

# Háztartási leltárprogram

Írta: Lakatos Gergő

# Tartalomjegyzék

---

- [Előkészület](#)
  - o [A kitűzött cél](#)
  - o [Tervezett funkciók](#)
  - o [Egyéb funkció ötletek](#)
  - o [Használni kívánt technológiák](#)
  - o [Tervezett adatbázis](#)
- [Megvalósítás](#)
  - o [Adatbázis](#)
  - o [A "token" táblák](#)
  - o [Használt technológiák](#)
  - o [Funkciók](#)
  - o [Technikai megoldások](#)
- [Szükséges változtatások](#)

# Előkészület

---

## A kitűzött cél:

A leltárprogram célja a bevásárlás megkönnyítése azáltal, hogy a felhasználó által szükségesnek jelölt termékeket a program egy külön listában tünteti fel, ha azok mennyisége nullára csökkent. Így a felhasználó megspórolja az időt, amit a bevásárló lista írására szánna és elkerülheti az esetleges véletlen kifelejtett elemeket a listáról. A program célja még, hogy a felhasználó egyszerűen tudomást szerezhessen az otthon található élelmiszerekről, és azok mennyiségéről. Így a felhasználó egyértelműen eltudja dönteni, hogy az általa elkészíteni kívánt étel elkészíthető-e a jelenlegi készlet alapján.

## Tervezett funkciók:

- Termékek felvitele és kategorizálása
- Lehetőség azon termékek megjelölésére, amelyeket a felhasználó rendszeresen vásárol
- Bevásárlólista készítése
- Regisztráció és bejelentkezés
- Egy háztartásban élő felhasználóknak képesnek kell lennie, hogy hozzáférjenek ugyanazon 'raktárkészlethez'

## Egyéb funkció ötletek:

- Lehetőség a termékek árának megadására minden vásárlás alkalmával
- Diagrammos adatkimutatás a különböző termékek árának változásáról (esetleg üzletszinten)
- Receptek mutatása a rendelkezésre álló készlet alapján

## Használni kívánt technológiák:

- html
- css, bootstrap
- php
- js
- MariaDB
- Apache
- Docker

## Tervezett adatbázis

### **user**

- id (int)
- name (varchar)
- password (varchar)
- household\_id (int)
- email (varchar)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- created\_by (int)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

### **item**

- id (int)
- name (varchar)
- category\_id (int)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- created\_by (int)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

### **item\_category**

- id (int)
- name (varchar)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- created\_by (int)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

### **inventory**

- id (int)
- household\_id (int)
- item\_id (int)
- quantity (int)
- important (int)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- created\_by (int)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

# Megvalósítás

---

## Adatbázis

### item

- id (int)
- name (varchar)
- category\_id (int)
- measurement\_id (int)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- created\_by (int)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

### item\_category

- id (int)
- name (varchar)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- created\_by (int)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

### measurement

- id (int)
- name (varchar)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- created\_by (int)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

### inventory

- id (int)
- household\_id (int)
- item\_id (int)
- quantity (int)
- important (tinyint)
- min\_quantity (int)
- in\_shopping\_list (tinyint)
- req\_quantity (int)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- created\_by (int)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

### household

- id (int)
- name (varchar)
- admin\_user\_id (int)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- created\_by (int)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

**users\_in\_household**

- id (int)
- user\_id (int)
- household\_id (int)
- created\_at (timestamp)
- updated\_at (timestamp)
- deleted\_at (timestamp)
- updated\_by (int)
- deleted\_by (int)
- deleted (tinyint)

**user**

- id (int)
- username (varchar)
- email (varchar)
- password (varchar)
- created\_at (timestamp)
- updated\_at (timestamp)

