megjelenik minden bevétel és kiadás (bevétel zöld, kiadás piros színben), még a családtagoké is, a havonta történő kiadások és bevételek eltérő egy világosabb árnyalatban jelennek meg. A naptár

alatt az oldal legalján található egy valuta átváltó, ami megmutatja mennyit ér egy pénznem a másikkal szemben.

Az "Incomes" (Bevételek) aloldal

Az "Incomes" (Bevételek) oldalon található egy "Add new income" (Új bevételek rögzítése) gomb, ami mindig az oldal tetején van. Erre a gombra kattintva a felhasználó az "Income Create" (Bevétel hozzáadása) aloldalra tud menni. Amennyiben már rendelkezik a felhasználó bevétellel akkor két ábra és egy táblázat megjelenik. Az egyik egy kör diagramm a másik egy vonal diagramm. A vonal diagramm hónapokra bontva az összegnek megfelelően helyez el pontokat majd köti össze azokat, kategóriákra bontva. A táblázat jeleníti meg a részletesebb adatokat, illetve, hogy valami havi rendszerességgel bekerül e vagy sem. A "Delete" (Törlés) gombra kattintva törölheti a felhasználó az előzőleg rögzített adatokat, amennyiben nem havonta történő adatot szeretne törölni. Amennyiben a törölni kívánt adat havonta ismétlődő, akkor a felhasználónak először módosítani kell az adatot, hogy az ne havonta ismétlődő legyen. Az megadott értékeket a felhasználó módosítani tudja a bal oldali egér gomb kétszer egymás utáni gyorsan kattintásával. Ekkor szerkeszthetővé válik az adat, amit vagy az enter billentyű, vagy, a felhasználó máshova kattintásával tud elmenteni.

Az "Add Income" (Bevételek hozzáadása) aloldal

A "Create income" (Bevétel hozzáadása) aloldal biztosít lehetőséget új bevétel és bevételi kategóriák hozzáadására. Új bevétel rögzítéséhez a következő adatokat kell megadnia a felhasználónak: 1, Bevétel neve 2, Bevétel összege 3, Bevétel kategóriája. Amennyiben új kategóriát szeretne a felhasználó hozzáadni, a "Category" (Kategória) lenyíló menüben az "Add new Category" opciót választva az lehetősége létrehozni. A bevételi mezőbe az új kategória nevét kell megadnia és a gombra kattintva kerül be a rendszerbe. Ezt követően már ez is választható opcióvá válik. A bevételt az "Add new" gombbal hozhatja létre a felhasználó és visszairányítja a felhasználót az "Incomes" aloldalra.

A "Spending" (Kiadások) aloldal

A "Spending" (Kiadások) oldalon található egy "Add new spending" (Új kiadás rögzítése) gomb, ami mindig az oldal tetején van. Erre a gombra kattintva a felhasználó az "Spending Create" (Kiadás hozzáadása) aloldalra tud menni. Amennyiben már rendelkezik a felhasználó kiadással akkor két ábra és egy táblázat megjelenik. Az egyik egy kör diagramm a másik egy vonal diagramm. A vonal diagramm hónapokra bontva az összegnek megfelelően helyez el pontokat majd köti össze azokat, kategóriákra bontva.

A táblázat jeleníti meg a részletesebb adatokat, illetve, hogy valami havi rendszerességgel bekerül e vagy sem. A "Delete" (Törlés) gombra kattintva törölheti a felhasználó az előzőleg rögzített adatokat, amennyiben nem havonta történő adatot szeretne törölni. Amennyiben a törölni kívánt adat havonta ismétlődő, akkor a felhasználónak először módosítania kell az adatot, hogy az ne havonta ismétlődő legyen. Az megadott értékeket a felhasználó módosítani tudja a bal oldali egér gomb kétszer egymás utáni gyorsan kattintásával. Ekkor szerkeszthetővé válik az adat, amit vagy az enter billentyű, vagy, a felhasználó máshova kattintásával tud elmenteni.

Az "Add Spending" (Kiadások hozzáadása) aloldal

A "Create spending" (Kiadás hozzáadása) aloldal biztosít lehetőséget új kiadás és kiadási kategóriák hozzáadására. Új kiadás rögzítéséhez a következő adatokat kell megadnia a felhasználónak: 1, Kiadás neve 2, Kiadás összege 3, Kiadás kategóriája. Amennyiben új kategóriát szeretne a felhasználó hozzáadni, a "Category" (Kategória) lenyíló menüben az "Add new Category" opciót választva a lehetősége létrehozni. A beviteli mezőbe az új kategória nevét kell megadnia és a gombra kattintva kerül be a rendszerbe. Ezt követően már ez is választható opcióvá válik. A kiadást az "Add new" gombbal hozhatja létre a felhasználó és visszairányítja a felhasználót a "Spending" aloldalra.

A "Setting" (Beállítások) aloldal

Itt található a felhasználó által módosítható személyes adatai, illetve itt van alkalma létrehozni családot, kilépni családból vagy törölni családot. A "Deactivate account" gombra kattintva a felhasználói fiókját fel tudja függeszteni. Az információk szerkesztése egyszerű, minden adat alatt található egy gomb, aminek a kattintásakor megjelenik egy felugró mező (modal), ahol az új információkat tudja megadni és az "Save Changes" (Módosítások mentése) gombra kattintva módosulnak az adatokat. A pénznem módosításához a pénznemet a lenyíló menüben kell a felhasználónak kiválasztania, ezután a "Change currency" (pénznem módosítása) gombra való kattintást követően megváltozik.

Továbbá a felhasználó beállíthatja, hogy kapjon-e értesítést. Az értesítés emailben lesz kiküldve, a felhasználó időzónája alapján 18:00-kor.

Az "Advanced Statistics" (Haladó Statisztikák) aloldal

Ez az oldal tartalmazza a felhasználó tulajdonában lévő kiadások és bevételek összesített

adatait. Ezeket az adatokat kategóriákra bontva is meg tudja tekinteni. Az oldal tetején a felhasználó jelenlegi egyenlegét írja ki. Alatta láthatja az összesített bevételeket és kiadásokat az elmúlt három hónapban. Amennyiben van családtag akkor az egész család összesített adatát is megjeleníti. Ez alatt található egy lenyíló menü, ahol kategóriára bontva tudja az adatokat megtekinteni három hónapos intervallumban. Kizárólag abban az esetben tudja a felhasználó megtekinteni, amennyiben rendelkezik bármilyen bevétellel vagy kiadással. Alatta található a mostani hónapban történt bevételek és kiadások listája kategóriára bontva, ezentúl amennyiben van hozzáadott családtagja, ki tudja választani azt a családtagját és össze tudja mérni a bevételeiket és kiadásaikat az adott hónapban. Az oldal legalján meg tudja tekinteni az adott pénznemmel rendelkező felhasználók átlagolt értékeit.

A “Documentation” (Dokumentáció) aloldal

Itt a felhasználó az adott kategória gombjára kattintva kap egy részletes magyarázatot. A gomb ismételt megnyomásával eltűnik a magyarázat.

Fejlesztői dokumentáció

Az alkalmazott fejlesztői eszközök

A Microsoft által kiadott Visual Studio Code-ot használtuk a weboldalak szerkesztéséhez. A MySQL szerveret a XAMPP Control Panel segítségével futtatjuk.

Egyes letöltött ikonok kinézetének módosításához az Aseprite nevű alkalmazást használtuk.

Teszteléshez a következő böngészőket használtuk:

- Opera GX
- Google Chrome
- Microsoft edge
- Vivaldi
- Firefox Mozilla
- Brave

A projekt a Laravel keretrendszerében készült.

A következő programozási, leíró nyelveket és stílus lapot használtuk:

- Php
- Javascript
- Html
- Css

Adatmodell leírása (Adatbázis)

Az adatbázis a MySQL-t használja. Az adatbázis neve Financial, amit egy SQL kód segítségével lehet létrehozni.

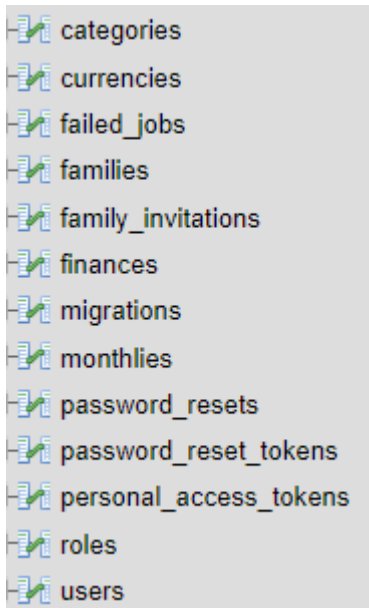
```
1 CREATE DATABASE Financial
2 CHARACTER SET utf8
3 COLLATE utf8_hungarian_ci;
```

Az adatbázis tábláit a Laravel által létrehozott database\migration mappában található fájlok hozzák létre, amiket a seedersben lévő fájlok töltenek fel adatokkal, a véletlenszerű adatokhoz a seederek a factories mappa tartalmát veszik igénybe.

A factoriesben olyan fájlok vannak, amik egy táblában lévő oszlopoknak a követelményeinek megfelelően képes adatokat létrehozni, amiket az adott seeder feltölt. Egy táblához egy seeder és egy factory tartozik. Az adatbázis feltöltéséhez a VS Code termináljában le kell futtatni az alábbi parancsot:

```
php artisan migrate:fresh --seed
```

Ezzel a nulláról hozza létre a táblákat és a bennük lévő adatokat, ha már volt benne adat vagy egyéb tábla akkor azokat először kitörli. A mi adatbázisunk 13 táblát használ:



Ezekből a táblákból főként az alábbi táblákat használjuk:

- Categories:

A Categories tábla tartalmazza a kiadások és bevételek létrehozásánál és szerkesztésénél lévő választható elemeket. Egy felhasználó csak a saját azonosítójával ellátott kategóriákat látja, plusz az alap kategóriákat, ahol a felhasználó azonosító

nulla.

| owner_id | name | type |
|----------|----------------|----------|
| 0 | Freetime | spending |
| 0 | Hobby | spending |
| 0 | Food | spending |
| 0 | Sport | spending |
| 0 | Transportation | spending |
| 0 | Salary | income |
| 0 | Bonus | income |
| 0 | Investment | income |
| 0 | Gift | income |
| 0 | Rent | income |

- Currencies:

A Currencies tartalmazza az összes pénznemet, azoknak neveivel, rövidítéseikkel és jelzésükkel. Ezt a regisztrációnál kell kiválasztani, illetve a beállítások oldalon lehet átállítani. A szimbólumokat automatikusan odateszi az oldal minden pénzösszeg mögé.

| name | code | symbol |
|--------------------|------|--------|
| Albanian Leke | ALL | Lek |
| US Dollars | USD | \$ |
| Afghan Afghanis | AFN | ؑ |
| Argentinian Pesos | ARS | \$ |
| Aruban Guilders | AWG | f |
| Australian Dollars | AUD | \$ |

- Families:

A Families tartalmazza a családfő (a családot létrehozó személy) család nevét és azonosítóját.

| name | creator_id |
|--------------|------------|
| Láng | 7 |
| Ambrus-Dobai | 8 |

- Family_invitations

A Family_invitations tartalmazza a meghívó fél családazonosítóját, a meghívott internetes levelező elérhetőségét, a küldő fél azonosítóját, egy titkosított családi

belépési ellenőrzőkódot és a meghívás jelenlegi állapotát.

