

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

SZAKDOLGOZAT

**Fenyvesi Ferenc
Regős Gábor Bence
Szaszovszki Kevin János**

2023.

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



SZAKDOLGOZAT

Mester Hármas Webshop

Konzulens: Wiezl Csaba

Készítette: Fenyvesi Ferenc

Regős Gábor Bence

Szászovszki Kevin János

Hallgatói nyilatkozat

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

Fenyvesi Ferenc

Regős Gábor Bence

Szaszovszki Kevin János

Konzultációs lap

Vizsgázók neve: Fenyvesi Ferenc, Regős Gábor Bence, Szaszovszki Kevin János

Szakdolgozat címe: Mester Hármas Webshop

Program nyújtotta szolgáltatások:

- Bejelentkezés/Regisztráció
- Termékek megjelenítése
- Termékek vásárlása
- Kategória szerinti szűrés
- Admin felület biztosítása
- Termékek hozzáadása, eltávolítása, módosítása

Sorszám	A konzultáció időpontja	A konzulens aláírása
1.	2022.10.10.	
2.	2022.11.14.	
3.	2022.12.12.	
4.	2023.01.16	
5.	2023.02.13.	
6.	2023.03.13	

A szakdolgozat beadható:

Vác, 2023.....

A szakdolgozatot átvettetem:

Vác, 2023.....

.....
Konzulens

.....
A szakképzést folytató
intézmény felelőse

Tartalomjegyzék

Hallgatói nyilatkozat	3
Konzultációs lap	4
Témaválasztás	7
1 Fejlesztői dokumentáció	8
1.1 Fejlesztői környezet.....	8
Visual Studio Code	8
1.2 Használt nyelvek	8
1.2.1 PHP	8
1.2.2 JavaScript	8
1.3 Technológiák.....	9
1.3.1 REST API	9
1.3.2 Bootstrap	9
1.3.3 JSON	9
1.3.4 AJAX	9
1.3.5 MVC.....	9
1.3.6 HTML.....	10
1.3.7 CSS.....	10
1.3.8 Twig	10
1.3.9 Xampp	10
1.3.10 Git.....	10
1.4 Adatbázis	11
1.4.1 Adatbázis alapadatai	11
1.4.2 Kapcsolatok.....	12
1.4.3 Táblák.....	13
1.5 Osztályok, modulok ismertetése.....	31
1.5.1 Adatbázis.....	31
1.5.2 A „vendor” mappa	31
1.5.3 A „composer.lock” fájl	32
1.5.4 A „composer.json” fájl	33
1.5.5 A „index.php” fájl.....	34
1.5.6 A „Controller” mappa	36
1.5.7 A „Model” mappa	43
1.5.8 A „View” mappa.....	54
1.6 Tesztelés	58

1.6.1	Tesztesetek	59
1.7	Továbbfejlesztési lehetőségek	60
2	Felhasználói dokumentáció	61
2.1	Alkalmazás rövid ismertetése	61
2.2	Rendszerkövetelmények	61
2.3	Alkalmazás telepítése, szükséges beállítások	61
2.4	Alkalmazás használata.....	62
2.4.1	Bejelentkezés/Regisztráció	62
2.4.2	Elfelejtett jelszó	63
2.4.3	Főoldal.....	63
2.4.4	Férfi/Női/Gyermekek oldalak	64
2.4.5	Termék keresés	67
2.4.6	Sötét mód.....	67
2.4.7	Profil adatok.....	68
2.4.8	Szállítási adatok.....	68
2.4.9	Jelszó megváltoztatása	69
2.4.10	Kosár	69
2.4.11	Fizetés véglegesítés	70
2.4.12	Admin felület	71
3	Irodalomjegyzék	76
3.1	Online tartalmak.....	76
3.2	Könyvek.....	76
4	Mellékletek	77
4.1	Webes alkalmazás	77

Témaválasztás

Szakdolgozatunk témájának egy webshopot választottunk, amely online webes felületen érhető el. A használt technológiák terén igyekeztünk minél újabbakat használni.

A képek, termékek keresése és lementése internetről történt.

Úgy gondoljuk, hogy ez a webshop hasznos lehet, mivel lerövidíti a vásárlás folyamatát, így utazás nélkül egyszerűen otthonról tudunk ruhákat vásárolni. A vásárló kedvére válogathat különböző világmárkák ruha fajtái közül.

A szoftvert úgy készítettük, hogy minél felhasználóbb barát, egyszerűen érhető, könnyen kezelhető legyen. A weboldal alkalmas lehet egy cég termékeinek megjelenítésére, forgalmazására, eladására. Az alkalmazást úgy készítettük el, hogy reszponzív jelenjen meg minden eszközön, legyen az egy tablet, telefon vagy egy számítógép.

A projekt során a feladatokat az alábbiak szerint osztottuk el: admin felület, backend, frontend, adatbázis.

A fejlesztői felületet Szaszovszki Kevin János hozta létre, amely rest api-val áll össze. Továbbá létrehozta a webshop kinézetét és adatbázisát Fenyvesi Ferenc segítségével.

A földali ajánlott termékek listázását, keresést, illetve a nemi oldalakon való szűrést valósította meg.

Az admin felületet Regős Gábor Bence készítette el. A regisztráció/bejelentkezés backend részét is ő készítette el. Általa lett megvalósítva a vásárlás véglegesítése a backend részről.

A termékek nem szerinti listázását, egyes termék megtekintési oldalát, illetve a kosarat Fenyvesi Ferenc készítette el.

A dokumentáció elkészítése közös folyamat volt.

1 Fejlesztői dokumentáció

1.1 Fejlesztői környezet

Visual Studio Code¹

Egy hatékony kódszerkesztő, amely csaknem minden nyelvhez használható, bármely operációs rendszeren fut, és használatával szerkesztést, hibakeresést és Azure-beli üzembe helyezést is végezhet.

1.2 Használt nyelvek

1.2.1 PHP²

Egy nyílt forráskódú szerveroldali szkriptnyelv dinamikus weboldalak készítéséhez. Ez egy olyan általános célú nyelv is, amellyel számos projektet készíthet, beleértve a grafikus felhasználói felületeket is.

1.2.2 JavaScript³

A JavaScript egy programozási nyelv, amelyet kifejezetten az internetre fejlesztettek ki. A legtöbb webböngésző szoftver, és modern okostelefonok is mind támogatják a JavaScriptet. A JavaScriptet első sorban arra használják, hogy gazdagabb, felhasználóbarát élményeket teremtsenek vele az interneten böngészők számára, például dinamikusan frissülő weboldalakat, intuitív felhasználói felületeket, menüket, párbeszédpaneleket, 2D-s és 3D-s grafikákat, interaktív térképeket, videólejátszókat, és számos egyéb elemet, illetve funkciót. A JavaScript egy komplett programnyelv-fordító, amely közvetlenül a webböngésző szoftverekben működik.

¹ <https://azure.microsoft.com/hu-hu/products/visual-studio-code/>

² <https://hu.wikipedia.org/wiki/PHP>

³ <https://matebalazs.hu/javascript.html>

1.3 Technológiák

1.3.1 REST API⁴

A REST API, vagyis a Reprzentációs Állapotátviteli interfész egy olyan webes szolgáltatás típus, amely az HTTP módszereket (például GET, POST, PUT, DELETE) használja az erőforrások (például adatok vagy funkcionálitás) elérésére és manipulálására egy szerveren. A REST API-k gyakran használatosak webes és mobil alkalmazásokban, hogy a felhasználók az adatokkal vagy funkciókkal kommunikálhassanak a szerveren. Népszerű választás a skálázható és karbantartható webes szolgáltatások építésére, mivel könnyen érhetőek, rugalmasak, és különböző programozási nyelvekkel és platformokkal működnek.

1.3.2 Bootstrap⁵

A Bootstrap egy ingyenes front-end keretrendszer a gyorsabb és egyszerűbb webfejlesztéshez. A Bootstrap HTML és CSS alapú tervezési sablonokat tartalmaz tipográfiához, űrlapokhoz, gombokhoz, táblázatokhoz, navigációhoz, modálokhoz, képkarusszelekhez és sok másra, valamint opcionális JavaScript pluginekhez. A Bootstrap lehetővé teszi a responzív dizájn egyszerű létrehozását is.