**forgotten\_password\_token**

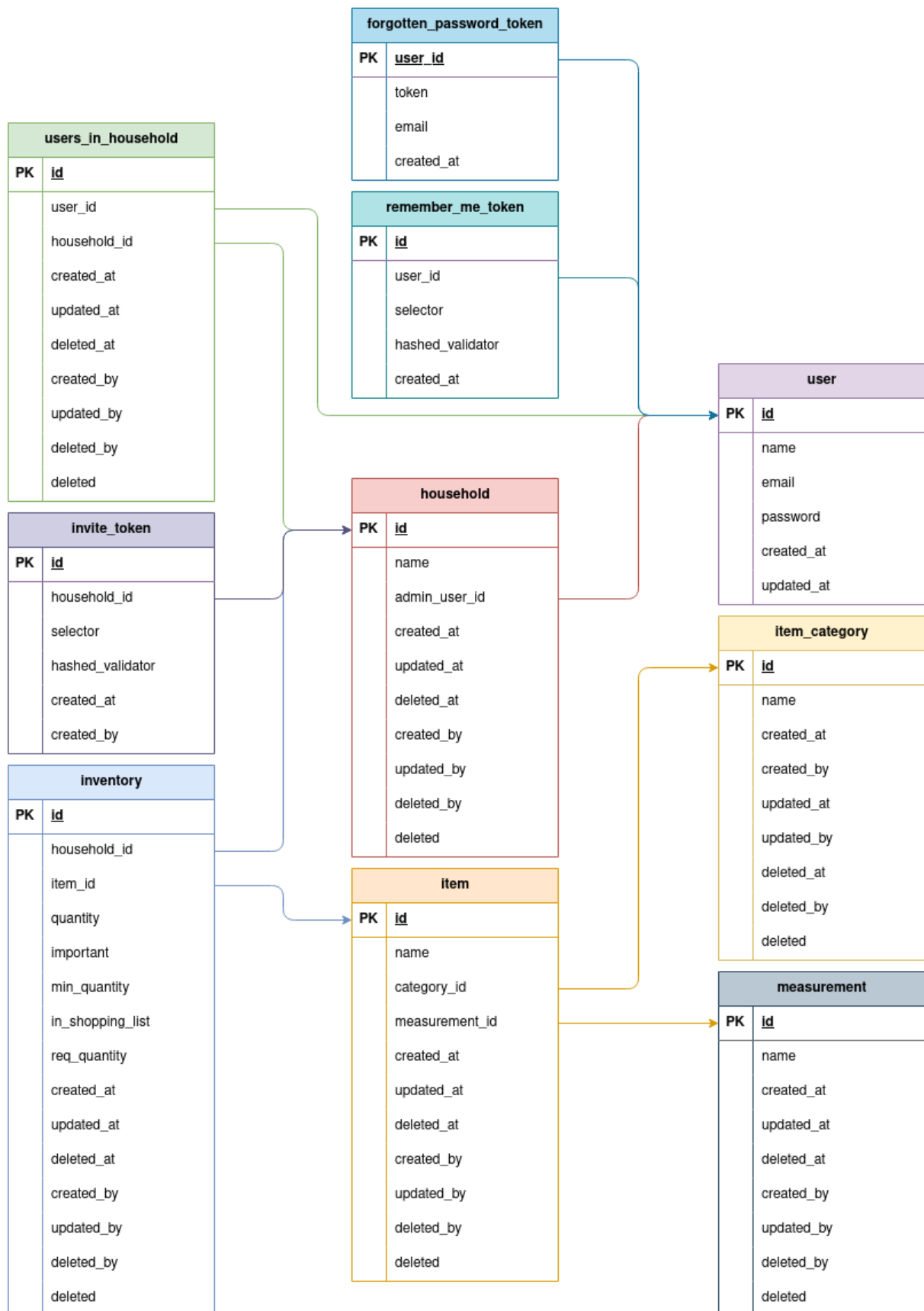
- user\_id (int)
- token (varchar)
- created\_at (timestamp)
- remember\_me\_token
- id (int)
- user\_id (int)
- selector (varchar)
- hashed\_validator (varchar)
- created\_at (timestamp)

**invite\_token**

- id (int)
- household\_id (int)
- selector (varchar)
- hashed\_validator (varchar)
- created\_at (timestamp)
- created\_by (int)

**remember\_me\_token**

- id (int)
- user\_id (int)
- selector (varchar)
- hashed\_validator (varchar)
- created\_at (timestamp)



## A “token” táblák

Bizonyos műveletek, mint az “emlékezz rám” funkció egy weboldalon megkívánnak valamilyen ellenőrző funkció meglétét, amit sok esetben egy úgynevezett token segítségével oldanak meg ami egy megadott hosszúságú karaktersor, amely sokszor egyszer használható és/vagy lejáratí dátummal rendelkezik.

Én a “selector:validator” nevű módszer mellett döntöttem mert ezt találtam a legbiztonságosabb megoldásnak.

### selector:validator

Ennek a módszernek az előnye az egyszerű token-nel szemben, hogy megoldást nyújt az időalapú támadásokkal szemben melyek az adatbázis lekérdezések válaszüldéből következtetik ki a token-t. Ez ellen egy selector és egy validátor nevű tokent használunk.

A selector egy “hagyományos” token, míg a validator valamilyen hash titkosítással van kezelve és az adatbázisban ebben a formájában van tárolva ebben az esetben hashed\_validator néven a titkosítás mentes selector-ral.

A jogosult felhasználó rendelkezik mind a selector mind pedig validator eredeti titkosítás mentes változatával. Az azonosítás úgy történik, hogy a szerver indít egy lekérdezést melyhez a selector-t használja fel. Válaszban megkapja a hashed\_validator-t és a hozzátartozó adatokat, azonban még szükséges egy további ellenőrzést is elvégezni. A validator “megfelelő” módon való összehasonlítása a hashed\_validator-ral kiküszöböli az időalapú támadások lehetőségét. Ebben az esetben a “megfelelő” mód a php két beépített függvénye a password\_hash() és password\_verify() melyek tovább növelik a biztonságot.