| family_id | recipient_email | sender_id | token | status |
|-----------|---------------------------|-----------|---|---------|
| 4 | lehner.claude@example.org | 8 | u7y1xJgF0X6FBBv5rTM2zwOLT60RGk5XkF3juYW3GPMhAxbNtW... | pending |

- Finances:

A Finances tartalmazza a kiadásokat és bevételeket az alábbi felosztásban. Tartalmazza a felhasználó azonosítóját, a típusát (bevétel vagy kiadás), a nevét, a mennyiségét/árát, létrehozásának dátumát, a választott kategória azonosítóját és a pénznem azonosítóját.

| user_id | type | name | price | time | category_id | currency_id |
|---------|----------|--------|-------|------------|-------------|-------------|
| 8 | Spending | Süti | 107 | 2024-04-29 | 4 | 2 |
| 8 | Spending | Kajcsi | 201 | 2024-04-29 | 3 | 2 |
| 8 | Income | Süti | 20 | 2024-04-29 | 8 | 2 |
| 7 | Income | Munka | 30 | 2024-04-30 | 6 | 2 |
| 7 | Spending | Süti | 20 | 2024-04-30 | 3 | 2 |
| 8 | Spending | Kajcsi | 201 | 2024-03-01 | 5 | 2 |
| 8 | Spending | Kajcsi | 201 | 2024-03-30 | 1 | 2 |
| 8 | Spending | Kajcsi | 201 | 2024-04-30 | 3 | 2 |

- Monthlies:

A Monthlies tartalmazza egy adott kiadás vagy bevétel azonosítóját, létrehozásának éve és hónapját, hogy a kódba ne keljen pluszba felosztani, ezért két külön táblába tároljuk.

| finance_id | year | month |
|------------|------|-------|
| 2 | 2024 | 5 |
| 4 | 2024 | 1 |
| 3 | 2024 | 5 |

Users

A Users a felhasználók adatait tartalmazzák.

A tábla az alábbi oszlopokból áll:

a felhasználó teljes neve, felhasználó neve, telefonszáma, profilképe, internetes levelezési elérhetősége,

| fullname | username | phone | picture | email |
|-----------------------------|--------------------------------|-----------------------|---------|----------------------------|
| Prof. Junior Crooks I | oconnell.meredith | 1-217-993-5986 | NULL | elroy.renner@example.net |
| Lloyd Hessel | damore.brennan | +18135871792 | NULL | felton99@example.org |
| Imogene Blanda | grempel | +1-251-970- 9173 | NULL | lehner.claude@example.org |
| Mr. Lowell Mertz I | jhodkiewicz | +1 (918) 264- 7791 | NULL | tia82@example.net |
| Jolie Ledner | hhagenes | +1.986.450.3503 | NULL | bernier.eugene@example.org |
| Molnár Csaba | Frontend deeloper | +36 00 000 0001 | NULL | frontenddeeloper@dev.com |
| Láng Ricsi | Ideamaker/backend developer | +36 00 000 0002 | NULL | ideamakerdeveloper@dev.com |
| Ambrus- Dobai Kristóf | Backend Developer | +360000000003 | NULL | backenddeveloper@dev.com |

internetes levelezési elérhetőségének hitelesítésének pillanatát, a laravel által titkosított jelszavát, hogy szeretne-e értesítéseket kapni, ahhoz mérten az időzónáját.

| email_verified_at | password | notification | timezone |
|------------------------|--|--------------|----------|
| 2024-04-29 19:22:47 | \$2y\$12\$oBM.aEqEq90oFVNHEfYsuO77cxxGLC72kT1E2DYDwrz... | 0 | NULL |
| 2024-04-29 19:22:47 | \$2y\$12\$oBM.aEqEq90oFVNHEfYsuO77cxxGLC72kT1E2DYDwrz... | 0 | NULL |
| 2024-04-29 19:22:47 | \$2y\$12\$oBM.aEqEq90oFVNHEfYsuO77cxxGLC72kT1E2DYDwrz... | 0 | NULL |
| 2024-04-29 19:22:47 | \$2y\$12\$oBM.aEqEq90oFVNHEfYsuO77cxxGLC72kT1E2DYDwrz... | 0 | NULL |
| 2024-04-29 19:22:47 | \$2y\$12\$oBM.aEqEq90oFVNHEfYsuO77cxxGLC72kT1E2DYDwrz... | 0 | NULL |
| 2024-04-29 19:22:47 | \$2y\$12\$oBM.aEqEq90oFVNHEfYsuO77cxxGLC72kT1E2DYDwrz... | 0 | NULL |
| 2024-04-29 19:22:47 | \$2y\$12\$oBM.aEqEq90oFVNHEfYsuO77cxxGLC72kT1E2DYDwrz... | 0 | NULL |
| 2024-04-29 19:22:47 | \$2y\$12\$oBM.aEqEq90oFVNHEfYsuO77cxxGLC72kT1E2DYDwrz... | 0 | NULL |

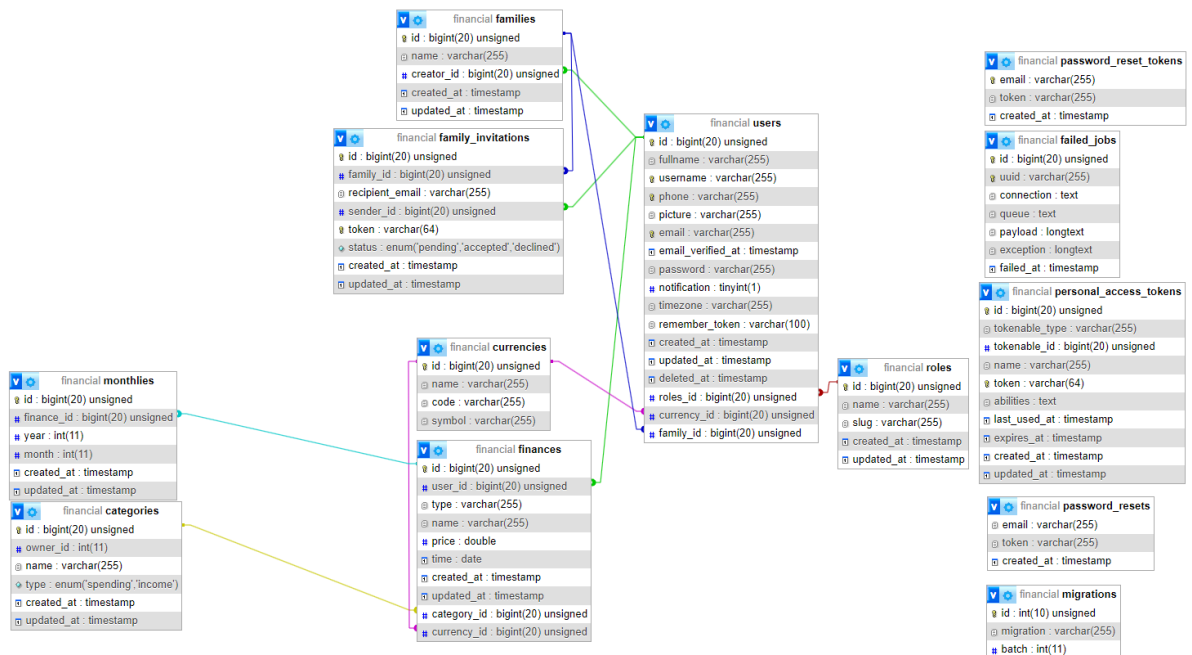
emlékeztetőjének azonosítóját, létrehozás pillanatát, szerkesztésének pillanatát (ezt a kettőt mindegyik tábla tartalmazza), törlésének pillanatát,

| remember_token | created_at | updated_at | deleted_at |
|---|------------------------|------------------------|------------|
| CclynI0jKx | 2024-04-29 19:22:47 | 2024-05-02 13:46:58 | NULL |
| yVjrT6x4b3 | 2024-04-29 19:22:47 | 2024-04-29 19:22:47 | NULL |
| a1RXWmc12T | 2024-04-29 19:22:47 | 2024-04-29 19:22:47 | NULL |
| H82U8rXDdO | 2024-04-29 19:22:47 | 2024-04-29 19:22:47 | NULL |
| gdLZdUtoY0 | 2024-04-29 19:22:47 | 2024-04-29 19:22:47 | NULL |
| 0V39sUufyY | 2024-04-29 19:22:47 | 2024-04-29 19:22:47 | NULL |
| DZCCNROuKhyIDRCRI0kVypCqCaXrcnoO75P1tikNCguKLxot16... | 2024-04-29 19:22:47 | 2024-04-30 11:19:05 | NULL |
| KCoYWY25TPBKQhdoNCLXER8L7Bfg4kgz5Ajb4K5BXjMqDeMxkg... | 2024-04-29 19:22:47 | 2024-05-03 13:24:40 | NULL |

jogosultság azonosítóját, pénznem azonosítóját és család azonosítóját.

| roles_id | currency_id | family_id |
|----------|-------------|-----------|
| 2 | 2 | 4 |
| 2 | 2 | NULL |
| 2 | 2 | NULL |
| 2 | 2 | NULL |
| 2 | 2 | NULL |
| 1 | 2 | NULL |
| 1 | 2 | 3 |
| 1 | 2 | 4 |

Ezek a táblák különböző pontokon kapcsolódnak egymáshoz:



Ezeket a kapcsolatokat a laravel is használja, ezzel leegyszerűsítve a különböző táblák használatát.

A laraveles kapcsolatok úgy épülnek fel, hogy a jogosultságok birtokolják a felhasználókat a role_id-n keresztül és a felhasználók ugyan így tartoznak a jogosultságokhoz:

Models/Role.php

```
public function users() : HasMany{
    return $this->hasMany(User::class);
}
```

Models/User.php

```
public function roles() : BelongsTo{
    return $this->belongsTo(Role::class);
}
```

A felhasználók még tartoznak a pénznemekhez és a családokhoz a currency_id és a family_id-n keresztül:

Models/User.php

```
public function currency() : BelongsTo{
    return $this->belongsTo(Currency::class);
}

public function family() : BelongsTo{
    return $this->belongsTo(Family::class, 'family_id','id');
}
```

Models/Currency.php

```
public function users() : BelongsToMany{
    return $this->belongsToMany(User::class);
}
```

Models/Family.php

```
public function users() : HasMany{
    return $this->hasMany(User::class, 'family_id', 'id');
}
```

A felhasználók ezen kívül még birtokolják a pénzügyek táblát a user_id-n keresztül:

Models/User.php

```
public function finances() : HasMany{
    return $this->hasMany(Finance::class);
}
```

Models/Finance.php

```
public function users() : BelongsTo{
    return $this->belongsTo(User::class);
}
```

Illetve a pénzügyek még a kategóriákhoz és a pénznemekhez is tartozik az azonosítóikon keresztül:

Models/Finance.php

```
public function category() : BelongsTo{
    return $this->belongsTo(Category::class);
}
public function currency() : BelongsTo{
    return $this->belongsTo(Currency::class, 'currency_id', 'id');
}
```

Models/Currency.php

```
public function finance() : HasMany{
    return $this->hasMany(Finance::class, 'currency_id', 'id');
}
```

Models/Category.php

```
public function finance() : HasMany{
    return $this->hasMany(Finance::class);
}
```

A pénzügyek birtokolják a havi táblát:

Models/Finance.php

```
public function monthly() : HasOne{
    return $this->hasOne(Monthly::class, 'finance_id', 'id');
}
```

Models/Monthly.php

```
public function finance() : BelongsTo{
    return $this->belongsTo(Finance::class, 'finance_id', 'id');
}
```

Hasonló módon kapcsolódik a családoknak a táblája a családmeghívás táblához ami még a felhasználókhöz is tartozik.