1.3.3 JSON⁶

A JSON (JavaScript Object Notation) egy kis méretű, szöveg alapú szabvány, ember által olvasható adatcserére.

1.3.4 AJAX⁷

Az Ajax (Asynchronous JavaScript and XML) interaktív webalkalmazások létrehozására szolgáló webfejlesztési technika. A weblap kis mennyiségű adatot cserél a szerverrel a háttérben, így a lapot nem kell újrátölteni minden egyes alkalommal, amikor a felhasználó módosít valamit. Ez növeli a honlap interaktivitását, sebességét és használhatóságát.

1.3.5 MVC⁸

A Model-View-Controller egy olyan programtervezési minta, amely az alkalmazást három fő logikai részre bontja: a modellre, a nézetre, és a vezérlőre.

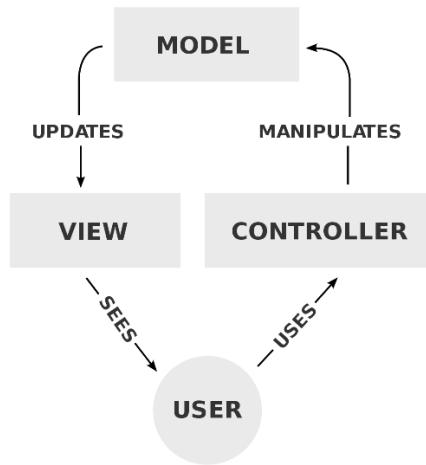
⁴ <https://www.ibm.com/topics/rest-apis>

⁵ https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

⁶ <https://hu.wikipedia.org/wiki/JSON>

⁷ [https://hu.wikipedia.org/wiki/Ajax_\(programoz%C3%A1s\)](https://hu.wikipedia.org/wiki/Ajax_(programoz%C3%A1s))

⁸ https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm



1.3.6 HTML⁹

HyperText Markup Language (HTML) szabványos jelölőnyelv a webböngészőben való megjelenítésre tervezett dokumentumokhoz.

1.3.7 CSS¹⁰

A Cascading Style Sheets (CSS) egy stíluslapnyelv, amelyet egy jelölőnyelven, például HTML-ben írt dokumentumok megjelenítésének leírására használnak.

1.3.8 Twig¹¹

A Twig egy sablonmotor a PHP programozási nyelvhez. Az eredeti verziót Armin Ronacher készítette. A Symfony PHP keretrendszer a 2. verzió óta a Twig alapértelmezett sablonmotorjának csomagolt támogatásával érkezik.

1.3.9 Xampp¹²

A XAMPP egy szabad és nyílt forássú platformfüggetlen webszerver-szoftvercsomag, amelynek alkotóelemei az Apache webszerver, a MariaDB adatbázis-kezelő, valamint a PHP és a Perl programozási nyelvek értelmezői.

1.3.10 Git¹³

A git egy nyílt forráskódú verziókövető rendszer.

⁹ <https://en.wikipedia.org/wiki/HTML>

¹⁰ <https://en.wikipedia.org/wiki/CSS>

¹¹ [https://en.wikipedia.org/wiki/Twig_\(template_engine\)](https://en.wikipedia.org/wiki/Twig_(template_engine))

¹² <https://hu.wikipedia.org/wiki/XAMPP>

¹³ <https://git-scm.com>

1.4 Adatbázis

1.4.1 Adatbázis alapadatai

Sql szerver: mysql

Neve: mesterh2_mesterharmas

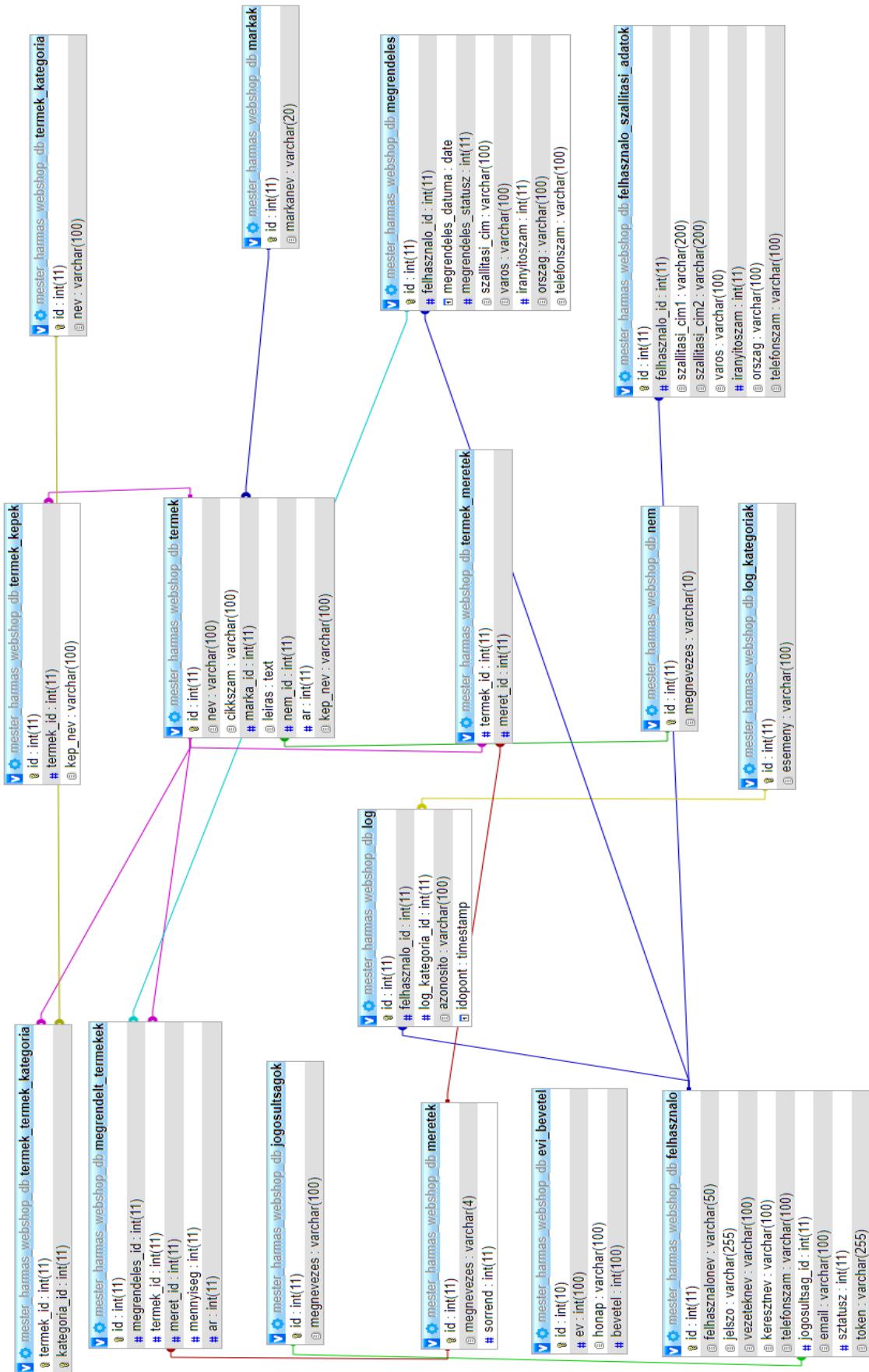
Illesztés: utf8_general_ci

Motor: innodb

SQL-parancs:

```
CREATE DATABASE IF NOT EXISTS `mesterh2_mesterharmas`  
DEFAULT CHARACTER SET utf8  
COLLATE utf8_general_ci;
```

1.4.2 Kapcsolatok



1.4.3 Táblák

1.4.3.1 Az „evi_bevetel” tábla

Az adott évi bevételeket tárolja el hónapokra osztva.