## Használt technológiák

- html
- css, bootstrap
- php 8.2.9
- javascript
- MariaDB
- Apache
- Docker
  - Docker Compose
- Adminer

Az oldal működéséhez a Docker-t használtam azon belül is a Docker Compose-t amely lehetővé teszi több Docker image használatát egyszerre. Azért választottam ezt a megoldást mert így nem szükséges a szerveren is külön bekonfigurálni a különböző komponenseket és az esetleges verzió béli eltérésekkel sem kell foglalkozni.

## Docker

A Docker egy korszerű konténerizációs technológia, ami lehetővé teszi, hogy egy alkalmazás olyan izolált környezetben fusson, ahol minden szükséges függősége jelen van, így biztosítva, hogy minden környezetben ugyan úgy működjön a program.

## Docker image

A Docker, fontos alapelemét image-nek hívjuk. Egy image olyan, mint egy tervrajz, amelyben közvetlenül nem végzünk változtatásokat, hanem konténereket hozunk létre belőle és ezekkel dolgozunk. Egy image-ből akár mennyi konténer létrehozható és mindegyik megegyezik az eredeti image-el.

## Docker Compose

A Docker Compose lehetővé teszi több image egyidejű használatát egy projektben melyhez egy docker-compose.yml fájl szükséges. Ez a fájl tartalmazza az összes szükséges konfigurációt, ami különböző lehet minden image esetében.

```
1 version: '3.1'
2 services:
3   php:
4     build:
5       context: .
6       dockerfile: PHP.Dockerfile
7     volumes:
8       - ./src:/var/www/html
9     ports:
10      - 80:80
11     depends_on:
12      - mysql
13   adminer:
14     image: adminer:latest
15     restart: always
16     ports:
17      - 8080:8080
18     volumes:
19      - ./adminer.css:/var/www/html/adminer.css
20   mysql:
21     image: mariadb:latest
22     environment:
23       MYSQL_ROOT_PASSWORD: 'secret'
24     volumes:
25       - mysqldata:/var/lib/mysql
26     ports:
27       - 3306:3306
28 volumes:
29   mysqldata: {}
```

Megadható a szükséges image neve a verziószámával együtt vagy a "latest" tag-et használva, amely mindig a legújabb verziót jelenti. Megadható az image helyett egy Dockerfile is mely tartalmazhat különböző az adott image-ből létrejött konténernek szánt utasításokat.

Többek között megadható még az image által használt port száma vagy számai. Megadható, hogy milyen adatokhoz férjen hozzá a konténeren kívül, de ezen kívül még sok más konfigurálási lehetőség áll rendelkezésre.

Alább található a Dockerfile.

```
1 FROM php:8.2.9-apache
2
3 # Install system dependencies and PHP extensions
4 RUN apt-get update && apt-get install -y \
5     libzip-dev \
6     && docker-php-ext-install zip pdo pdo_mysql mysqli
7
8 # Install GD library and other required packages
9 RUN apt-get install -y libpng-dev libjpeg-dev && \
10     docker-php-ext-configure gd --with-jpeg && \
11     docker-php-ext-install -j$(nproc) gd
12
13 COPY custom-php.ini /usr/local/etc/php/php.ini
14
15 # Install Composer
16 COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
17
18 # Copy your application code
19 COPY src/ /var/www/html/
20
21 # Modify httpd.conf to enable AllowOverride
22 RUN sed -i 's/AllowOverride None/AllowOverride All/g' /etc/apache2/apache2.conf
23
24 # Enable mod_rewrite
25 RUN a2enmod rewrite
26
27 # Install and enable xdebug
28 RUN pecl install xdebug && docker-php-ext-enable xdebug
29
30 # Create a non-root user
31 RUN addgroup --gid 1000 mygroup && \
32     adduser --system --no-create-home --disabled-password --disabled-login --uid 1000 --ingroup mygroup myuser
33
34 # Change ownership of files
35 RUN chown -R myuser:mygroup /var/www
36 USER myuser
37
38 # Set working directory
39 WORKDIR /var/www/html
```

## Adminer

Az Adminer egy webes adatbáziskezelő felület, amit azért raktam bele a projectbe, hogy megkönnyítse a fejlesztést. A fájlok között található egy különálló css is, ami az Adminer felületét hivatott szebbé, korszerűbbé tenni. Ezt a css-t nem én készítettem, a fájlt az internetről töltöttem le és tartalmazza a megfelelő hivatkozást bár ez a projekt szempontjából irreleváns mert a végfelhasználó nem találkozik ezzel a felülettel semmilyen módon.

## MariaDB

A MariaDB és a MySQL között viszonylag kevés eltérés található, viszont a MariaDB open source licensszel rendelkezik így ezt választottam a projekt adatbázisának.

## PHP

A php szintén egy Docker image-ből származik, ami a 8.2.9-es verziója mellet tartalmazza az Apache szerveret is. A php-hoz használtam egy Dockerfile-t mert így tudtam bekapcsolni és konfigurálni a komponenseit. Emellett telepítettem is így két kiegészítőt, az Xdebug-ot és a Composer-t.