Részletes feladatspecifikáció, algoritmusok

Controllerek működése

HomeController működése:

A HomeController felel a home oldal backendjéért. A home oldalon jelenítjük meg a felhasználó havi kiadásait és bevételeit egy kördiagrammon. Ezenkívül, ha a felhasználónak van családja, és az oldalon hozzá vannak adva családtagok, akkor a családtagok kiadásait és bevételeit is megjelenítjük százalékos megoszlásban.

A naptárban külön színekkel jelöljük a kiadásokat és bevételeket. A színek eltérőek attól függően, hogy kiadásról vagy bevételről van-e szó, illetve, hogy havi kiadásról vagy bevételről van-e szó. Ha a felhasználónak van családja, akkor a naptárban megjelenítjük a családtagok adatait is.

Bónuszként az oldal legalján egy árfolyam átváltó, úgynevezett árfolyam figyelő is működik, amely lehetővé teszi bármely pénznem átváltását egy másikra.

A controllerben két függvény van, amelyek visszaadnak (return) egy nézetet. A többi függvény külön van szedve a különböző számításokhoz és az úgynevezett Eloquent query-khoz, amelyeket aztán az index és a calculate függvények hívnak meg.

Kiegészítő függvények:

ListCurrencies()

Ez a függvény a currency model használatával lekéri az összes valutát, majd visszaadja azt.

```

public function ListCurrencies()
{
    $currencies = Currency::all();
    return [
        'currencies' => $currencies,
    ];
}

```

spendingGraphData()

Ez a függvény felelős a home oldalon található havi kiadás grafikonért, először lekérdezzük azokat a kiadásokat, amelyek a felhasználóhoz tartoznak, és ebben az évben, illetve ebben a hónapban lett hozzáadva az adatbázishoz. Ezután létrehozunk egy üres tömböt, ahova feltöltjük az imént lekért adatokból a nevet és az összeget.

```

public function spendingGraphData()
{
    // Get the current month and year
    $currentMonth = Carbon::now()->format('m');
    $currentYear = Carbon::now()->format('Y');

    // Query to get finances data for the current month
    $spendingFinances = Finance::where('type', 'Spending')
        ->whereMonth('time', '=', $currentMonth)
        ->whereYear('time', '=', $currentYear)
        ->where('user_id', auth()->user()->id)
        ->get();

    // Organize the finances data by category
    $categoryPrices = [];
    foreach ($spendingFinances as $finance) {
        $categoryName = $finance->name;
        $price = $finance->price;
        if (!isset($categoryPrices[$categoryName])) {
            $categoryPrices[$categoryName] = $price;
        } else {
            $categoryPrices[$categoryName] += $price;
        }
    }

    return [
        'categoryPrices' => $categoryPrices
    ];
}

```

incomeGraphData()

Ez a függvény tulajdonképpen majdnem ugyanazt csinálja, mint az imént leírt azzal a különbséggel, hogy a bevétellel csináljuk meg ugyanezt.

```

public function incomeGraphData()
{
    // Get the current month and year
    $currentMonth = Carbon::now()->format('m');
    $currentYear = Carbon::now()->format('Y');

    // Query to get finances data for the current month
    $pendingFinances = Finance::where('type', 'Income')
        ->whereMonth('time', '=', $currentMonth)
        ->whereYear('time', '=', $currentYear)
        ->where('user_id', auth()->user()->id)
        ->get();

    // Organize the finances data by category
    $categoryPrices = [];
    foreach ($pendingFinances as $finance) {
        $categoryName = $finance->name;
        $price = $finance->price;
        if (!isset($categoryPrices[$categoryName])) {
            $categoryPrices[$categoryName] = $price;
        } else {
            $categoryPrices[$categoryName] += $price;
        }
    }

    return [
        'categoryPrices' => $categoryPrices
    ];
}

```

getFinances()

A függvény lekéri azokat a kiadásokat és bevételeket, amelyek a felhasználóhoz tartoznak, de csak a nevet, az árat, a hozzáadás dátumát, a típusát és az id-t (azonosítót) kérjük át.

```

public function getFinances()
{
    $userfinances = Finance::where('finances.user_id', '=', auth()->user()->id)
        ->select('name', 'price', 'time', 'type', 'id')
        ->get();
    return [
        'userfinances' => $userfinances
    ];
}

```

getFamilyMembers()

Itt lekérjük a felhasználóhoz tartozó családtagokat.


```

public function getFamilyMembers() {
    $currentUserFamilyId = auth()->user()->family_id;

    $familymembers = User::whereNotNull('family_id')
        ->where('family_id', $currentUserFamilyId)
        ->get();
    return [
        'familymembers' => $familymembers
    ];
}

```

getFamilyFinances()

Ez a függvény először meghívja a fentebb leírt getFamilyMembers() függvényt. Egy foreach-el végig megyünk a családtagokon és minden egyes családtagnak a pénzügyeit hozzáadjuk a userFinances tömbhöz, amit aztán visszaadunk

```

public function getFamilyFinances()
{
    $getFamilyMembers = $this->getFamilyMembers();
    $familymembers = $getFamilyMembers['familymembers'];

    $userFinances = collect();
    foreach ($familymembers as $member) {
        $finances = Finance::where('user_id', $member->id)->get();
        $userFinances = $userFinances->concat($finances);
    }
    return [
        'familyFinances' => $userFinances
    ];
}

```

generateFamilyIncomeArticles()

A **getFamilyMembers()** meghívásával a családtagokon végig menve lekérjük azokat az bevételeket, amelyek a jelenlegi évben, illetve hónapban lettek hozzáadva és összeadjuk, majd az articles tömbbe feltöltjük a user id-ját, a teljes nevét, és az imént összeadott értéket a '\$familyIncome' változóból. Ez a függvény felelős a család bevétel (income) grafikon adataiért.

```

public function generateFamilyIncomeArticles()
{
    $familyFunc = $this->getFamilyMembers();
    $familymembers = $familyFunc['familymembers'];

    $articles = [];

    foreach ($familymembers as $user) {
        $familyIncome = $user->finances()
            ->where('type', 'Income')
            ->whereMonth('time', now()->month)
            ->whereYear('time', now()->year)
            ->sum('price');

        $articles[] = [
            'user_id' => $user->id,
            'user_fullname' => $user->fullname,
            'family_income' => $familyIncome,
        ];
    }

    return [
        'articles' => $articles
    ];
}

```

generateFamilySpendingArticles()

Ez a függvény ugyanazért felel, csak annyi különbséggel, hogy kiadásokat összegzi családtagonként.

```

public function generateFamilySpendingArticles()
{
    $familyFunc = $this->getFamilyMembers();
    $familymembers = $familyFunc['familymembers'];

    $articles = [];

    foreach ($familymembers as $user) {
        $familySpending = $user->finances()
            ->where('type', 'Spending')
            ->whereMonth('time', now()->month)
            ->whereYear('time', now()->year)
            ->sum('price');

        $articles[] = [
            'user_id' => $user->id,
            'user_fullname' => $user->fullname,
            'family_spending' => $familySpending ,
        ];
    }

    return [
        'articles2' => $articles
    ];
}

```

getFinanceColor(\$financeId)

Itt történik a színmeghatározás a naptárhoz, megnézzük, hogy az adott kiadás vagy bevétel szerepel-e a havi (Monthly) táblában, megkeressük a pénzügy (Finance) táblában, a típusa meghatározásához. és ha havi bevétel akkor **lime** színt kap, ha kiadás akkor **pirost**, ha nem havi bevétel akkor **zöld**, illetve, ha kiadás akkor **crimson** színt kap a naptárban.

```

public function getFinanceColor($financeId)
{
    $isMonthly = Monthly::where('finance_id', $financeId)->exists();
    $finance = Finance::find($financeId);
    $type = $finance->type;

    if ($isMonthly) {
        return ($type === 'Income') ? 'lime' : 'red';
    } else {
        return ($type === 'Income') ? 'green' : 'crimson';
    }
}

```

Az `index()` függvényben végigmegyünk a **userfinances**, illetve a **familyFinances**-eken (ha létezik), és az id-jukat átadva meghívjuk a `getFinanceColor` metódust a színük meghatározására, és ezen színeket adjuk tovább az `index` függvényben a **financeColors**, és a **familyFinanceColors** tömbökben.

```

$getfinances = $this->getFinances();
$userMonthlyFinances = $getfinances['userfinances'];

$getFamilyFinances = $this->getFamilyFinances();
$familyFinances = $getFamilyFinances['familyFinances'];

$financeColors = [];
foreach ($userMonthlyFinances as $finance) {
    $financeColors[$finance->id] = $this->getFinanceColor($finance->id);
}

if ($familyFinances != null) {
    $familyFinanceColors = [];
    foreach ($familyFinances as $finance) {
        $familyFinanceColors[$finance->id] = $this->getFinanceColor($finance->id);
    }
}

```

Az **allIncomesZero()**, valamint az **allSpendingZero()** metódusok felelősek azért, hogy ne rajzolódhasson ki egy olyan grafikon, ahol 0 adat van, mivel a `generateFamilySpendingArticles()` összesíti a családtagok bevételeit és kiadásait a jelenlegi hónapra, és továbbadja a 0-át ezért kirajzolódik egy vonal a kördiagramban. Ezt megelőzően, csak akkor mutatjuk meg a grafikont, ha legalább az egyik családtag beleértve a mi fiókunkat is, adott hozzá kiadást vagy bevételt a hónapban. Ha legalább

egy családtag bevétele vagy kiadása nem 0, akkor az \$allIncomesZero, illetőleg az \$allSpendingZero változót false-ként küldjük vissza, ellenkező esetben null-t ad vissza.

```
public function allIncomesZero()
{
    $familyIncomesCall = $this->generateFamilyIncomeArticles();
    $familyIncomes = $familyIncomesCall['articles'];
    $allIncomesZero = true;
    foreach ($familyIncomes as $income) {
        if ($income['family_income'] != 0) {
            $allIncomesZero = false;
            break;
        }
    }
    return $allIncomesZero ? null : false;
}

public function allSpendingZero()
{
    $familySpendingCall = $this->generateFamilySpendingArticles();
    $familySpending = $familySpendingCall['articles2'];
    $allSpendingZero = true;
    foreach ($familySpending as $spending) {
        if ($spending['family_spending'] != 0) {
            $allSpendingZero = false;
            break;
        }
    }
    return $allSpendingZero ? null : false;
}
```

Calculate()

Ez a függvény végzi el a valuta átváltást (a CurrencyBeacon currency api segítségével), megkapjuk a nézettől (view), hogy melyik valutát szeretnénk átváltani, illetve, hogy melyikbe majd meghívjuk az exchangeRate függvényt, az ExchangeRate class-ból, amit a Laravel exchange Rates packageből használunk (<https://github.com/ash-jc-allen/laravel-exchange-rates>).

```
$from = Currency::find($request->currency_id)->code;
$fromSymbol = Currency::find($request->currency_id)->symbol;
$to = Currency::find($request->currency_id2)->code;
$toSymbol = Currency::find($request->currency_id2)->symbol;
$exchangeRate = app(ExchangeRate::class)->exchangeRate($from, $to);
```

Ezt az adatot visszaadás előtt 2 tizedesjegyre kerekítjük:

```
'exchangeRate' => round($exchangeRate, 2),
```