Mező neve	Típusa	Leírás	FK/PK
id	int(10)	Egyedi belső azonosító	PK
ev	int(100)	Adott év	
honap	varchar(100)	Hónap megnevezése	
bevetel	int(100)	Bevétel mennyisége	

SQL

```
CREATE TABLE `evi_bevetel` (
    `id` int(10) NOT NULL,
    `ev` int(100) NOT NULL,
    `honap` varchar(100) COLLATE utf8_hungarian_ci NOT NULL,
    `bevetel` int(100) NOT NULL
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_hungarian_ci;
ALTER TABLE `evi_bevetel`
ADD PRIMARY KEY (`id`);
ALTER TABLE `evi_bevetel`
MODIFY `id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;
```

1.4.3.2 A „felhasznalo” tábla

A felhasználók adatait tárolja el.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
felhasznalonev	varchar(50)	Felhasználónévet tárolja	
jelszo	varchar(255)	Jelszót tárolja	
vezeteknev	varchar(100)	Vezetéknévét tárolja	
keresztnév	varchar(100)	Keresztnévét tárolja	
telefonszám	varchar(100)	Telefonszámot tárolja	
jogosultsag_id	int(11)	Jogosultság azonosítója	FK
email	varchar(100)	Email címet tárolja	
sztatusz	int(11)	Felhasználó státuszát tárolja	
token	varchar(255)	Felhasználó token-jét tárolja	

SQL

```
CREATE TABLE `felhasznalo` (
    `id` int(11) NOT NULL,
    `felhasznalonev` varchar(50) CHARACTER SET utf8mb4 NOT NULL,
    `jelszo` varchar(255) CHARACTER SET utf8mb4 NOT NULL,
    `vezeteknev` varchar(100) NOT NULL,
    `keresztnev` varchar(100) NOT NULL,
    `telefonszam` varchar(100) NOT NULL,
    `jogosultsag_id` int(11) NOT NULL,
    `email` varchar(100) CHARACTER SET utf8mb4 NOT NULL,
    `szstatusz` int(11) NOT NULL,
    `token` varchar(255) CHARACTER SET utf8mb4 NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `felhasznalo`
    ADD PRIMARY KEY (`id`),
    ADD KEY `jogosultsag_id` (`jogosultsag_id`);

ALTER TABLE `felhasznalo`
    MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=39;

ALTER TABLE `felhasznalo`
    ADD CONSTRAINT `felhasznalo_ibfk_1` FOREIGN KEY (`jogosultsag_id`) REFERENCES
`jogosultsagok` (`id`);
```

1.4.3.3 A „*felhasznalo_szallitasi_adatok*” tábla

A felhasználók szállítási adatait tárolja el.

Mező neve	Típusa	Leírás	FK/ PK
<code>id</code>	<code>int(11)</code>	Egyedi belső azonosító	PK
<code>felhasznalo_id</code>	<code>int(11)</code>	Felhasználó azonosítója	FK
<code>szallitasi_cim1</code>	<code>varchar(200)</code>	Első szállítási címet tárolja	
<code>szallitasi_cim2</code>	<code>varchar(200)</code>	Másodlagos szállítási címet tárolja	
<code>varos</code>	<code>varchar(100)</code>	Város nevét tárolja	
<code>iranyitoszam</code>	<code>int(11)</code>	Irányítószámot tárolja	
<code>orszag</code>	<code>varchar(100)</code>	Ország nevét tárolja	
<code>telefonszam</code>	<code>varchar(100)</code>	Telefonszámot tárolja	

SQL

```
CREATE TABLE `felhasznalo_szallitasi_adatok` (
  `id` int(11) NOT NULL,
  `felhasznalo_id` int(11) NOT NULL,
  `szallitasi_cim1` varchar(200) NOT NULL,
  `szallitasi_cim2` varchar(200) DEFAULT NULL,
  `varos` varchar(100) NOT NULL,
  `iranyitoszam` int(11) NOT NULL,
  `orszag` varchar(100) NOT NULL,
  `telefonszam` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `felhasznalo_szallitasi_adatok`
  ADD PRIMARY KEY (`id`),
  ADD KEY `felhasznalo_id` (`felhasznalo_id`);

ALTER TABLE `felhasznalo_szallitasi_adatok`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=16;

ALTER TABLE `felhasznalo_szallitasi_adatok`
  ADD CONSTRAINT `felhasznalo_szallitasi_adatok_ibfk_1` FOREIGN KEY
(`felhasznalo_id`) REFERENCES `felhasznalo` (`id`);
```

1.4.3.4 A „jogosultsagok” tábla

A jogosultságok nevét tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
megnevezes	varchar(100)	Jogosultság nevét tárolja	

SQL

```
CREATE TABLE `jogosultsagok` (
    `id` int(11) NOT NULL,
    `megnevezes` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `jogosultsagok`
ADD PRIMARY KEY (`id`);

ALTER TABLE `jogosultsagok`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
```

1.4.3.5 A „log” tábla

Az eseményeket tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
felhasznalo_id	int(11)	Felhasználó azonosítója	FK
log_kategoria_id	int(11)	Esemény kategória azonosítója	FK
azonosito	varchar(100)	Esemény azonosítója	
idopont	timestamp	Esemény idejét tárolja	

SQL

```
CREATE TABLE `log` (
    `id` int(11) NOT NULL,
    `felhasznalo_id` int(11) NOT NULL,
    `log_kategoria_id` int(11) NOT NULL,
    `azonosito` varchar(100) NOT NULL,
    `idopont` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `log`
    ADD PRIMARY KEY (`id`),
    ADD KEY `felhasznalo_id`(`felhasznalo_id`),
    ADD KEY `log_kategoria_id`(`log_kategoria_id`);

ALTER TABLE `log`
    MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=52;

ALTER TABLE `log`
    ADD CONSTRAINT `log_ibfk_1` FOREIGN KEY (`log_kategoria_id`) REFERENCES `log_kategoriak`(`id`),
    ADD CONSTRAINT `log_ibfk_2` FOREIGN KEY (`felhasznalo_id`) REFERENCES `felhasznalo`(`id`);
```

1.4.3.6 A „log_kategoriak” tábla

Események kategóriáját tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
esemeny	varchar(100)	Esemény nevét tárolja	

SQL

```
CREATE TABLE `log_kategoriak` (
  `id` int(11) NOT NULL,
  `esemeny` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `log_kategoriak`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `log_kategoriak`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=19;
```

1.4.3.7 A „markak” tábla

Márkák adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
markanev	varchar(20)	Márka nevét tárolja	

SQL

```
CREATE TABLE `markak` (
  `id` int(11) NOT NULL,
  `markanev` varchar(20) COLLATE utf8_hungarian_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_hungarian_ci;
ALTER TABLE `markak`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `markak`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;
```

1.4.3.8 A „megrendeles” tábla

Megrendelések adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
felhasznalo_id	int(11)	Felhasználó zonosítója	FK
megrendeles_datuma	date	Megrendelés dátumát tároéja	
megrendeles_statusz	int(11)	Megrendelés státuszát tárolja	
szallitasi_cim	varchar(100)	Szállítási címet tárolja	
varos	varchar(100)	Város nevét tárolja	
iranyitoszam	int(11)	Irányítószámot tárolja	
orszag	varchar(100)	Ország nevét tárolja	
telefonszam	varchar(100)	Telefonszámot tárolja	

SQL

```
CREATE TABLE `megrendeles` (
  `id` int(11) NOT NULL,
  `felhasznalo_id` int(11) NOT NULL,
  `megrendeles_datuma` date NOT NULL,
  `megrendeles_statusz` int(11) NOT NULL,
  `szallitasi_cim` varchar(100) NOT NULL,
  `varos` varchar(100) NOT NULL,
  `iranyitoszam` int(11) NOT NULL,
  `orszag` varchar(100) NOT NULL,
  `telefonszam` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `megrendeles`
ADD PRIMARY KEY (`id`),
ADD KEY `felhasznalo_id` (`felhasznalo_id`);

ALTER TABLE `megrendeles`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

ALTER TABLE `megrendeles`
ADD CONSTRAINT `megrendeles_ibfk_2` FOREIGN KEY (`felhasznalo_id`) REFERENCES
`felhasznalo` (`id`);
```