## Xdebug

Az Xdebug egy php-hoz készült kiegészítés, amely a hibajavítást teszi könnyebbé a fejlesztő számára azáltal, hogy érthetőbben jeleníti meg a hibákat.

## Composer

A Composer egy függőség kezelő rendszer a php-hoz. Ebben a projektben azért volt rá szükségem mert egy PHPMailer nevű könyvtárat ezt használva egyszerűbb volt telepítenem.

## PHPMailer

A PHPMailer az egyik legnépszerűbb email küldő könyvtár php-hoz. Erre azért volt szükségem mert ugyan a php tartalmaz egy mail() nevű függvényt, azt nehezebb lenne bekonfigurálni, hogy működjön is.

Továbbá szükségem volt egy SMTP (Simple Mail Transfer Protocol) szerverre is, hogy működjön a levélküldés. Ehhez a [Brevo](#) szolgáltatását választottam mert ingyenesen naponta 300 email-t engednek küldeni, ami jelenleg több mint elég.

Készítettem még egy külön classt, hogy az SMTP szerverhez szükséges csatlakozási adatokat eléglegyen egyszer megadni.

```
1 <?php
2
3 namespace App\Utils;
4
5 require_once "vendor/autoload.php";
6
7 use PHPMailer\PHPMailer\PHPMailer;
8 use PHPMailer\PHPMailer\SMTP;
9
10 You, last week | I author (You)
11 class Mail {
12     public function send($to, $from, $subject, $message) {
13         try {
14             $mail = new PHPMailer(true);
15             $mail->isSMTP();
16             $mail->SMTPAuth = true;
17             $mail->Host = "smtp-relay.brevo.com";
18             $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
19             $mail->Port = 587;
20             $mail->Username = [REDACTED];
21             $mail->Password = [REDACTED];
22
23             $mail->setFrom($from, '');
24             $mail->addAddress($to);
25
26             $mail->CharSet = "UTF-8";
27             $mail->Subject = "=?UTF-8?B? . base64_encode($subject) . "?=";
28
29             $mail->Body = $message;
30
31             $mail->send();
32             return true;
33         } catch (\Exception $e) {
34             return false;
35         }
36     }
37 }
```

## Funkciók

### Regisztráció

Az oldalra bárkinek lehetősége van regisztrálni amennyiben rendelkezik egy email címmel. Az oldal ellenőrzi a megadott adatokat, hogy ne lehessen ugyan azzal a felhasználó névvel vagy email címmel regisztrálni kétszer.



Háztartási Leltárprogram

Bejelentkezés → Regisztráció

### Regisztráció

Felhasználónév \*

Email \*

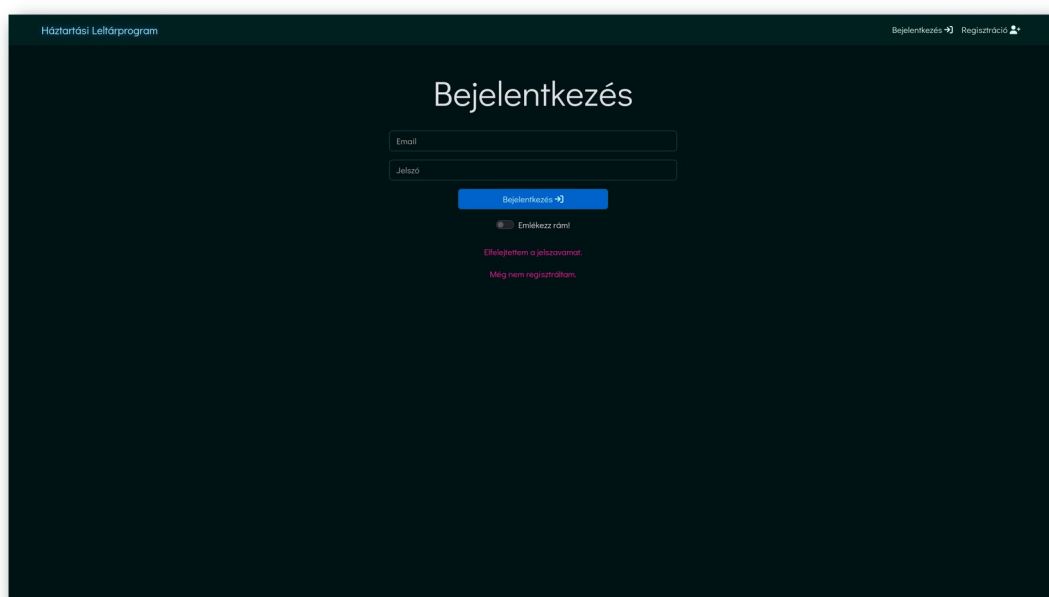
Jelszó \*

Jelszó újra \*

Regisztráció →

### Bejelentkezés

A bejelentkezési oldal tartalmaz még a bejelentkezésen kívül további funkciókat is, amik az „Emlékezz rám!”, „Elfelejtettem a jelszavamat.” és a „Még nem regisztráltam.”.