Az **index()**, illetve a **calculate()** függvényekben meg csak meghívjuk a többi függvényt és aztán továbbadjuk a nézetnek (view)

```
return view('home', [
    'currencies' => $currencies,
    //user graphs
    'incomeCategoryPrices' => $incomeCategoryPrices,
    'spendingCategoryPrices' => $spendingCategoryPrices,
    //family graph
    'familyIncomes' => $familyIncomes,
    'familySpending' => $familySpending,
    'familymembers' => $familymembers,
    //calendar
    'userMonthlyFinances' => $userMonthlyFinances,
    'financeColors' => $financeColors,
    'familyFinances' => $familyFinances ?? null,
    'familyFinanceColors' => $familyFinanceColors ?? null,
    'allIncomesZero' => $allIncomesZero ?? '',
    'allSpendingZero' => $allSpendingZero ?? ''
]);
```

```
return view('home', [
    'exchangeRate' => round($exchangeRate, 2),
    'toSymbol' => $toSymbol,
    'fromSymbol' => $fromSymbol,
    'currencies' => $currencies,
    //graphicon data
    'incomeCategoryPrices' => $incomeCategoryPrices,
    'spendingCategoryPrices' => $spendingCategoryPrices,
    'familyIncomes' => $familyIncomes,
    'familySpending' => $familySpending,
    'familymembers' => $familymembers,
    'allIncomesZero' => $allIncomesZero ?? '',
    'allSpendingZero' => $allSpendingZero ?? '',
    //calendar
    'userMonthlyFinances' => $userMonthlyFinances,
    'financeColors' => $financeColors,
    'familyFinances' => $familyFinances ?? null,
    'familyFinanceColors' => $familyFinanceColors ?? null
]);
```

FinanceController működése:

A FinanceController csinálja meg a kiadás és a bevétel felvételét az adatbázisba.

Store függvény:

Ha valamit üresen hagytál akkor visszaküld, egyébként meg létrehozza a bevételt/kiadást a megadott adatok alapján, illetve a mostani dátumot írja bele. Ha bejelölted, hogy havinak szeretnéd, akkor a havi (Monthly) táblába is elmenti, majd visszaküld arra az oldalra, amit hozzáadtál.

```
public function store(StoreFinanceRequest $request)
{
    $bool = str_contains(url()->previous(), 'spending');
    if($request->category_id == null || $request->name == null || $request->price == null)
    {
        toastr()->warning("Please fill in each field");
        return back();
    }

    $finance = Finance::create([
        'user_id' => auth()->user()->id,
        'type' => $bool ? 'Spending' : 'Income',
        'name' => $request->name,
        'price' => $request->price,
        'time' => date("Y/m/d") . '-' . date("H:i:s"),
        'category_id' => $request->category_id,
        'currency_id' => auth()->user()->currency_id
    ]);
    $finance->save();
    if($request->monthly){
        Monthly::create([
            'finance_id' => $finance->id,
            'year' => date("Y"),
            'month' => date("m")
        ]);
    }
    if($bool){
        return redirect()->route('spending');
    }
    else{
        return redirect()->route('income');
    }
}
```

IncomeController működése:

Amikor hozzáadunk bevételt (income) akkor a FinanceController átirányít minket az income pagere, az IncomeController felelős az oldalon kirajzolódó összesítő grafikonokért, illetve a JQuery table-ért, ami lehetőséget ad a hozzáadott bevételek módosítására és törlésére.

Kiegészítő függvények:

addCategory(Request \$request)

Ennek a függvénynek a segítségével hozatsz létre új kategóriát az incomeCreate oldalon.

```
public function addCategory(Request $request)
{
    $request->validate([
        'new_category' => 'required|string|max:255',
    ]);

    $category = new Category();
    $category->name = $request->input('new_category');
    $category->owner_id = auth()->id();
    $category->type = 'income';
    $category->save();

    toastr()->success($category->name . ' has been added to the categories successfully');
    return back();
}
```

create()

A függvény az incomeCreate nézetnek (viewnak) adja át azokat a kategóriákat, amelyeknek az owner_id-ja 0, vagy a te felhasználó id-jával megegyezik és income (bevétel) típusú.

```
public function create(){
    $user = Auth::user();
    $availableCategories = Category::where(function($query) use ($user) {
        $query->where('owner_id', 0)
            ->where('type', 'income');
    })
    ->orWhere(function($query) use ($user) {
        $query->where('owner_id', $user->id)
            ->where('type', 'income');
    })
    ->get();
    return view('includes.incomeCreate',[
        'categories' => $availableCategories,
        'currency' => $user->currency->symbol]);
}
```

editIncomeValue(Request \$request)

Ez a függvény a táblázat (jQuery table), módosításáért felelős. Itt megkapjuk a táblázat sorát, oszlopát és a kiadásunk id-ját. Megkeressük az adatbázisból azt az adatot, majd rámentünk és egy json választ küldünk vissza, hogy sikerült (success), illetve egy refresh true-t, ami az oldal újra töltéséért felelős.


```

public function editIncomeValue(Request $request)
{
    $request->validate([
        'row' => 'required',
        'column' => 'required',
        'value' => 'required',
    ]);

    $id = $request->row;
    $column = $request->column;
    $value = $request->value;

    $finance = Finance::findOrFail($id);
    $finance->$column = $value;
    $finance->save();

    return response()->json(['success' => true, 'refresh' => true]);
}

```

deleteFiance(\$id)

A függvény a táblázatból való törlést végzi el id alapján.

```

public function deleteFinance($id)
{
    $deleteFinanceById = Finance::where('id', '=', $id)->delete();
    toastr()->success("Finance record deleted successfully");
    return back();
}

```

Index()

Az index metódus az income oldal, szükséges adatainak átadásáért, illetve a szükséges nézet átadását végzi. Lekérjük azokat a kiadásokat a \$finances változóba, amelyek a bejelentkezett felhasználóhoz tartoznak, ezenkívül azokat a kategóriákat, amelyeknek az owner_idja 0, vagy a bejelentkezett felhasználó id-jával megegyezik (vagyis a felhasználó adta hozzá külön a kategóriát) és a userhez tartozó valuta szimbólumát, hogy megjeleníthessük az oldalon.

```

public function index()
{
    $userId = Auth::id();
    $user = Auth::user();
    $finances = Finance::where('type', 'Income')->where('user_id', $userId)->get();
    $categories = Category::where(function($query) use ($user) {
        $query->where('owner_id', 0)
            ->where('type', 'income');
    })
    ->orWhere(function($query) use ($user) {
        $query->where('owner_id', $user->id)
            ->where('type', 'income');
    })->get();
    $user = auth()->user();
    $currencySymbol = $user->currency->symbol;
    return view('includes.income', [
        'finances' => $finances,
        'categories' => $categories,
        'currencySymbol' => $currencySymbol,
    ]);
}

```

A SpendingController működése ugyan ezen az elven működik azt leszámítva, hogy bevétel (income), helyett kiadással (spending) működik.

RestoreAccountController működése:

A settings oldalon a felhasználóknak lehetőségük van a fiókjuk felfüggesztésére (soft delete), amit bármikor visszaállíthat.

A RestoreAccountController felel azért, hogy a deaktivált felhasználó visszaállíthassa a fiókját.

Függvények:

- **restoreAccountIndex()**

Itt csak megmutatjuk a accountRestore oldalt, ahol az email cím beírásával és a gomb megnyomásával történik meg a visszaállítás.

```

public function restoreAccountIndex()
{
    return view('includes.accountRestore');
}

```

- **checkIfUserDisabled()**

Ez a függvény az ajax call számára lett létrehozva, az accountRestore oldalon oninput meghívódik a checkIfUserDisabled ajax function, ami megnézi, hogy a beírt email alapján a fiókod létezik-e és fel van-e függesztve, ha nem akkor egy response üzenetet küld vissza, hogy a fiókod nem található vagy nincs felfüggesztve.

```
public function checkIfUserDisabled(Request $request)
{
    $email = $request->input('email');

    $deletedUser = User::where('email', $email)->withTrashed()->exists();
    if(!$deletedUser){
        return response()->json([
            'status' => 'failed',
            'message' => 'Your account is not disabled or does not exists'
        ]);
    }
    else
    {
        return response()->json([
            'status' => 'success',
        ]);
    }
}
```

Ha nem létezik a fiókod vagy nincs felfüggesztve, akkor a gombot kikapcsoljuk.

```
function checkIfUserDisabled(input) {
    $.ajax({
        type: 'POST',
        url: '{{ route("checkIfUserDisabled") }}',
        data: {
            '_token': '{{ csrf_token() }}',
            'email': input
        },
        success: function(data) {
            if (data.status === "failed") {
                $('#responseTextEmail').removeClass('text-success').addClass('text-danger').html(data.message);
                $('#reactivateBTN').prop('disabled', true);
            } else if (data.status === "success") {
                $('#responseTextEmail').removeClass('text-danger').html('');
                $('#reactivateBTN').prop('disabled', false);
            }
        },
        error: function(xhr, status, error) {
            console.error("Error: " + status + " " + error);
            $('#responseTextEmail').html('An error occurred. Please try again later.').addClass('text-danger');
        }
    });
}
```

- **reactivateAccount(Request \$request)**

A reactivateAccount függvény az email alapján megkeresi a felhasználót és visszaállítja, ezután kijelentkeztet és a login oldalra küld vissza.

DocumentationController működése:

A controller visszaadja a documentation oldal nézetét.

```
public function index()
{
    return view('includes.documentation');
}
```

LoginController működése:

A LoginController a felhasználók bejelentkeztetéséért szolgáló controller. Alapvetően az alapvető laravel bejelentkeztetést használjuk, de írtunk hozzá egyéb kiegészítéseket. Például az authenticated(Request \$request, \$user) függvény megnézi, hogy van-e a sessionben eltárolt token, akkor átirányítjuk a felhasználót a FamilyController accpetInvitation(\$token) függvényére. Ezenfelül itt végig megyünk az aktuális felhasználó pénzügyein, majd ellenőrizzük, hogy van-e havi kiadása vagy bevétele hozzáadva. Ha van akkor ellenőrizzük, hogy a tranzakció a jelenlegi hónapban lett-e létrehozva, ha nem akkor kiszámolja, hogy hány hónapot kell hozzáadnia a jelenlegi dátumhoz, majd létrehoz egy vagy több tranzakciót a korábbiak alapján, ezután frissíti a havi tranzakciók létrehozásának dátumát a jelenlegi hónapra és évre.

```

foreach (auth()->user()->finances as $key => $value) {
    $monthly = $value->monthly;
    //dd($monthly);
    if($monthly!=null){
        //dd($monthly);
        if($monthly->month!=date('m')||$monthly->year!=date('Y')){
            $years=date('Y')-$monthly->year;
            $times=0;
            if($years >1){
                $times=12*($years-1)+(12-$monthly->month)+date('m');
            }
            else if($years==1){
                $times=12-$monthly->month+date('m');
            }
            else{
                $times=date('m')-$monthly->month;
            }
            for ($i=0; $i < $times; $i++) {
                $monthly->month++;
                if($monthly->month>12){
                    $monthly->month=1;
                    $monthly->year++;
                }
                if($monthly->month>9){
                    $date=$monthly->year.'-'. $monthly->month.'-'.date("d");
                }
                else{
                    $date=$monthly->year.'-'. '0'.'$monthly->month.'-'.date("d");
                }
                $date=date('Y/m/d',strtotime($date));
                $finance=Finance::create([
                    'user_id'=> auth()->user()->id,
                    'type' => $value->type,
                    'name'=> $value->name,
                    'price' => $value->price,
                    'time' => $date,
                    'category_id'=> $value->category_id,
                    'currency_id' =>$value->currency_id,
                ]);
                $finance->save();
            }
            $monthly->year = date('Y');
            $monthly->month = date('m');
            $monthly->save();
        }
    }
}

```

FamilyController működése:

Ez a controller a család létrehozását végzi, vagyis, amikor a SettingsController elküldi a meghívó kódot (invitation email) ez a controller ellenőrzi le, hogy a meghívó token amit az adatbázisba eltároltunk megegyezik-e.