1.4.3.9 A „megrendelt_termek” tábla

Megrendelt termékek adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
megrendeles_id	int(11)	Megrendelés azonosítója	FK
termek_id	int(11)	Termék azonosítója	FK
meret_id	int(11)	Méret azonosítója	FK
mennyiseg	int(11)	Mennyiséget tárolja	
ar	int(11)	Árat tárolja	

SQL

```
CREATE TABLE `megrendelt_termek` (
  `id` int(11) NOT NULL,
  `megrendeles_id` int(11) NOT NULL,
  `termek_id` int(11) NOT NULL,
  `meret_id` int(11) NOT NULL,
  `mennyiseg` int(11) NOT NULL,
  `ar` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `megrendelt_termek`
  ADD PRIMARY KEY (`id`),
  ADD KEY `megrendeles_id` (`megrendeles_id`),
  ADD KEY `termek_id` (`termek_id`),
  ADD KEY `meret_id` (`meret_id`);

ALTER TABLE `megrendelt_termek`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;

ALTER TABLE `megrendelt_termek`
  ADD CONSTRAINT `megrendelt_termek_ibfk_1` FOREIGN KEY (`termek_id`)
  REFERENCES `termek` (`id`),

  ADD CONSTRAINT `megrendelt_termek_ibfk_2` FOREIGN KEY (`megrendeles_id`)
  REFERENCES `megrendeles` (`id`),

  ADD CONSTRAINT `megrendelt_termek_ibfk_3` FOREIGN KEY (`meret_id`)
  REFERENCES `meretek` (`id`);
```

1.4.3.10 A „meretek” tábla

Méretek adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
megnevezes	varchar(4)	Méret nevét tárolja	
sorrend	int(11)	Sorrendet tárolja	

SQL

```
CREATE TABLE `meretek` (
    `id` int(11) NOT NULL,
    `megnevezes` varchar(4) COLLATE utf8_hungarian_ci NOT NULL,
    `sorrend` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_hungarian_ci;
ALTER TABLE `meretek`
    ADD PRIMARY KEY (`id`);
ALTER TABLE `meretek`
    MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
```

1.4.3.11 A „nem” tábla

Nemek adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
megnevezes	varchar(10)	Nem nevét tárolja	

SQL

```
CREATE TABLE `nem` (
  `id` int(11) NOT NULL,
  `megnevezes` varchar(10) COLLATE utf8_hungarian_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_hungarian_ci;
ALTER TABLE `nem`
  ADD PRIMARY KEY (`id`);
ALTER TABLE `nem`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;
```

1.4.3.12 A „termek” tábla

Termékek adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
nev	varchar(100)	Termék nevét tárolja	
cikkszam	varchar(100)	Cikkszámot tárolja	
marka_id	int(11)	Márka azonosítója	FK
leiras	text	Termék leírását tárolja	
nem_id	int(11)	Nem azonosítója	FK
ar	int(11)	Termék árát tárolja	
kep_nev	varchar(100)	Termék képének a nevét tárolja	

SQL

```
CREATE TABLE `termek` (
  `id` int(11) NOT NULL,
  `nev` varchar(100) NOT NULL,
  `cikkszam` varchar(100) NOT NULL,
  `marka_id` int(11) NOT NULL,
  `leiras` text NOT NULL,
  `nem_id` int(11) NOT NULL,
  `ar` int(11) NOT NULL,
  `kep_nev` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `termek`
  ADD PRIMARY KEY (`id`),
  ADD KEY `nem_id`(`nem_id`),
  ADD KEY `nem_id_2`(`nem_id`),
  ADD KEY `ar_2`(`ar`),
  ADD KEY `marka_id`(`marka_id`);

ALTER TABLE `termek`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=20;

ALTER TABLE `termek`
  ADD CONSTRAINT `termek_ibfk_1` FOREIGN KEY (`nem_id`) REFERENCES `nem`(`id`),
  ADD CONSTRAINT `termek_ibfk_2` FOREIGN KEY (`marka_id`) REFERENCES `markak`(`id`);
```

1.4.3.13 A „termek_kategoria” tábla

Termék kategóriák adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
nev	varchar(100)	Termék kategória nevét tárolja	

SQL

```
CREATE TABLE `termek_kategoria` (
    `id` int(11) NOT NULL,
    `nev` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
ALTER TABLE `termek_kategoria`
ADD PRIMARY KEY (`id`);
ALTER TABLE `termek_kategoria`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=16;
```

1.4.3.14 A „termek_kepek” tábla

Termék képeinek az adatait tárolja.

Mező neve	Típusa	Leírás	FK/PK
id	int(11)	Egyedi belső azonosító	PK
termek_id	int(11)	Termék azonosítója	FK
kep_név	varchar(100)	Kép nevét tárolja	

SQL

```
CREATE TABLE `termek_kepek` (
    `id` int(11) NOT NULL,
    `termek_id` int(11) NOT NULL,
    `kep_név` varchar(100) COLLATE utf8_hungarian_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_hungarian_ci;
ALTER TABLE `termek_kepek`
    ADD PRIMARY KEY (`id`),
    ADD KEY `termek_id` (`termek_id`);
ALTER TABLE `termek_kepek`
    MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
ALTER TABLE `termek_kepek`
    ADD CONSTRAINT `termek_kepek_ibfk_1` FOREIGN KEY (`termek_id`) REFERENCES
`termek` (`id`);
```

1.4.3.15 A „termek_meretek” tábla

Termékek és méretek közötti kötőtábla.

Mező neve	Típusa	Leírás	FK/PK
termek_id	int(11)	Termék azonosítója	FK
meret_id	int(11)	Méret azonosítója	FK

SQL

```
CREATE TABLE `termek_meretek` (
    `termek_id` int(11) NOT NULL,
    `meret_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_hungarian_ci;
ALTER TABLE `termek_meretek`
    ADD KEY `termek_id` (`termek_id`),
    ADD KEY `meret_id` (`meret_id`);
ALTER TABLE `termek_meretek`
    ADD CONSTRAINT `termek_meretek_ibfk_1` FOREIGN KEY (`termek_id`) REFERENCES
`termek` (`id`),
    ADD CONSTRAINT `termek_meretek_ibfk_2` FOREIGN KEY (`meret_id`) REFERENCES
`meretek` (`id`);
```

1.4.3.16 A „termek_termek_kategoria” tábla

Termékek és kategóriák közötti kötőtábla.

Mező neve	Típusa	Leírás	FK/PK
termek_id	int(11)	Termék azonosítója	FK
kategoria_id	int(11)	Kategória azonosítója	FK

SQL

```
CREATE TABLE `termek_termek_kategoria` (
    `termek_id` int(11) NOT NULL,
    `kategoria_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

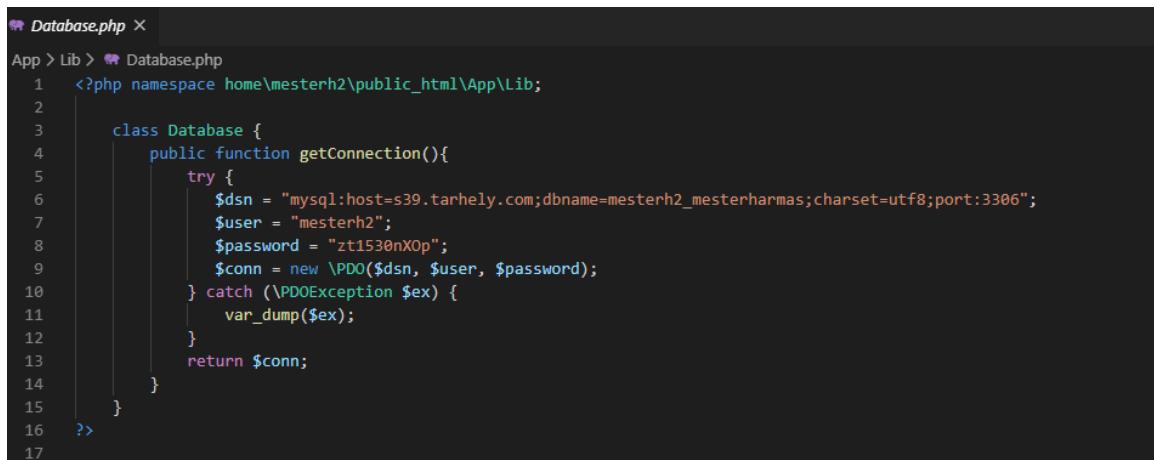
ALTER TABLE `termek_termek_kategoria`
    ADD CONSTRAINT `termek_termek_kategoria_ibfk_1` FOREIGN KEY (`termek_id`)
    REFERENCES `termek` (`id`),
    ADD CONSTRAINT `termek_termek_kategoria_ibfk_2` FOREIGN KEY (`kategoria_id`)
    REFERENCES `termek_kategoria` (`id`);
```