Háztartási Leltárprogram

Bejelentkezés → Regisztráció

### Bejelentkezés

Email

Jelszó

Bejelentkezés →

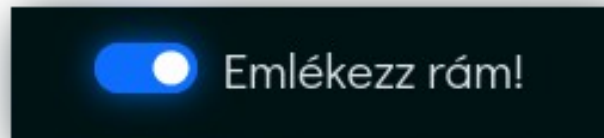
☐ Emlékezz rám!

[Elfelejtettem a jelszavamat.](#)

[Még nem regisztráltam.](#)

## Emlékezz rám

Az „Emlékezz rám!” egy checkbox mező, ami “bepipálva” eltárol egy tokent a felhasználóról a selector:validator módszert használva, természetesen csak sikeres bejelentkezés esetén. Így egy cookie-ban lesz eltárolva a generált token a felhasználó eszközén 7 napig.



## Elfelejtettem a jelszavam

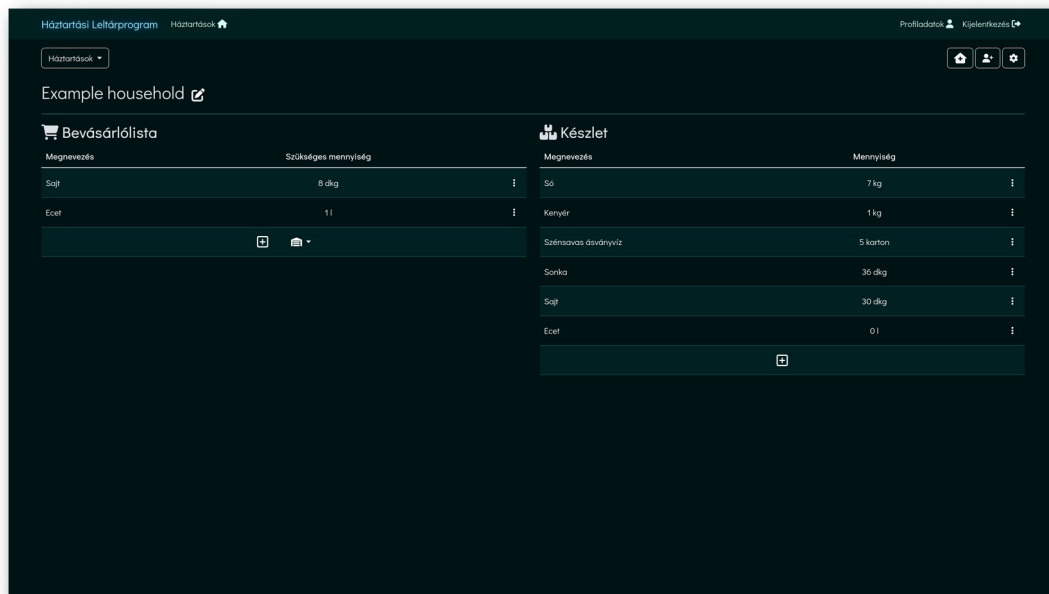
Az „*Elfelejtettem a jelszavam*.” egy hivatkozás, ami átirányít egy lapra, ahol a felhasználó megadhatja az email címét és amennyiben helyesen adta meg kap egy emailt egy hivatkozással az oldalra, amely tartalmaz egy generált token-t ami átirányítja az oldalra, ahol megadhatja új jelszavát, ezután a token törlődik.

The screenshot shows a web form titled "Jelszó visszaállítás kérése" (Reset Password Request). The form is set against a dark background with light text. At the top left, it says "Háztartási Leltárprogram" and at the top right, "Bejelentkezés" and "Regisztráció". The form contains a single text input field labeled "Email". Below the input field are two buttons: a blue button labeled "Küldés" (Send) with a right-pointing arrow, and a dark button labeled "Vissza" (Back) with a left-pointing arrow.

The screenshot shows a web form titled "Új jelszó megadás" (New Password Setting). The form is set against a dark background with light text. At the top left, it says "Háztartási Leltárprogram" and at the top right, "Bejelentkezés" and "Regisztráció". The form contains two text input fields: the first is labeled "Új jelszó" (New password) and the second is labeled "Új jelszó újra" (New password again). Below the input fields are two buttons: a dark button labeled "Küldés" (Send) with a right-pointing arrow, and a dark button labeled "Vissza a főoldalra" (Back to homepage) with a left-pointing arrow.


## Háztartások kezelése

A felhasználó bejelentkezés/regisztráció után a háztartások oldalra kerül, ahol különféle műveleteket hajthat végre.