- acceptInvitation(\$token)

Lekérjük a LoginController által átadott token a sessionból, ha nem egyezik meg, akkor visszaküld a főmenüre az 'Invalid invitation token' hibaüzenettel. Amennyiben a meghívó státusza 'pending' vagyis folyamatban, akkor ugyan úgy visszaküldjük a felhasználót, csak a hibaüzenet fog megváltozni ('This invitation has been processed'). Amennyiben nincs bejelentkezve a felhasználó a sessionbe eltároljuk a token és login

oldalra irányítjuk a felhasználót. Megnézzük, hogy a felhasználó már családtag-e és ha igen akkor visszaküldjük a főmenüre egy hibaüzenettel. Illetve mentjük az adatokat és az meghívó (invitation) státuszát elfogadottra (acceptedre) állítjuk.

```
public function acceptInvitation($token)
{
    $invitation = FamilyInvitations::where('token', $token)->first();

    if (!$invitation) {
        // No invitation found with this token
        return redirect('/')->with('error', 'Invalid invitation token.');
```

```
    }

    if ($invitation->status !== 'pending') {
        // Invitation is not in a pending state
        return redirect('/')->with('error', 'This invitation has already been processed.');
```

```
    }

    // Check if the user is logged in
    if (!Auth::check()) {
        // Store the token in the session to use after login
        session(['invitation_token' => $token]);

        // Redirect to login page with a callback to this route after login
        return redirect()->route('login')->with('info', 'Please log in to accept the invitation.');
```

```
    }

    $user = Auth::user();
    if ($user->family_id) {
        // User is already part of a family
        return redirect('/')->with('error', 'You are already part of a family.');
```

```
    }

    $user->family_id = $invitation->family_id;
    $user->save();

    // Update the invitation to mark it as accepted
    $invitation->status = 'accepted';
    $invitation->save();

    // Redirect with a success message
    toastr()->success('You have successfully joined the family');
    return redirect('/');
```

RegisterController működése:

A RegisterController a megtörténik a regisztrációs oldalon (nézetén) történő adatok validálás a teljes névtől a jelszóig. Egyedül a telefonszám az, amit nem muszáj megadnia a felhasználónak. Ha van telefonszám átadva a regisztrációnál, akkor eltávolítjuk a szükségtelen karaktereket (nem numerikus). A megadott adatokkal létrehozuk a felhasználót és elküldünk neki egy “köszönjük a regisztrációt” emailt.

Kiegészítő függvények:

- **create()**

```
protected function create(array $data)
{
    $roles_id = 3; // basic role, 2 pro, 1 dev
    if($data['phone']!=null){
        $phone = $data['phone'];
        $numbers="";
        $firstchar=true;
        $data['phone']=null;
        for ($i=0; $i < strlen($phone); $i++) {
            if($i!=0){
                if(is_numeric($phone[$i])){
                    $numbers.=strval($phone[$i]);
                }
            }
            else{
                if(is_numeric($phone[$i]) || $phone[$i]=='+'){
                    $numbers.=strval($phone[$i]);
                    $firstchar = false;
                }
            }
        }
        $data['phone']=$numbers;
        //@dd($data['phone']);
    }

    $user = User::create([
        'fullname' => $data['fullname'],
        'username' => $data['username'],
        'phone' => $data['phone'] ?: null,
        'email' => $data['email'],
        'password' => Hash::make($data['password']),
        'roles_id' => $roles_id,
        'currency_id' => $data['currency_id']
    ]);

    Mail::to($data['email'])->send(new RegistrationSuccessful($data['username']));
    return $user;
}
```

- **showregistrationForm()**

A függvény átadja az összes valutát a Currency Modelből, majd átadja a regisztrációs oldalnak, hogy ott kilistázhassuk őket, amiből a user választhat a regisztrációkor.

```
public function showRegistrationForm()
{
    $currencies = Currency::all();
    return view('auth.register', compact('currencies'));
}
```

- **checkNameIsTaken(Request \$request), checkEmailIsTaken(Request \$request)**

Ez a két ajax funkció, ami változás után (oninput) megnézi, hogy az adott felhasználónév, illetve email foglalt-e, és ha igen, akkor visszaad egy 'failed' státuszt,

ezenfelül egy üzenete, hogy a felhasználónév vagy email foglalt már. Ha foglalt, akkor blokkolja a regisztrációs gombot, de ha valaki átírja a html-t akkor a validációnál le van kezelve ez az eshetőség is.

A felhasználónév foglalt.

Register

Fullname

Username

Username is already taken

Phone (optional)

Currency

Email Address

Password

Confirm Password

Register


Az email foglalt.

Register

Fullname

Username

Phone (optional)



(201) 555-0123

Currency

Please Choose...

Email Address

mzboncak@example.com

Email is already taken

Password

Confirm Password

Register

- **calculateEntropyScore(Request \$request)**

A jelszó alapján, amit a calculateEntropy(Request \$request) funkcióból kap meg kiszámolja a jelszavunk entrópiáját (a jelszó véletlenszerűségét és bonyolultságát méri). A képlete a következő.

$$E = \log_2(R^L)$$

A karakter készletet emeljük a jelszóhosszának hatványára, majd ennek vesszük a 2-es alapú logaritmusát.

```

private function calculateEntropyScore($password)
{
    $charset = 0;

    if (preg_match('/[a-z]/', $password)) {
        $charset += 26;
    }
    if (preg_match('/[A-Z]/', $password)) {
        $charset += 26;
    }
    if (preg_match('/[0-9]/', $password)){
        $charset += 10;
    }
    if (preg_match('/^[a-zA-Z0-9]/', $password)) {
        $charset += 30;
    }
    $entropy = round(log(pow($charset, strlen($password)),2));

    return $entropy;
}

```

Majd átadja a calculateEntropy függvénynek, ami továbbítja az ajaxnak, és az entropyja alapján írja ki alatta a jelszó erősséget.

```

public function calculateEntropy(Request $request)
{
    $password = $request->input('password');

    $entropy = $this->calculateEntropyScore($password);
    return response()->json(['entropy' => $entropy]);
}

```

A gyenge jelszó piros, a közepes jelszó sárga, az erős jelszó kék, és a nagyon erős pedig zöld.

```

/* Passw is strong enough or not */
document.getElementById('password').addEventListener('input', function() {
    var password = this.value;
    $.ajax({
        type: 'POST',
        url: '/calculate-entropy',
        data: {
            password: password,
            _token: '{{ csrf_token() }}',
        },
        success: function(response) {
            var strength;
            var colorClass;
            switch (true) {
                case (response.entropy <= 35):
                    strength = 'Weak';
                    colorClass = 'text-danger';
                    break;
                case (response.entropy >= 36 && response.entropy <= 59):
                    strength = 'Moderate';
                    colorClass = 'text-warning';
                    break;
                case (response.entropy >= 60 && response.entropy <= 119):
                    strength = 'Strong';
                    colorClass = 'text-primary';
                    break;
                case (response.entropy >= 120):
                    strength = 'Very Strong';
                    colorClass = 'text-success';
                    break;
                default:
                    strength = 'Something went wrong';
                    colorClass = 'text-dark';
                    break;
            }
            var passwordStrengthElement = document.getElementById('passwordStrength');
            passwordStrengthElement.innerText = 'Password Strength: ' + strength;
            passwordStrengthElement.className = colorClass;
        },
        error: function(xhr, textStatus, errorThrown) {
            console.log("Error:", errorThrown);
        }
    });
});

```

ForgotPasswordController működése:

A controller az elfelejtett jelszó visszaállítására, megváltoztatására hivatott.

- **show ForgetPasswordForm()**

A funkció a jelszóvisszaállító oldal nézetét hivatott átadni.

```

public function showForgetPasswordForm()
{
    return view('auth.showForgetPassword');
}

```

- **submitForgetPasswordForm(Request \$request)**

Levalidáljuk az emailt, majd legenerálunk egy 64 karakter hosszú token-t, ezután a `password_reset_tokens` táblába beszúrunk egy új sort, az emaillel, a tokennel és a created_at-hez a mostani dátumot, majd a tokenet csatolva kiküldönk egy emailt.

```

public function submitForgetPasswordForm(Request $request)
{
    $request->validate([
        'email' => 'required|email|exists:users',
    ]);

    $token = Str::random(64);

    DB::table('password_reset_tokens')->insert([
        'email' => $request->email,
        'token' => $token,
        'created_at' => Carbon::now()
    ]);

    Mail::send('email.forgetPassword', ['token' => $token], function ($message) use ($request) {
        $message->to($request->email);
        $message->subject('Reset Password');
    });

    toastr()->info('We have emailed your password reset link!');
    return back();
}

```

- **submitResetPasswordForm(Request \$request)**

Ez a függvény fogadja és validálja a jelszó visszaállítási űrlap által küldött adatokat, ellenőrzi a jelszó visszaállítási token érvényességét, majd, ha az érvényes, frissíti a felhasználó jelszavát a megadott új jelszóra, és törli az összes jelszó visszaállítási token a felhasználóhoz tartozóan az újrafelhasználás megakadályozása érdekében. Ezután sikeres jelszóváltozás üzenetet jelenít meg, és átirányítja a felhasználót a bejelentkező oldalra.

```

public function submitResetPasswordForm(Request $request)
{
    $request->validate([
        'email' => 'required|email|exists:users',
        'password' => 'required|string|min:8|confirmed',
    ]);

    if (!$request->has('token')) {
        return back()->withInput()->with('error', 'Invalid token 1 !');
    }

    // Retrieve all tokens for the user's email
    $tokens = DB::table('password_reset_tokens')
        ->where('email', $request->email)
        ->get();

    $tokenFound = false;
    foreach ($tokens as $tokenRecord) {
        if (Hash::check($request->token, $tokenRecord->token)) {
            $tokenFound = true;
            break; // Exit the loop if a matching token is found
        }
    }

    if (!$tokenFound) {
        return back()->withInput()->with('error', 'Invalid token 2 !');
    }

    // Token is valid, proceed with password update
    User::where('email', $request->email)
        ->update(['password' => Hash::make($request->password)]);

    // Delete all reset tokens for this user to prevent reuse
    DB::table('password_reset_tokens')->where('email', $request->email)->delete();

    toastr()->success('Your password has been changed');
    return redirect('/login');
}

```

AboutUsController működése:

Ez a controller az about us oldal megjelenítését végzi.

- **index()**

```
public function index()
{
    return view('includes.about_us');
}
```

MonthlyController működése:

A controller a havi kiadás/bevétel hozzáadásáért, és törléséért felelős.

- **store(\$id)**

Havi pénzügy hozzáadása id alapján.

```
public function store($id)
{
    Monthly::create([
        'finance_id' => $id,
        'year' => date("Y"),
        'month' => date("m")
    ]);
    return back();
}
```

- **deleteMonthly(\$id)**

Törlés id- alapján.

```
public function deleteMonthly($id)
{
    $deleteFinanceById = Monthly::where('finance_id', '=', $id)->delete();
    //toastr()->success("Monthly record deleted successfully");
    return back();
}
```

SettingsController működése:

A settingscontroller lehetőséget ad a felhasználó adatainak megváltoztatására, családtagok hozzáadására, törlésére (kezelésére).