1.5 Osztályok, modulok ismertetése

A webes alkalmazás MVC (Model-View-Controller) felépítése alapján íródott, mely által a kód sokkal átláthatóbb, illetve értelmezhetőbb egy külsős fejlesztő számára. Számunkra ez az architektúra igen fontos, mert ezzel rengeteg időt és pénzt tudunk spórolni a programozás során, hisz így könnyebben átvészeltethetőek, illetve kitapinthatóbbak a hibák.

1.5.1 Adatbázis

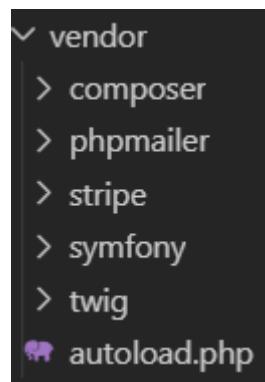
Az adatbázishoz való csatlakozás a Database.php fájlban történik PDO-val a getFunction() metóduson belül.



```
Database.php
App > Lib > Database.php
1  <?php namespace home\mesterh2\public_html\App\Lib;
2
3  class Database {
4      public function getConnection(){
5          try {
6              $dsn = "mysql:host=s39.tarhely.com;dbname=mesterh2_mesterharmas;charset=utf8;port:3306";
7              $user = "mesterh2";
8              $password = "zt1530nX0p";
9              $conn = new \PDO($dsn, $user, $password);
10         } catch (\PDOException $ex) {
11             var_dump($ex);
12         }
13         return $conn;
14     }
15 }
16 ?>
17
```

1.5.2 A „vendor” mappa

A *src/vendor* mappa tartalmazza az összes külső függőséget, amelyeket az alkalmazás használ, például a nyílt forráskódú keretrendszereket vagy más könyvtárakat.



1.5.3 A „composer.lock” fájl

A *src/composer.lock* fájl pedig a Composer által telepített függőségek konkrét verzióinak listáját tartalmazza, amelyek az alkalmazás stabil működését garantálják. Tartalmazza az összes külső függőséget, amelyeket az alkalmazás használ.

```
composer.lock > ...
1
2     "_readme": [
3         "This file locks the dependencies of your project to a known state",
4         "Read more about it at https://getcomposer.org/doc/01-basic-usage.md#installing-dependencies",
5         "This file is @generated automatically"
6     ],
7     "content-hash": "225173eee4634811ab0af13465f4e743",
8     "packages": [
9         {
10             "name": "phpmailer/phpmailer",
11             "version": "v6.8.0",
12             "source": {
13                 "type": "git",
14                 "url": "https://github.com/PHPMailer/PHPMailer.git",
15                 "reference": "df16b615e371d81fb79e506277faea67a1be18f1"
16             },
17             "dist": {
18                 "type": "zip",
19                 "url": "https://api.github.com/repos/PHPMailer/PHPMailer/zipball/df16b615e371d81fb79e506277faea67a1be18f1",
20                 "reference": "df16b615e371d81fb79e506277faea67a1be18f1",
21                 "shasum": ""
22             },
23             "require": {
24                 "ext-ctype": "*",
25                 "ext-filter": "*",
26                 "ext-hash": "*",
27                 "php": ">=5.5.0"
28             },
29             "require-dev": {
30                 "dealerdirect/phpcodesniffer-composer-installer": "^0.7.2",
31                 "doctrine/annotations": "^1.2.6 || ^1.13.3",
32                 "php-parallel-lint/php-console-highlighter": "^1.0.0",
33                 "php-parallel-lint/php-parallel-lint": "^1.3.2",
34                 "phpcompatibility/php-compatibility": "^9.3.5",
35                 "roave/security-advisories": "dev-latest",
36                 "squizlabs/php_codesniffer": "^3.7.1",
37                 "yousat/phpunit-polyfills": "^1.0.4"
38             },
39             "suggest": {
40                 "ext-mbstring": "Needed to send email in multibyte encoding charset or decode encoded addresses"
41             }
42         }
43     ]
44 
```

1.5.4 A „composer.json” fájl

A *src/composer.json* fájl a Composer függőségkezelő konfigurációs fájlya, amely az alkalmazás függőségeinek listáját és verziót tartalmazza.

```
{} composer.json > ...
1  {
2      "name": "edes_harmas/webshop",
3      "description": "Édes harmas",
4      "autoload": {
5          "psr-4": {
6              "home\\": "App/"
7          }
8      },
9      "require": {
10         "twig/twig": "^3.0",
11         "phpmailer/phpmailer": "^6.8.0",
12         "stripe/stripe-php": "^10.8"
13     }
14 }
15 }
```

1.5.5 A „index.php” fájl

Az *src/index.php* fájl a fő alkalmazás belépési pontja, amely az alkalmazás futását indítja.

Ez egy PHP kód, amely felállít egy egyszerű útválasztási rendszert. Az útválasztó az URL címeket és a hozzájuk tartozó függvényeket rendeli össze. A függvényeket a megfelelő osztályokban definiálták. Az osztályokat a kódban a `require_once` függvényekkel töltik be.

A szükséges fájlokat a kódban definiált konstansok segítségével töltik be. A fájlok elérési útvonala a projekt struktúrájától függ, és a ROOT konstans tartalmazza a projekt gyökérkönyvtárának elérési útvonalát.

A kód támogatja a HTTP GET és POST kéréseket, és az URL címekben is lehetnek paraméterek. Az útválasztók az URL címekre adott kérésekre válaszolnak, és a megfelelő függvényeket hívják meg. A függvények általában a felhasználói bemenet feldolgozásával és az adatbázis-műveletek végrehajtásával foglalkoznak.

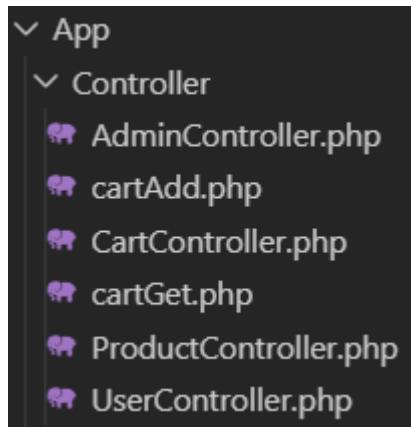
A router a PHP változóinak használatával érzékel bizonyos eseményeket (például a `$_GET` vagy `$_POST` változók), és ezekre a megfelelő függvényeket hívja meg. A kód a legtöbb esetben a HTTP válaszokat kezeli és generálja, például a HTML oldalakat és a JSON adatokat.