## Háztartás

A háztartás egy csoport vagy halmaz mely magában foglalja a "készlet"-et, a "bevásárlólistát" és egy vagy több felhasználót.

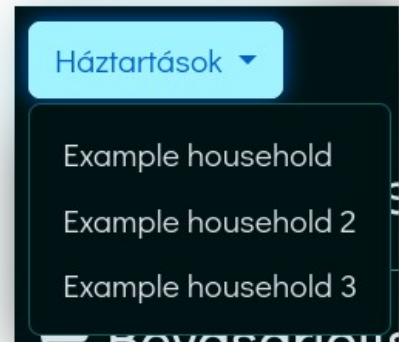
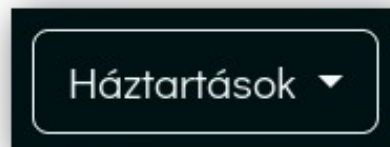
A felhasználó módosíthatja a már meglévő háztartások nevét , létrehozhat új háztartásokat, meghívhat más felhasználókat vagy eltávolíthat felhasználókat a háztartásból vagy törölheti a háztartást, ha a megfelelő joggal rendelkezik az adott művelethez.

A felsoroltakból egyedül a háztartás nevét módosíthatja ezen az oldalon a felhasználó, a többi művelet 3 különböző gomb segítségével érhető el.





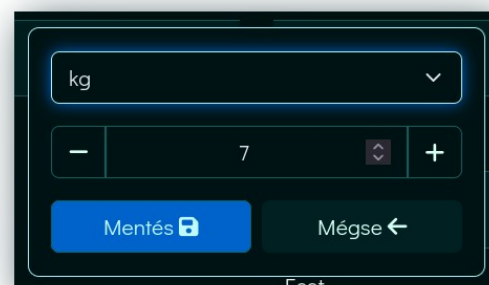
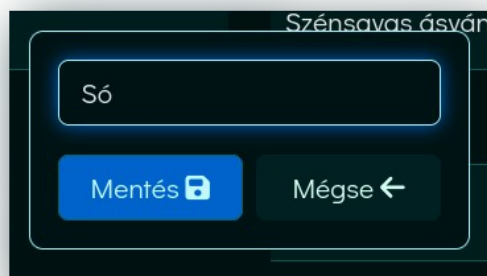
Továbbá lehetősége van kiválasztani az éppen használt háztartást, ha több háztartásnak is a tagja.



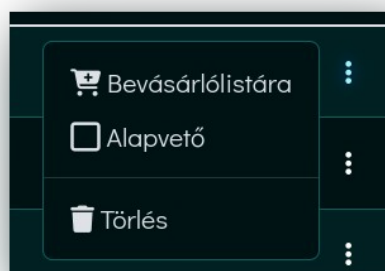
### Készlet

A készlet a háztartásban található összes termék listája.

A felhasználó módosíthatja a készleten lévő elemek nevét, mennyiségét és mértékegységét egy felugró ablakban, ha rákattint az elem nevére vagy mennyiségére.




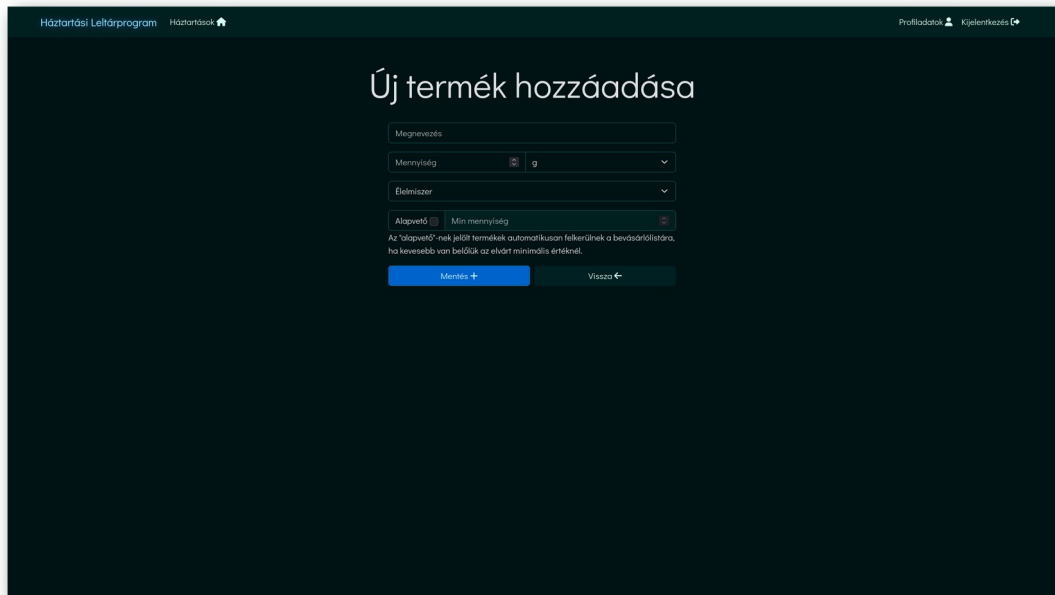
Továbbá minden elemnél lehetőség van feltenni a bevásárlólistára, "alapvetővé" tenni vagy törölni.