Funkciók: index(), changeFullName(), changeUserName(), changeEmail(), changeCurrency(), enableNotification(), changePhone(), createFamily(), checkIfUserExists(Request \$request), deleteFamily(), deleteFamilyMember(\$id), addFamilymember(Request \$request), leaveFamily(), softDeleteAccount()

- changeFullName(Request \$request)

A validációnál megadjuk, hogy egyedinek kell lenni. A régi \$fullname változót felülírjuk az újra.

```
public function changeFullName(Request $request)
{
    $request->validate([
        'newFullname' => 'required|string|max:255|min:5|unique:users,fullname'
    ]);

    $user = Auth::user();

    $user->fullname = $request->newFullname;
    $user->update();
    toastr()->success("Fullname changed successfully");
    return back();
}
```

A changeUserName, changeEmail, changePhone ugyanezen az elven működik.

A checkIfUserExist egy ajax függvény, ami csak akkor enged rányomni az invite family member buttonra, ha a felhasználó létezik az adatbázisban, illetve nincsen már hozzáadva

```

public function checkIfUserExists(Request $request)
{
    $username = $request->input('username');
    $userFound = User::where('username', $username)->exists();
    if (!$userFound) {
        return response()->json([
            'status' => 'failed',
            'message' => 'User not found',
        ]);
    }

    $userFamilyId = auth()->user()->family_id;
    $inviteduser = User::where('username', $username)
        ->where('family_id', $userFamilyId)
        ->whereNotNull('family_id')
        ->first();
    if ($inviteduser) {
        return response()->json([
            'status' => 'failed2',
            'message' => 'User is already in your family',
        ]);
    }

    return response()->json([
        'status' => 'success',
    ]);
}

```

- **createFamily()**

Ha még nincs család létrehozva, abban az esetben a felhasználó teljes nevét eltördeljük (spliteljük) és az eltördelt név változó 0. indexét mentjük el, mint családnév. Ezután a szükséges adatokból létrehozuk a családot.

```

public function createFamily()
{
    $user = auth()->user();
    //we can only have 1 family, so if you already have 1 you can't create
    if($user->family_id == null)
    {
        $userSplittedName = explode(" ", $user->fullname);

        $family = new Family();
        $family->name = $userSplittedName[0];
        $family->creator_Id = $user->id;
        $family->save();
        $user->family_id = $family->id;
        $user->save();
        toastr()->success("Family created");
        return back();
    }
    else {
        toastr()->warning('You already have family created');
        return back();
    }
}

```

- **deleteFamily(), deleteFamilyMember(\$id)**

A deleteFamily és a deleteFamilyMember(\$id) a család, illetve a családtag törlését teszi lehetővé a családalapító (családfő) számára.

```

public function deleteFamily()
{
    $user = auth()->user();
    $userFamilyId = $user->family_id;
    $deleteFamily = Family::where('id', '=', $userFamilyId)->delete();
    $deleteFamilyMembers = User::where('family_id', '=', $userFamilyId)->update(['family_id' => null]);

    toastr()->success("Family deleted");
    return back();
}

public function deleteFamilyMember($id)
{
    $deleteFamilyMemberById = User::where('id', '=', $id)->update(['family_id' => null]);
    toastr()->success("Family member deleted successfully");
    return back();
}

```

- **leaveFamily()**

A leaveFamily funkció a családtagok számára elérhető, hogyha el szeretnék hagyni a családot.


```
public function leaveFamily()
{
    $user = auth()->user();
    $leaveFamily = User::where('id', '=', $user->id)->update(['family_id' => null]);
    toastr()->success("You left the family successfully");
    return back();
}
```

- **softDeleteAccount()**

Ez a függvény a felhasználói fiók felfüggesztését végzi el, ha felhasználó a settings (beállítások) oldalon a deactivate account gombra kattint, akkor a program softdeleteli a fiókját, vagyis felfüggeszti, illetve kijelentkezteti.

```
public function softDeleteAccount()
{
    $user = auth()->user();
    toastr()->success("You deactivated your account successfully");
    $user->delete();
    return redirect()->route('home');
}
```

- **changeCurrency()**

A függvény először átváltja a userhez kapcsolódó valutát a kiválasztottra, majd az összes bevételét és kiadását is átváltja arra a valutára. A hosszú tizedesjegyeket elkerülve, 3 tizedesjegyre kerekítjük az árfolyamot.

- **enableNotification()**

Itt tudjuk be, illetve kikapcsolni az értesítéseket. A böngészőből lekéri a javascript az időzónát, majd a notification-t igazra (true-ra) állítja.

```

public function enableNotification(Request $request)
{
    $user = auth()->user();
    if($user->notification == false){
        $userTimezone = $request->input('timezone');
        $user->notification = '1';
        $user->timezone = $userTimezone;
        $user->save();
        toastr()->success('Notifications enabled successfully');
        return back();
    }
    else
    {
        $user->notification = '0';
        $user->timezone = null;
        $user->save();
        toastr()->success('Notifications disabled successfully');
        return back();
    }
}

```

AdvancedStatisticsController működése:

A controller az egyéb statisztikákat mutatja meg, különböző statisztikák jelennek meg attól függően, hogy családdal rendelkezik-e a felhasználó.

- **AvarageWithTheSameCurrency()**

Ez a függvény, amely egy teszt jelleggel létrehozott funkció, ami visszaadja azoknak a felhasználók kiadásának és bevételének átlagát az adott hónapra, amit továbbadunk az indexnek, a handleformnak, illetve a handleFamilyForm funkcióknak. Az összes szükséges függvény meghívódik a fentebb említett függvényeknél, hogy a szükséges adatok megjelenhessenek.

- **CategoriesWithFinance(\$userId)**

Ez a függvény kategorizálja a felhasználó bevételét és kiadásait a rendelkezésre álló kategóriák szerint.

```

function CategoriesWithFinance($userId){
    $spendingCategoriesWithFinance = Category::join('finances', 'categories.id', '=', 'finances.category_id')
    ->where('categories.type', '=', 'spending')
    ->where(function ($query) use ($userId) {
        $query->where('categories.owner_id', '=', 0)
        ->orWhere('categories.owner_id', '=', $userId);
    })
    ->where('finances.user_id', '=', $userId) // Filter by user_id in finances table
    ->select('categories.*')
    ->distinct()
    ->get();

    $incomeCategoriesWithFinance = Category::join('finances', 'categories.id', '=', 'finances.category_id')
    ->where('categories.type', '=', 'income')
    ->where(function ($query) use ($userId) {
        $query->where('categories.owner_id', '=', 0)
        ->orWhere('categories.owner_id', '=', $userId);
    })
    ->where('finances.user_id', '=', $userId)
    ->select('categories.*')
    ->distinct()
    ->get();
    return[
        "spendingCategoriesWithFinance"=>$spendingCategoriesWithFinance,
        "incomeCategoriesWithFinance"=>$incomeCategoriesWithFinance
    ];
}

```

- **categoriesForAuthUserMonthly(\$userId)**

Ez a függvény a felhasználó havi bevételét és kiadásait kategorizálja. Első lépésként megkapja a felhasználó azonosítóját (\$userId), amely alapján lekéri a felhasználó havi bevételét és kiadásait az adatbázisból. Ezután a függvény kategorizálja ezeket a bevétel- és kiadáspozíciókat az alkalmazásban definiált kategóriák szerint. Tehát minden bevételt és kiadást besorol azokba a kategóriákba, amelyekhez tartoznak. Az így kategorizált adatokat tárolja a \$this->incomeCategoriesForAuthUserMonthly és a \$this->spendingCategoriesForAuthUserMonthly változókban, amelyeket később a index() függvényben felhasznál a megfelelő statisztikák megjelenítéséhez a felhasználó számára.

```

function CategoriesForAuthUserMonthly($userId){
    $this->incomeCategoriesForAuthUserMonthly = Category::join('finances', 'categories.id', '=', 'finances.category_id')
    ->where('categories.type', '=', 'income')
    ->where(function ($query) use ($userId) {
        $query->where('categories.owner_id', '=', 0)
        ->orWhere('categories.owner_id', '=', $userId);
    })
    ->where('finances.user_id', '=', $userId)
    ->whereMonth(['finances.time', '=', date('m')])
    ->whereYear('finances.time', '=', date('Y'))
    ->select('categories.*')
    ->distinct()
    ->get();

    $this->spendingCategoriesForAuthUserMonthly = Category::join('finances', 'categories.id', '=', 'finances.category_id')
    ->where('categories.type', '=', 'spending')
    ->where(function ($query) use ($userId) {
        $query->where('categories.owner_id', '=', 0)
        ->orWhere('categories.owner_id', '=', $userId);
    })
    ->where('finances.user_id', '=', $userId)
    ->whereMonth('finances.time', '=', date('m'))
    ->whereYear('finances.time', '=', date('Y'))
    ->select('categories.*')
    ->distinct()
    ->get();
}

```

- **FamilyInfos()**

Ez a függvény gyűjti össze és számolja ki a család összegzett bevételét és kiadásait, valamint a család tagjainak azonos valutájú összbevételét és összkiadását. Családtagok Keresése: Először is, a függvény lekérdezi az összes családtagot az adatbázisból, akik ugyanahhoz a családhoz tartoznak, mint a bejelentkezett felhasználó. Valuták Kigyűjtése: A függvény minden családtaghoz rendeli a hozzájuk tartozó valutát. Arra az esetre, hogyha esetleg más valutát használnának egyes családtagok. Összegezés: A függvény végigmegy az összes családtagon, és összegezi az egyes tagok bevételét és kiadását. Ez megtörténik a pénzügyi tranzakciók lekérdezésével és azok összegzésével, amelyeket a családtag hozott létre.