Az osztályok különböző funkciókat hajtanak végre, például a UserController a felhasználók kezelésével, az AdminController az adminisztrátorok kezelésével, a ProductController pedig az áruk kezelésével foglalkozik. A Router osztály felelős az URL címek kezeléséért és a megfelelő függvények hívásáért. A Request és a Response osztályok pedig az adatok kezelésére szolgálnak a kérések és a válaszok közötti kommunikáció során.

```
index.php
1  <?php
2  session_start();
3
4  require __DIR__ . '/vendor/autoload.php';
5  define('ROOT', __DIR__);
6  require_once(ROOT . '/App/Lib/Router.php');
7  require_once(ROOT . '/App/Lib/Request.php');
8  require_once(ROOT . '/App/Lib/Response.php');
9  require_once(ROOT . '/App/Controller/ProductController.php');
10 require_once(ROOT . '/App/Controller/CartController.php');
11 require_once(ROOT . '/App/Controller/UserController.php');
12 require_once(ROOT . '/App/Controller/AdminController.php');
13
14 use home\mesterh2\public_html\App\Lib\App;
15 use home\mesterh2\public_html\App\Lib\Router;
16 use home\mesterh2\public_html\App\Lib\Request;
17 use home\mesterh2\public_html\App\Lib\Response;
18 use home\mesterh2\public_html\App\Controller\Home;
19 use home\mesterh2\public_html\App\Controller\ProductController;
20 use home\mesterh2\public_html\App\Controller\CartController;
21 use home\mesterh2\public_html\App\Controller\UserController;
22 use home\mesterh2\public_html\App\Controller\AdminController;
23
24 //A program belépési pontja
25 Router::get('/', function () {
26     return ((new ProductController())->list());
27 });
28
29 Router::get('/products', function () {
30     return ((new ProductController())->list());
31 });
32
33 Router::get('/cartItems', function () {
34     (new CartController())->randomListazas();
35 });
36
37 Router::get('/ferfiOldal', function () {
38     (new ProductController())->ferfiTermekListazas(0);
39 });
40
```

1.5.6 A „Controller” mappa

Az *App/Controller* mappa tartalmazza az alkalmazás vezérlőit, amelyek kezelik a beérkező kéréseket és válaszokat generálnak.



1.5.6.1 Az „AdminController.php” fájl

Az *AdminController.php* egy PHP osztályt ír le, amely az Admin oldalak vezérlésére szolgál. Az osztály névterét és a függvényeket a namespace kulcsszó és a function definíciói határozzák meg. A kód használ néhány más osztályt is, például az *AdminDao* és a *UserDao* osztályokat, amelyek az adatbázis műveleteket végzik el.

Az osztályban szerepelnek különböző függvények, mint például az *adminLoginPage()*, *adminRegisterPage()*, *adminLogin()*, *adminPage()*, *adminLogout()*, *adminRegist()*, *verify_admin()*, *adminVerify()*, *adminProducts()*, *adminProductSearch()*, *productChange()* és *adminNewProduct()*. Ezek a függvények a különböző oldalak és műveletek vezérlésére szolgálnak.

Az osztály használja a Twig sablonmotort is, amely lehetővé teszi a HTML sablonok egyszerűbb és hatékonyabb kezelését.

Az osztály függvényei között a felhasználói bemenetet is ellenőrzik, például az adminLogin() és az adminRegist() függvényekben. Ha az adatok helyesek, akkor a megfelelő adatbázis műveletet hajtják végre, ellenkező esetben pedig a felhasználót visszairányítják az admin kezdőoldalra.

```
App > Controller > AdminController.php
1  <?php namespace home\mesterh2\public_html\App\Controller;
2
3  require_once(ROOT . '/App/Model/AdminDao.php');
4  require_once(ROOT . '/App/Model/UserDao.php');
5
6  use home\mesterh2\public_html\App\Model\AdminDao;
7  use home\mesterh2\public_html\App\Model\UserDao;
8  use Twig\Environment;
9  use Twig\Loader\FilesystemLoader;
10
11 class AdminController
12 {
13     public function adminLoginPage()
14     {
15         $twig = (new AdminController())->setTwigEnvironment();
16         echo $twig->render('admin/adminLogin.html.twig');
17     }
18
19     public function adminRegisterPage()
20     {
21         $twig = (new AdminController())->setTwigEnvironment();
22         echo $twig->render('admin/adminRegister.html.twig');
23     }
24
25     public function adminLogin()
26     {
27         if (isset($_POST['login']))
28         {
29             AdminDao::adminLogin();
30         } else {
31             header("Location: /admin");
32         }
33     }
34
35     public function adminPage()
36     {
37         $adminData = AdminDao::adminData();
38         $honapJovedelem = AdminDao::incomePerMonth();
39         $adminUserCount = AdminDao::adminAndUser();
40         $twig = (new AdminController())->setTwigEnvironment();
41
42         $honapJovedelem = AdminDao::incomePerMonth();
43         $adminUserCount = AdminDao::adminAndUser();
44         $adminData = AdminDao::adminData();
45         $twig = (new AdminController())->setTwigEnvironment();
46
47         echo $twig->render('admin/index.html.twig', ['admin'=>$adminData, 'bevetelek'=>$honapJovedelem, 'adminSzam'=>$adminUserCount["admin"]];
48     }
49
50     public function adminLogout() {
51         AdminDao::adminLogout();
52     }
53
54     public function adminRegist()
55     {
56         if (isset($_POST['regist']))
57         {
58             AdminDao::adminRegist();
59         } else {
60             header("Location: /admin");
61         }
62     }
63
64     public function verify_admin($adminToken, $userToken)
65     {
66         if (isset($adminToken)) {
67             $adminData = UserDao::userToken($adminToken);
68             $userData = UserDao::userToken($userToken);
69             $twig = (new AdminController())->setTwigEnvironment();
70             echo $twig->render('admin/admin_verify.html.twig', ['admin'=>$adminData, 'user'=>$userData]);
71         } else {
72             header("Location: /admin");
73         }
74     }
75
76     public function adminVerify()
77     {
78         if (isset($_POST['megerosit']) || isset($_POST["torol"])) {
79             AdminDao::adminVerify();
80         } else {
81         }
82     }
83 }
```

1.5.6.2 A „cartAdd.php” fájl

Ez egy PHP kód, ami a kosár funkcióval kapcsolatos tevékenységeket valósít meg. Az első sorban a "session_start" függvényt hívjuk meg, ami elindítja a munkamenetet, amely lehetővé teszi az adatok tárolását az egész oldal látogatása során.

Ezután egy üres tömböt hozunk létre, amely a visszatérési értékeket tartalmazza. Az "if" utasítással ellenőrizzük, hogy a szükséges adatok (productId, productAr, mennyiseg, meret és user_id) megfelelően vannak-e elküldve a POST metódussal a szerverhez. Ha igen, akkor elkezdjük a kosár műveleteket.

A következő sorokban először a bejövő adatokat formázzuk megfelelően (integer vagy string típusú változók), majd ellenőrizzük, hogy létezik-e már kosár. Ha nem, akkor létrehozzuk egy üres tömb formájában.

Ezután ellenőrizzük, hogy az adott termék méretével és azonosítójával már szerepel-e a kosárban. Ha nem, akkor hozzáadjuk az új tételelt, ha igen, akkor a meglévő tétel mennyiségét módosítjuk (hozzáadás vagy kivonás a mennyiségből). Ha az új mennyiség negatív lenne, akkor a darabszámot 0-ra állítjuk.