### Alapvető:

Az "alapvető"-nek jelölt termékek automatikusan felkerülnek a bevásárlólistára, ha kevesebb van belőlük az elvárt minimális értéknél.

Lehetőség van még a listához új elemet hozzáadni a  gombra kattintva egy másik oldalon.




Háztartási Leltárprogram    Háztartások

Profiladatok    Kijelentkezés

## Új termék hozzáadása

Megnevezés

Mennyiség  g

Értékszer

Alapvető ☐    Min mennyiség

Az "alapvető"-nek jelölt termékek automatikusan felkerülnek a bevásárlólistára, ha kevesebb van belőlük az elvárt minimális értéknél.

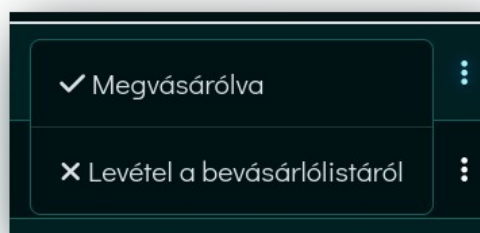
Mentés +    Vissza <



Itt a felhasználónak lehetősége van megadni a termék nevét, mennyiségét, mértékegységét, kategóriáját vagy azt, hogy alapvető-e és ha igen akkor kötelező megadnia a minimum szükséges mennyiségét.

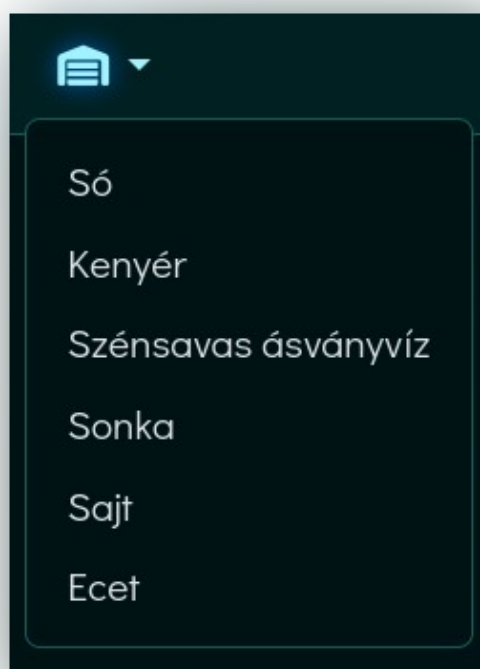
## Bevásárlólista

A bevásárlólista tartalmaz minden olyan terméket, amit egy felhasználó feltett a listára vagy automatikusan felkerült.

A felhasználónak lehetősége van módosítani a listán lévő elem szükséges mennyiségét. Továbbá lehetőség van levenni az elemet a listáról, ha mégsem szeretné, hogy a listán legyen vagy megadhatja, hogy megvásárolta a terméket, ekkor lekerül az elem a listáról és a megadott szükséges mennyisége hozzáadódik a készleten lévő mennyiségéhez.