Összegzett Statisztikák Visszaadása: A függvény visszaadja a család összegzett bevételét és kiadásait, valamint minden családtag azonos valutájú összbevételét és összkiadását. Ezek az adatok segítenek abban, hogy a család együttműködően és átláthatóan kezelje pénzügyeit, és megértsék, hogy mennyit költenek és mennyit kereshetnek a családtagok.

```
function FamilyInfos(){
  $familyMembers = User::where('family_id', auth()->user()->family_id)
    ->whereNotNull('family_id')
    ->get();
  $memberIncome = 0;
  $memberSpending = 0;
  $familyCurrencySymbols = [];
  foreach ($familyMembers as $member) {
    $familyCurrencySymbols[$member->id] = $member->currency->symbol;
  }
  $totalIncome = 0;
  $totalSpending = 0;
  foreach($familyMembers as $member)
  {
    $userFinances = Finance::where('user_id', $member->id)->get();

    $memberIncome = $userFinances->where('type', 'Income')->sum('price');
    $memberSpending = $userFinances->where('type', 'Spending')->sum('price');

    $totalIncome += $memberIncome;
    $totalSpending += $memberSpending;
  }
  return[
    "members"=>$familyMembers,
    "symbol"=>$familyCurrencySymbols,
    "sincome"=>$memberIncome,
    "sspending"=>$memberSpending,
    "tincome"=>$totalIncome,
    "tspending"=>$totalSpending
  ];
}
```

Views (Nézetek) funkciói

```
/* Bg image */
document.body.style.backgroundImage = "url('../storage/pictures/registerPic.jpg')";
```

Ahol a háttérkép eltérő, ott ez a funkció változtatja meg a háttérképet

register.blade.php

```
/* Phone num input field masking */
const input = document.querySelector("#phone");
window.intlTelInput(input, {
  initialCountry: "auto",
  geoIpLookup: callback => {
    fetch("https://ipapi.co/json")
      .then(res => res.json())
      .then(data => callback(data.country_code))
      .catch(() => callback("us"));
  },
  strictMode: true,
  utilsScript: "https://cdn.jsdelivr.net/npm/intl-tel-input@21.1.1/build/js/utils.js",
});
```

Ez a funkció a telefonszám beviteli mező kinézetéért és viselkedéséért felel.

```
/* Passw is strong enough or not */
document.getElementById('password').addEventListener('input', function() {
  var password = this.value;
  $.ajax({
    type: 'POST',
    url: '/calculate-entropy',
    data: {
      password: password,
      _token: '{{ csrf_token() }}',
    },
    success: function(response) {
      var strength;
      var colorClass;
      switch (true) {
        case (response.entropy <= 35):
          strength = 'Weak';
          colorClass = 'text-danger';
          break;
        case (response.entropy >= 36 && response.entropy <= 59):
          strength = 'Moderate';
          colorClass = 'text-warning';
          break;
        case (response.entropy >= 60 && response.entropy <= 119):
          strength = 'Strong';
          colorClass = 'text-primary';
          break;
        case (response.entropy >= 120):
          strength = 'Very Strong';
          colorClass = 'text-success';
          break;
        default:
          strength = 'Something went wrong';
          colorClass = 'text-dark';
          break;
      }
      var passwordStrengthElement = document.getElementById('passwordStrength');
      passwordStrengthElement.innerText = 'Password Strength: ' + strength;
      passwordStrengthElement.className = colorClass;
    },
    error: function(xhr, textStatus, errorThrown) {
      console.log("Error:", errorThrown);
    }
  });
});
```

Ez a funkció levizsgálja és visszajelzést küld a felhasználónak, hogy elég erős-e a jelszava.

```
function checkEmailTaken(input) {  
  $.ajax({  
    type: 'POST',  
    url: '/checkEmailTaken',  
    data: {  
      '_token': '{{ csrf_token() }}',  
      'email': input  
    },  
    success: function(data) {  
      console.log(data);  
      if (data.status == "failed") {  
        $('#responseTextEmail').removeClass('text-danger text-success')  
        $('#responseTextEmail').html(data.message);  
        $('#responseTextEmail').addClass('text-danger');  
        $('#logButton').prop('disabled', true);  
      } else if (data.status == "success") {  
        $('#responseTextEmail').removeClass('text-danger').html('');  
        $('#logButton').prop('disabled', false);  
      }  
    },  
    error: function(xhr, status, error) {  
      console.error("Error: " + status + " " + error);  
      $('#responseText').html('An error occurred. Please try again later.').addClass(  
        'text-danger');  
    }  
  });  
}
```

Ez a funkció levizsgálja és visszajelzést küld a felhasználónak, hogy foglalt-e már az email címe.

```

/* Check username */
function checkUsernameTaken(input) {
    $.ajax({
        type: 'POST',
        url: '/checkUsernameTaken',
        data: {
            '_token': '{{ csrf_token() }}',
            'username': input
        },
        success: function(data) {
            if (data.status == "failed") {
                $('#responseText').removeClass('text-danger text-success')
                $('#responseText').html(data.message);
                $('#responseText').addClass('text-danger');
                $('#logButton').prop('disabled', true);
            } else if (data.status == "success") {
                $('#responseText').removeClass('text-danger').html('');
                $('#logButton').prop('disabled', false);
            }
        },
        error: function(xhr, status, error) {
            console.error("Error: " + status + " " + error);
            $('#responseText').html('An error occurred. Please try again later.').addClass(
                'text-danger');
        }
    });
}

```

Ez a funkció levizsgálja és visszajelzést küld a felhasználónak, hogy foglalt-e már az felhasználó neve.

about_us.blade.php

```

<script>
  const wrapper = document.querySelector(".wrapper");
  const carousel = document.querySelector(".carousel");
  const firstCardWidth = carousel.querySelector(".card").offsetWidth;
  const arrowBtns = document.querySelectorAll(".wrapper i");
  const carouselChildrens = [...carousel.children];
  let isDragging = false,
      isAutoPlay = true,
      startX, startScrollLeft, timeoutId;
  // Get the number of cards that can fit in the carousel at once
  let cardPerView = Math.round(carousel.offsetWidth / firstCardWidth);
  // Insert copies of the last few cards to beginning of carousel for infinite scrolling
  carouselChildrens.slice(-cardPerView).reverse().forEach(card => {
    carousel.insertAdjacentHTML("afterbegin", card.outerHTML);
  });
  // Insert copies of the first few cards to end of carousel for infinite scrolling
  carouselChildrens.slice(0, cardPerView).forEach(card => {
    carousel.insertAdjacentHTML("beforeend", card.outerHTML);
  });
  // Scroll the carousel at appropriate postition to hide first few duplicate cards on Firefox
  carousel.classList.add("no-transition");
  carousel.scrollLeft = carousel.offsetWidth;
  carousel.classList.remove("no-transition");
  // Add event listeners for the arrow buttons to scroll the carousel left and right
  arrowBtns.forEach(btn => {
    btn.addEventListener("click", () => {
      carousel.scrollLeft += btn.id == "left" ? -firstCardWidth : firstCardWidth;
    });
  });
  const dragStart = (e) => {
    isDragging = true;
    carousel.classList.add("dragging");
    // Records the initial cursor and scroll position of the carousel
    startX = e.pageX;
    startScrollLeft = carousel.scrollLeft;
  }

```



```

const dragging = (e) => {
  if (!isDragging) return; // if isDragging is false return from here
  // Updates the scroll position of the carousel based on the cursor movement
  carousel.scrollLeft = startScrollLeft - (e.pageX - startX);
}

const dragStop = () => {
  isDragging = false;
  carousel.classList.remove("dragging");
}

const infiniteScroll = () => {
  // If the carousel is at the beginning, scroll to the end
  if (carousel.scrollLeft === 0) {
    carousel.classList.add("no-transition");
    carousel.scrollLeft = carousel.scrollWidth - (2 * carousel.offsetWidth);
    carousel.classList.remove("no-transition");
  }
  // If the carousel is at the end, scroll to the beginning
  else if (Math.ceil(carousel.scrollLeft) === carousel.scrollWidth - carousel.offsetWidth) {
    carousel.classList.add("no-transition");
    carousel.scrollLeft = carousel.offsetWidth;
    carousel.classList.remove("no-transition");
  }
  // Clear existing timeout & start autoplay if mouse is not hovering over carousel
  clearTimeout(timeoutId);
  if (!wrapper.matches(":hover")) autoplay();
}

const autoPlay = () => {
  if (window.innerWidth < 800 || !isAutoPlay)
    return; // Return if window is smaller than 800 or isAutoPlay is false
  // Autoplay the carousel after every 2500 ms
  timeoutId = setTimeout(() => carousel.scrollLeft += firstCardWidth, 2500);
}

autoplay();
carousel.addEventListener("mousedown", dragStart);
carousel.addEventListener("mousemove", dragging);
document.addEventListener("mouseup", dragStop);
carousel.addEventListener("scroll", infiniteScroll);
wrapper.addEventListener("mouseenter", () => clearTimeout(timeoutId));
wrapper.addEventListener("mouseleave", autoPlay);
</script>

```

Minden, ami a `<script>` és `</script>` “tag” -ek között található, az a reklámkártyák animációjáért és interaktivitásáért felel.

advancedStatistics.blade.php

```

<div class="card-body">
  @php
    $totalIncome = 0;
    $totalSpending = 0;
    $currentMonth = date('m');
  @endphp
  @foreach ($incomes as $income)
    @if (substr($income->time, 5, 2) == $currentMonth)
      @php
        $totalIncome += $income->price;
      @endphp
    @endif
  @endforeach
  @foreach ($spendings as $spend)
    @if (substr($spend->time, 5, 2) == $currentMonth)
      @php
        $totalSpending += $spend->price;
      @endphp
    @endif
  @endforeach
  @php
    $availableBalance = $totalIncome - $totalSpending;
  @endphp
  <h1 class="available-balance">Available Balance: {{ $availableBalance }} {{ $currencySymbol }}
</h1>
</div>

```

Ez a funkció a megmaradt egyenleget számolja ki az eddigi kiadás és bevétel alapján.

```

@if (!auth()->user()->family || (auth()->user()->family && $familyMembers->count() == 1))
  {{-- If user dont have a family or user's family doesnt have members except the user --}}
  <div class="specialcard-container">
    @php
      $currentYear = date('Y');
      $monthsToShow = [$currentMonth - 2, $currentMonth - 1, $currentMonth];
      $monthsLabels = [
        date('F', mktime(0, 0, 0, $currentMonth - 2, 1, $currentYear)), // Two months ago
        date('F', mktime(0, 0, 0, $currentMonth - 1, 1, $currentYear)), // One month ago
        date('F', mktime(0, 0, 0, $currentMonth, 1, $currentYear)), // Current month
      ];
    @endphp
    @for ($i = 0; $i < 3; $i++)
      <div class="specialcard bg-dark text-light">
        <div class="specialcard-body">
          <h5 class="specialcard-title">{{ $monthsLabels[$i] }}</h5>
          @php
            $totalIncome = 0;
            $totalSpending = 0;
          @endphp
          @foreach ($incomes as $income)
            @if (date('m', strtotime($income->time)) == $monthsToShow[$i] && date('Y', strtotime($income->time)) == $currentYear)
              @php
                $totalIncome += $income->price;
              @endphp
            @endif
          @endforeach

          @foreach ($spendings as $spend)
            @if (date('m', strtotime($spend->time)) == $monthsToShow[$i] && date('Y', strtotime($spend->time)) == $currentYear)
              @php
                $totalSpending += $spend->price;
              @endphp
            @endif
          @endforeach
          <p class="specialcard-text">Total Income:<br>{{ $totalIncome }} {{ $currencySymbol }}</p>
          <hr>
          <p class="specialcard-text">Total Spending:<br>{{ $totalSpending }} {{ $currencySymbol }}</p>
        </div>
      </div>
    @endfor
  </div>
@else

```

Ez a funkció három havi összegzett bevételt és kiadást jelenít meg hónapokra bontva.

```

<div class="specialcard-container">
  @php
    $currentMonth = date('m');
  @endphp
  @php
    $currentYear = date('Y');
    $monthsToShow = [$currentMonth - 2, $currentMonth - 1, $currentMonth];
    $monthsLabels = [
      date('F', mktime(0, 0, 0, $currentMonth - 2, 1, $currentYear)), // Two months ago
      date('F', mktime(0, 0, 0, $currentMonth - 1, 1, $currentYear)), // One month ago
      date('F', mktime(0, 0, 0, $currentMonth, 1, $currentYear)), // Current month
    ];
  @endphp
  @foreach ($monthsToShow as $index => $month)
    <div class="specialcard h-130 bg-dark text-light">
      <div class="specialcard-body">
        <h5 class="specialcard-title">{{ $monthsLabels[$index] }}</h5>
        @foreach ($familyMembers as $member)
          @php
            $memberCurrencySymbol = $familyCurrencySymbols[$member->id];
          @endphp
          @php
            $userFinances = \App\Models\Finance::where('user_id', $member->id)
              ->whereMonth('time', $month)
              ->whereYear('time', $currentYear)
              ->get();
            $totalIncome = $userFinances->where('type', 'Income')->sum('price');
            $totalSpending = $userFinances->where('type', 'Spending')->sum('price');
          @endphp
          <p class="specialcard-text">Family member:<br>
            @if ($member->username == auth()->user()->username)
              <u> My account </u> @else<u>{{ $member->username }}</u>
            @endif
          </p>
          <p class="specialcard-text">Total Income:<br>{{ $totalIncome }} {{ $memberCurrencySymbol }}
          </p>
          <p class="specialcard-text">Total Spending:<br>{{ $totalSpending }}
            {{ $memberCurrencySymbol }}
          </p>
          <hr>
        @endforeach
      </div>
    </div>
  @endforeach
</div>