Az utolsó részben kiszedjük a kosárból azokat a tételeket, amelyek darabszáma 0 vagy negatív értékű, majd összesítjük a kosár összegét. Végül az eredményeket egy JSON objektumban küldjük vissza a kliensnek. Ha valamilyen adat hiányzik, akkor a "vissza" tömb "kod" értéke 100-ra állítódik.

```
App > Controller > cartAdd.php
1  <?php
2  session_start();
3  $vissza=[];
4  $vissza['kod']=0;
5  $vissza['dat']=[];
6  if (isset($_POST['productId'],$_POST['productAr'],$_POST['mennyiseg'],$_POST['meret'],$_SESSION['user_id'])) {
7      $id=(int)$_POST['productId'];
8      $ar=(int)$_POST['productAr'];
9      $db=(int)$_POST['mennyiseg'];
10     $meret=$_POST['meret'];
11     $mod=$_POST['mod'];
12     //ha nem létezik a kosár
13     //unset($_SESSION['kosar']);
14     if (!isset($_SESSION['kosar'])) {
15         $_SESSION['kosar']=[];//üres kosár
16     }
17     //keresés a kosában:id+méret
18     $n=count($_SESSION['kosar']);
19     $i=0;
20     $hol=-1;
21     while ($i<$n) {
22         if ($_SESSION['kosar'][$i]['id']==$id && $_SESSION['kosar'][$i]['meret']==$meret) {
23             $hol=$i;
24             $i=$n;
25         }
26         $i++;
27     }
28
29     if ($hol==-1) { //ha nincs ilyen id+meret
30         $_SESSION['kosar'][]=array('id'=>$id,'ar'=>$ar,'meret'=>$meret,'db'=>$db);
31     }else { //van a kosában, indexe hol, db szám módosítás
32         if ($mod=='hozzaad') {
33             $_SESSION['kosar'][$hol]['db'] += $db;
34         }else {
35             $_SESSION['kosar'][$hol]['db'] = $db;
36         }
37         if ($_SESSION['kosar'][$hol]['db']<0) {
38             $_SESSION['kosar'][$hol]['db'] = 0;
39         }
40     }
}
```

1.5.6.3 A „CartController.php” fájl

A Controller/CartController.php egy olyan osztályt ír le, amely a kosárhoz kapcsolódó feladatokat végzi el.

Az osztályban négy metódus található:

- A randomlistazas() metódus a kosárban lévő véletlenszerű termékeket listázza ki, és megjeleníti a kapcsolódó HTML oldalon a Twig sablonmotorral.
- A cartPay() metódus átirányítja a felhasználót a fizetéshez.
- A cartSuccess() metódus pedig az adatbázisban rögzíti a megrendelést.
- A deleteItem() metódus törli a terméket a session-ből.

```
App > Controller > CartController.php
1  <?php namespace home\mesterh2\public_html\App\Controller;
2
3  require_once(ROOT . '/App/Model/CartDao.php');
4  require_once(ROOT . '/App/Model/UserDao.php');
5
6  use home\mesterh2\public_html\App\Model\CartDao;
7  use home\mesterh2\public_html\App\Model\UserDao;
8  use Twig\Environment;
9  use Twig\Loader\FilesystemLoader;
10
11 class CartController
12 {
13
14     public function randomlistazas()
15     {
16         $data = CartDao::randomTermek();
17         $loggedin = UserDao::logedinUser();
18         $twig = (new CartController())->setTwigEnvironment();
19         if (empty($data)) {
20             echo $twig->render('cart/cartItems.html.twig', ['termek'=>0, 'loggedin'=>$loggedin]);
21         } else {
22             echo $twig->render('cart/cartItems.html.twig', ['termek'=>$data["termek"], 'dbok'=>$data["dbok"] , 'loggedin'=>$loggedin]);
23         }
24     }
25
26     public function cartPay() {
27         CartDao::cartPay();
28     }
29
30     public function cartSuccess() {
31         CartDao::cartSuccess();
32     }
33
34     public function deleteItem($deleteItem, $meretId) {
35         CartDao::deleteItem($deleteItem, $meretId);
36     }
37
38     public function setTwigEnvironment()
39     {
40         $loader = new FilesystemLoader('App\View');
```

1.5.6.4 A „cartGet.php” fájl

A Controller/cartGet.php egy PHP kód, amely egy ajax kérésre válaszol. A kód első sorában a session_start() függvény meghívásával kezdi a munkát, amely lehetővé teszi a PHP számára, hogy az aktuális munkamenetben tárolja az adatokat.

Ezután inicializál egy üres tömböt, \$vissza néven, amely visszaadja a választ a kliensnek. A következő sorokban két adattagot állít be a \$vissza tömbben: \$vissza['kod'] és \$vissza['adat']. A \$vissza['kod'] értéke alapértelmezetten 0, ami azt jelzi, hogy a munkamenetben lévő felhasználó be van jelentkezve.

Ezután megvizsgálja, hogy a felhasználó be van-e jelentkezve, azaz a \$_SESSION['user_id'] változó be van-e állítva. Ha igen, akkor megvizsgálja, hogy létezik-

e a kosár az aktuális munkamenetben, ha nem, akkor inicializál egy üres kosarat. Ezt követően kiszámolja a kosárban lévő tételek számát és az összegüket. Az \$osszegStr változóba beállítja a formázott összeget, amelyet a number_format() függvény segítségével hoz létre. Az összeg numeraikus értékét az \$osszegNum változóban tárolja.

Ha a felhasználó nincs bejelentkezve, akkor a \$vissza['kod'] értékét 100-ra állítja, ami azt jelzi, hogy a felhasználó nincs bejelentkezve.

Végül a választ az echo json_encode(\$vissza); sorral küldi vissza a kliensnek JSON formátumban. A JSON adatstruktúra az adatok könnyű kezelését teszi lehetővé a kliens oldalon.

```
App > Controller > cartGet.php
1  <?php
2  session_start();
3  $vissza=[];
4  $vissza[ 'kod']=0;
5  $vissza[ 'adat']=[];
6  if (isset($_SESSION[ 'user_id'])) {
7      //ha nem létezik a kosár
8      if (!isset($_SESSION[ 'kosar'])) {
9          $_SESSION[ 'kosar']=[];//üres kosár
10     }
11     //kosár értéke
12     $n=count($_SESSION[ 'kosar']);
13     $osszeg=0;
14     for ($i=0; $i < $n; $i++) {
15         $osszeg+=$_SESSION[ 'kosar'][ $i][ 'db']*$_SESSION[ 'kosar'][ $i][ 'an'];
16     }
17     $osszegStr=number_format($osszeg,0,',','');
18     $vissza[ 'adat']=array('tetel'=>$n,'osszeg'=>$osszegStr,'osszegNum'=>$osszeg);
19 }
20 else {
21     $vissza[ 'kod']=100;
22 }
23 echo json_encode($vissza);
24 ?>
```

1.5.6.5 A „ProductController.php” fájl

A Controller/ProductController.php egy PHP osztály, amely a webalkalmazás termékoldalait kezeli.

A ProductController osztály a home\mesterh2\public_html\App\Controller névtérbe van helyezve. Az osztály a következő módszereket tartalmazza:

- ferfiTermekListazas: Egy adott kategóriához tartozó férfi termékek listáját jeleníti meg a ProductDao::ferfiTermeket metódus használatával.
- noiTermekListazas: Egy adott kategóriához tartozó női termékek listáját jeleníti meg a ProductDao::noiTermeket metódus használatával.
- gyermekTermekListazas: Egy adott kategóriához tartozó gyermek termékek listáját jeleníti meg a ProductDao::gyermekTermeket metódus használatával.

- list: Az ajánlott termékek listáját jeleníti meg a ProductDao::ajanlottTermeket metódus használatával.
- productById: Egy adott termék oldalát jeleníti meg a ProductDao::productById metódus használatával. Ha a felhasználó nincs bejelentkezve, csak a termék adatai jelennek meg, ha be van jelentkezve, akkor a felhasználó adatai is megjelennek.
- search: A keresett termék adatait jeleníti meg a ProductDao::search metódus használatával. Ha nincs találat, akkor a felhasználót figyelmeztetik.
- cartItems: A felhasználó kosárának adatait jeleníti meg. Ha a felhasználó be van jelentkezve, akkor a felhasználó kosara jelenik meg, különben a felhasználó figyelmeztetést kap.