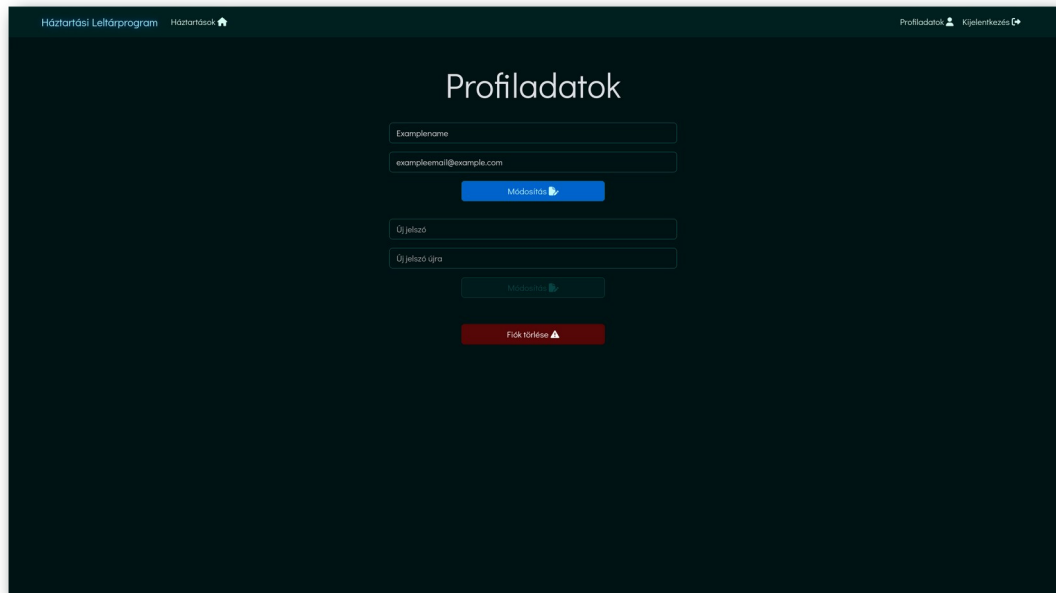


Lehetőség van még közvetlenül a listára tenni elemeket a listára, ami lehet olyan elem, ami nincs hozzáadva a készlethez a  gombra kattintva, ami készlethez adáshoz hasonló oldalt hoz be, vagy a  gombra kattintva a készleten lévő elemek listájáról is adhat hozzá terméket a felhasználó.



## Profiladatok szerkesztése

Minden felhasználónak lehetősége van utólag módosítani a regisztrációkor megadott adatait, továbbá törölheti is a profilját, ha úgy kívánja.



Háztartási Leltárprogram    Háztartások

Profiladatok    Kijelentkezés

### Profiladatok

Example name

exampleemail@example.com

Módosítás

Új jelszó

Új jelszó újra

Módosítás

Fiók törlése

## Technikai megoldások

A fejlesztésnél az MVC (Model-View-Controller) programfejlesztési mintát követtem. A fájlokat a következő mappastruktúrába rendeztem:

- src
  - o App
    - \_utils
    - Controller
    - Model
    - View
  - o public
    - css
    - js
    - pictures
  - o vendor
    - composer
    - phpmailer

Az App mappában van minden szerveroldali php kód.

Az \_utils mappában kiegészítő segítő class-ok vannak.

A Controller mappa azokat a class-okat tartalmazza melyek kezelik a view-ból kapott inputokat és kommunikálnak a model-lel.

A Model mappa tartalmazza a model class-okat amik kommunikálnak az adatbázissal.

A View mappa azokat a fájlokat tartalmazza melyek tartalmazznak html-t de nem tartalmazznak business logic-ot.

A public mappa tartalmaz minden olyan fájlt, amihez bárki hozzáférhet, mint képek, javascript fájlok és css-ek.

A vendor mappa külső könyvtárak fájljait tartalmazza.

## RESTful API

A REST (Representational State Transfer) API egy olyan programozási interfész, amely lehetővé teszi az alkalmazások közötti kommunikációt.

Ezt az URL-ben *controller* és *action* segítségével értem el. Ez alapján az oldal URL címe a következőképpen alakul:

*oldal\_domain\_neve?controller=kontroller\_neve&action=művelet\_neve*

Ezt egy index.php nevű fájl kezeli, úgy, hogy lekéri az URL-ből a controller és az action értékét majd switch-ek és if-ek segítségével meghívja a megfelelő függvényt a megfelelő controller class-ból.

```
17 $currentController = isset($_GET['controller']) ? $_GET['controller'] : "index";  
18 $currentAction = isset($_GET['action']) ? $_GET['action'] : "login";  
19
```

Továbbá minden esetben megvizsgálja egy ellenőrző class, hogy jogosult-e a felhasználó az adott oldal elérésére.

```
1 <?php  
2  
3 namespace App\utils;  
4  
5 ...  
6 class PermissionManager {  
7     public static function IsPermitted($requiredLevel) {  
8         @session_start();  
9  
10        if (!isset($_SESSION['usr_level'])) {  
11            return false;  
12        }  
13  
14        if ($requiredLevel === "guest") {  
15            if ($_SESSION['usr_level'] !== "guest") {  
16                header('Location: index.php?controller=household&action=list');  
17                return false;  
18            }  
19            return true;  
20        } else if ($requiredLevel === "user") {  
21            if ($_SESSION['usr_level'] !== "user") {  
22                header('Location: index.php?controller=index&action=login');  
23                return false;  
24            }  
25            return true;  
26        } else if ($requiredLevel === "admin") {  
27            if ($_SESSION['usr_level'] !== "admin") {  
28                return false;  
29            }  
30            return true;  
31        }  
32        return false;  
33    }  
34 }
```

Ugyan tartalmaz egy admin lehetőséget is azonban nem tudtam/tudom, hogy szükségem lesz-e rá a jövőben. Ezért meghagytam mert gondot nem okoz.

# Szükséges változtatások

---

A következő változtatásokat szükséges lenne beiktatni ám még nem jutott rájuk idő vagy nem gondoltam előre, hogy szükségesek lennének. Viszont biztonsági kockázatot jelenthetnek és egy részük jogi kötelezettségek miatt szükséges így a jövőben mindenképpen beiktatom ezeket a változtatásokat.

- Ellenőrző email küldése regisztráció után
- Captcha beiktatása regisztrációhoz
- Felhasználási feltételek és adatkezelési nyilatkozat megírása
- A háztartások azonosító száma helyett token használata
- Automatikus record törlés a token táblákból, ha azoknak lejár az ideje