```

Amennyiben van család létrehozva és van hozzá adva legalább 1 személy a következő funkció fog működni az előző helyett.

```

{{-- SELECT BY CATEGORY --}}
<div class="container text-center mb-3 mt-3">
  <div class="row">
    <div class="col-md-3"></div>
    <div class="col-md-6">
      @if (auth()->user()->finances()->where('type', 'spending')->exists() ||
        auth()->user()->finances()->where('type', 'income')->exists())
        <div class="card bg-dark text-light">
          <div class="card-body">
            <form id="selectForm" action="{{ route('handleForm') }}" method="POST">
              @csrf
              <label for="options">Filter by category: </label>
              <select class="form-control" name="options" id="options" onchange="submitForm1()">
                <option value="" selected disabled>Choose one</option>
                <optgroup label="Spending categories">
                  @foreach ($spendingCategories as $category)
                    <option value="{{ $category->id }}">{{ $category->name }}</option>
                  @endforeach
                </optgroup>
                <optgroup label="Income categories">
                  @foreach ($incomeCategories as $category)
                    <option value="{{ $category->id }}">{{ $category->name }}</option>
                  @endforeach
                </optgroup>
              </select>
            </form>
          </div>
        </div>
      @endif
    </div>
    <div class="col-md-3"></div>
  </div>
</div>

```

Ez a funkció kategóriánként szűri az adatokat.

```

161 > @if (isset($selected_category)) ...
210 @endif

```

Ez a funkció jeleníti meg a kategóriaként szűrt adatokat

```

{{-- SELECT FAMILY MEMBER IF USER HAS FAMILY MEMBERS --}}
@if ($familyMembers->count() > 1) ...
@endif

```

Ez a funkció felel azért, hogy a családtagok adatait le tudjuk kérni, és meg tudjuk nézni.

Documentation.blade.php

```

<script>
    var activeButtonId = null;

    function handleButtonClick(buttonType) {
        var contentDiv = document.getElementById(buttonType);
        var button = document.getElementById(buttonType + 'Btn');
        if (buttonType === activeButtonId) {
            contentDiv.style.display = contentDiv.style.display === 'none' ? 'block' : 'none';
        } else {
            if (activeButtonId) {
                document.getElementById(activeButtonId).style.display = 'none';
            }
            contentDiv.style.display = 'block';
            if (activeButtonId) {
                document.getElementById(activeButtonId + 'Btn').classList.remove('active');
            }
            button.classList.add('active');
            activeButtonId = buttonType;
        }
    }
}
</script>

```

A gombokra kattintva hívódik meg ez a függvény. Az elrejtett és megjelenített tartalmakért felel.

Income.blade.php és spending.blade.php

```

186      /* Graphs: */
187      /* Pie graph */
188 >      var options = { ...
243      };
244      /* render apexcharts */
245      var chart = new ApexCharts(document.querySelector('#chart'), options);
246      chart.render();
247      /* Line graph */
248 >      var options = { ...
326      };
327      /* Create the chart */
328      var chart = new ApexCharts(document.querySelector('#monthlyByCategories'), options);
329      chart.render();

```

Ezek a funkciók a grafikonok kinézetéért és létrehozásáért felelősek.

```

330      /* JQUERY Table */
331 >      $(document).ready(function() { ...
455      });

```

Ez a függvény a “JQuery datatable” (táblázat) kinézetéért és létrehozásáért felelős.

```

114      /* Functions: */
115      //user associated finances php array to json string
116      var financesData = @json($finances);
117      //user associated categories php array to json string
118      var categoryPrices = {};
119 >   for (let i = 0; i < financesData.length; i++) {...
120
121   };
122
123   /* passing the data to the apexcharts */
124   /* The date inside the graphs */
125   var seriesData = [];
126 >   for (var categoryName in categoryPrices) {...
127
128   }
129   var prices = financesData.map(function(item) {
130       return item.price;
131   });
132   var sum = 0;
133   for (let i = 0; i < prices.length; i++) {
134       sum += prices[i];
135   }
136
137   /* Avg income */
138   let avarage = 'Avarage income: ' + Math.round(sum / prices.length) + ' ' + '{{ $currencySymbol }}';
139   /* Sum income */
140   sum = 'income in total: ' + sum + ' ' + '{{ $currencySymbol }}';
141   document.getElementById('sum').innerHTML = sum;
142   document.getElementById('avarage').innerHTML = avarage;
143   var monthlyCategoryPrices = {};
144 >   financesData.forEach(function(finance) {...
145
146   });
147   var categories = {};
148   var seriesData = [];
149 >   for (var key in monthlyCategoryPrices) {...
150
151   }
152 >   for (var categoryName in categories) {...
153
154   }
155

```

Ezek a funkciók az adatok lekéréséért, átlag és összesen adatok számításáért, illetve a grafikonok adatokkal való feltöltéséért felelnek.

incomeCreate.blade.php és spendingCreate.blade.php

Ez a funkció a kategóriák kilistázásáért felelős.

settings.blade.php

```
//notification handler
document.addEventListener('DOMContentLoaded', function() {
    var checkbox = document.getElementById('notification');
    document.getElementById('enableNotificationBtn').addEventListener('click', function(event) {
        var userTimezone = Intl.DateTimeFormat().resolvedOptions().timeZone;
        document.getElementById('timezone').value = userTimezone;
    });
});
```

Az értesítések beállításáért felelős ez a funkció.

```
function checkIfUserExists(input) {
    $.ajax({
        type: 'POST',
        url: '/checkIfUserExists',
        data: {
            '_token': '{{ csrf_token() }}',
            'username': input
        },
        success: function(data) {
            if (data.status === "failed") {
                $('#responseText').removeClass('text-success').addClass('text-danger').html(data.message);
                $('#addFamilyMemberBTN').prop('disabled', true)
            } else if (data.status === "failed2") {
                $('#responseText').removeClass('text-success').addClass('text-danger').html(data.message);
                $('#addFamilyMemberBTN').prop('disabled', true)
            } else {
                // If the user exists, you may choose not to display any message
                $('#responseText').removeClass('text-danger').removeClass('text-success').html('');
                $('#addFamilyMemberBTN').prop('disabled', false);
            }
        },
        error: function(xhr, status, error) {
            console.error("Error: " + status + " " + error);
            $('#responseText').html('An error occurred. Please try again later.').addClass('text-danger');
        }
    });
}
```

Ez a funkció megnézi, hogy foglalt-e már a felhasználónév, amelyre a felhasználó módosítani szeretne.

Tesztelési dokumentáció

Külső testtelési szoftvert nem használtunk, viszont a laravel beépített “dd(); (Dump and Die) funkciójával teszteltük az általunk megírt funkciókat.

```
//avarage income, spending with the same currency
$result = $this->AvarageWithTheSameCurrency();
$spendingsAverage = $result['spendingsWithTheSameCurrency'];
$incomesAverage = $result['incomesWithTheSameCurrency'];
dd($result);
```

Eredmény:

```
array:2 [▼ // app\Http\Controllers\AdvancedStatisticsController.php:135
  "spendingsWithTheSameCurrency" => 1.0
  "incomesWithTheSameCurrency" => 0.0
]
```


Összefoglalás

Továbbfejlesztési lehetőségek

- Több statisztika megjelenítése az “Advanced statistics” aloldalon a nagyobb variáció és sokoldalúság érdekében.
- Mélyebb, nagyobb szintű keresési rendszer az “Advanced statistics” aloldalon a szabadabb keresés érdekében.
- A főoldalon (Home) lévő naptár szebbé tétele.
- Két faktoros bejelentkezés a magasabb felhasználói védelem érdekében.
- Az alkalmazás több platformra (Telefonos és Asztali alkalmazás) való kiadása, hogy bármikor és bárhol használható legyen.
- Bizonyos funkciók elérése csak is prémium verzió vásárlása esetén.
- A “Documentation” oldal részletesebbé tétele, hogy a felhasználók minden kérdésükre megtalálhassák a választ.
- Amennyiben kiadásra kerülne reklámok hirdetés az oldalon.
- A “Login” és a “Register” aloldalon lévő “card” (kártya) egy úgy nevezett “Glass effect” (Üveg hatás) kinézetet kaphat a szebb megjelenés érdekében.
- Az oldal fordítási lehetősége más nyelvekre, hogy mindenkinek beszélt nyelvtől függetlenül elérhető legyen.
- Az oldal témájának válthatósága a jobb felhasználói élmény érdekében.
- Csv import, hogyha eddig valaki máshol vezette a pénzügyeit az be tudja importálni, az eddigi adatait.
- Minden felhasználónak (usernek) egy úgynevezett financial score (pénzügyi pontszám), amit egy még nem meghatározott képlet alapján számítanánk ki egy skálán.
- A regisztrációs oldalon megadható email címet leellenőrizni regisztrációkor, hogy helyes formátumban van-e.

Felhasznált irodalom (források)

Bootstrap Team (2024) Bootstrap <https://getbootstrap.com> Letöltve: 2024.01.07.

Font Awesome Team (2024) Font Awesome <https://fontawesome.com> Letöltve: 2024.01.07.

SpryMedia Ltd. (2024) JQuery table <https://datatables.net> Letölte: 2024.02.13.

Juned Chhipa (2024) Apexcharts <https://apexcharts.com> Letöltve: 2024.02.19.

Intl-tel-input Team (2024) International phone input <https://intl-tel-input.com> Letölte: 2024.04.15.

FullCalendar LLC (2024) Fullcalndar <https://fullcalendar.io> Letöltve: 2024.04.28.

Career Development Lab (2024) Fullcalndar <https://www.youtube.com/@LaravelLover> Letöltve: 2024.04.28.

Career Development Lab (2024) Fullcalndar Git hub <https://github.com/HadiNiazi/fullCalendar-js-in-Laravel> Letöltve: 2024.04.28.

CodingNepal (2024) About us page használó mozgó kártyák <https://www.youtube.com/watch?v=6QE8dXq9SOE> Felhasználva: 2024.04.11.

Currency Beacon Team (2024) Valutaváltó <https://currencybeacon.com/> Letöltve: 2024.04.26.

Currency Beacon Team (2024) Valutaváltó <https://github.com/ash-jc-allen/laravel-exchange-rates> Letöltve: 2024.04.26.

Laravel Team (2024) Laravel dokumentáció <https://laravel.com>

Freepic Team (2024) Médiafájlok <https://www.freepik.com> Letöltve: 2024.01.14

Vecteesy Team (2024) Médiafájlok <https://www.vecteezy.com> Letölte: 2024.01.14

Konzulens tanár:

Dr. Péter Szabó Richárd

Készítette:

Molnár Csaba

Láng Rihárd

Ambrus-Dobai Kristóf

Keletkezett 2024.05.07. Budapest.