A megjelenítés a Twig sablonmotor segítségével történik.

```
App > Controller > ProductController.php
  1  <?php namespace home\mesterh2\public_html\App\Controller;
  2
  3  require_once(ROOT . '/App/Model/ProductDao.php');
  4  require_once(ROOT . '/App/Model/UserDao.php');
  5
  6  use home\mesterh2\public_html\App\Model\ProductDao;
  7  use home\mesterh2\public_html\App\View;
  8  use home\mesterh2\public_html\App\Model\UserDao;
  9  use Twig\Environment;
10  use Twig\Loader\FilesystemLoader;
11
12 class ProductController
13 {
14     public function ferfiTermekListazas($kategoria_id)
15     {
16
17         $data = ProductDao::ferfiTermek($kategoria_id);
18         $loggedin = UserDao::loggedinUser();
19         $twig = (new ProductController())->setTwigEnvironment();
20         echo $twig->render('products/ferfi_oldal.html.twig', ['termek'=>$data["termek"], 'kategoria'=>$data["kategoria"], 'aktualisKategoria'=>$kategoria_id]);
21     }
22
23     public function noiTermekListazas($kategoria_id)
24     {
25
26         $data = ProductDao::noiTermek($kategoria_id);
27         $loggedin = UserDao::loggedinUser();
28         $twig = (new ProductController())->setTwigEnvironment();
29         echo $twig->render('products/noi_oldal.html.twig', ['termek'=>$data["termek"], 'kategoria'=>$data["kategoria"], 'aktualisKategoria'=>$kategoria_id]);
30     }
31     public function gyermekTermekListazas($kategoria_id)
32     {
33
34         $data = ProductDao::gyermekTermek($kategoria_id);
35         $loggedin = UserDao::loggedinUser();
36         $twig = (new ProductController())->setTwigEnvironment();
37         echo $twig->render('products/gyermek_oldal.html.twig', ['termek'=>$data["termek"], 'kategoria'=>$data["kategoria"], 'aktualisKategoria'=>$kategoria_id]);
38     }
39     public function list()
40     {

```

1.5.6.6 A „UserController.php” fájl

A *Controller/UserController.php* egy PHP osztályt tartalmaz, amely a felhasználói műveletek vezérlését végzi egy webalkalmazásban. A namespace deklaráció azt jelzi, hogy az osztály a *home\mesterh2\public_html\App\Controller* névtérben található.

Az osztály három fontos komponenst tartalmaz: metódusok, követelmények, és a Twig sablonmotor használata. A metódusok közé tartozik az oldalak megjelenítése (*loginPage()*, *welcomePage()*, *shippingPage()*, stb.), az új felhasználók regisztrációja és bejelentkezése (*register()*, *login()*, *forgot_password()*, stb.), valamint az adatok frissítése (*shippingChange()*, *userChange()*, stb.).

A követelmények megadása azt jelzi, hogy az adott művelet elvégzéséhez bizonyos feltételeknek kell teljesülnie. Például a login() metódus csak akkor hajtódik végre, ha a "signin" POST változó értéke megegyezik a megfelelő értékkel.

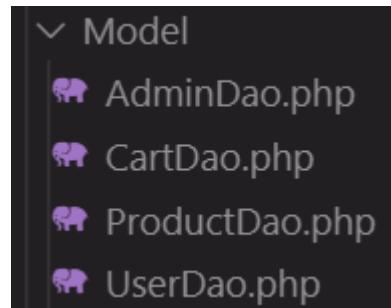
A Twig sablonmotor használatával az osztály képes dinamikusan létrehozni HTML oldalakat, amelyeket az adatokkal való dinamikus betöltéssel a felhasználó számára személyre szabhat. Az osztály a Twig környezetének beállításához egy setTwigEnvironment() metódust használ.

Az UserDao.php fájl importálása segítségével az osztály használja a UserDao osztály metódusait a felhasználói adatok kezelésére. Összességében ez az osztály segíti a webalkalmazás működését és vezérli a felhasználói interakciókat a felhasználói felületen.

```
App > Controller > UserController.php
1  <?php namespace home\mesterh2\public_html\App\Controller;
2
3  require_once(ROOT . '/App/Model/UserDao.php');
4
5  use home\mesterh2\public_html\App\Model\UserDao;
6  use Twig\Environment;
7  use Twig\Loader\FilesystemLoader;
8
9  class UserController
10 {
11     public function loginPage()
12     {
13         $twig = (new UserController())->setTwigEnvironment();
14         echo $twig->render('user/index.html.twig');
15     }
16
17     public function register()
18     {
19         if (isset($_POST['signup']))
20         {
21             UserDao::register();
22         } else {
23             header("Location: /loginPage");
24         }
25     }
26
27     public function login()
28     {
29         if (isset($_POST['signin']))
30         {
31             UserDao::login();
32         } else {
```

1.5.7 A „Model” mappa

Az *App/Model* az alkalmazás adatelérési rétegét (model) tartalmazó mappa, amelyek a különböző adatbázis műveleteket végzik.



1.5.7.1 Az „AdminDao.php” fájl

A Model/AdminDao.php egy PHP osztály, amely az adminisztrátorokat kezeli egy webalkalmazásban. Az osztályt a "home\mesterh2\public_html\App\Model" névterben definiálják, és a "Database.php" fájlt igényli a működéséhez.

A *adminLogin* metódus az adminisztrátorok bejelentkezését kezeli. A metódus ellenőrzi, hogy az adminisztrátori e-mail cím és jelszó helyes-e, majd ha igen, bejelentkezteti az adminisztrátort, és átirányítja az adminisztrátor oldalára. Ha az adatok helytelenek, akkor egy hibaüzenet jelenik meg.

```
App > Model > AdminDao.php
 1  <?php namespace home\mesterh2\public_html\App\Model;
 2
 3  require_once(ROOT . '/App/Lib/Database.php');
 4
 5  use home\mesterh2\public_html\App\Lib\Database;
 6  use PHPMailer\PHPMailer\PHPMailer;
 7  use PHPMailer\PHPMailer\SMTP;
 8  use PHPMailer\PHPMailer\Exception;
 9
10 class AdminDao
11 {
12     public static function adminLogin() {
13         $dbObj = new Database();
14         $conn = $dbObj->getConnection();
15
16         $email = $_POST["email"];
17
18         $sql = "SELECT * FROM felhasznalo WHERE email='$email' AND szatusz='1' AND jogosultsag_id='1'";
19         $prepare = $conn->prepare($sql);
20         $prepare->execute();
21         $row = $prepare->fetch(\PDO::FETCH_ASSOC);
22         $password = $row['jelszo'];
23
24         $passwordVerify = password_verify($_POST['password'], $password);
25
26         if ($passwordVerify) {
27             $_SESSION["user_id"] = $row['id'];
28             AdminDao::adminLog($row['id'], 2, "-");
29             header("Location: /adminPage");
30         } else {
31             echo "
32                 <script src='
```

Az *adminData* metódus lekérdezi az aktuális adminisztrátor adatait az adatbázisból. Az aktuális adminisztrátor azonosítóját a *\$_SESSION["user_id"]* változóban tároljuk. A metódus ellenőrzi, hogy a *\$_SESSION["user_id"]* változó beállítva van-e, és ha igen, lekéri az adminisztrátor adatait az adatbázisból és visszaadja azokat.

Az *adminLogout* metódus a jelenlegi adminisztrátor kijelentkezését kezeli. Az adminisztrátor kilépés előtt naplózza a kilépési eseményt, majd törli a *\$_SESSION* változókat, majd átirányítja az adminisztrátorok bejelentkezési oldalára.

Az *adminRegist* metódus az új adminisztrátorok regisztrációját kezeli. A metódus ellenőrzi, hogy az adott e-mail cím már létezik-e az adatbázisban, majd ha nem, regisztrálja az új adminisztrátort. Az új adminisztrátor jelszavát kódolja a *password_hash* függvény segítségével, majd beilleszti az adatbázisba. Ha az e-mail cím már létezik, vagy a jelszavak nem egyeznek meg, akkor hibaüzenet jelenik meg.

```
App > Model > AdminDao.php
  72     AdminDao::adminLog($id, 4, "-");
  73     session_unset();
  74     session_destroy();
  75     header("Location: /admin");
  76   }
  77
  78   public static function adminRegist() {
  79     require 'vendor/autoload.php';
  80
  81     error_reporting(0);
  82
  83     $dbObj = new Database();
  84     $conn = $dbObj->getConnection();
  85
  86     $username = $_POST["username"];
  87     $email = $_POST['email'];
  88     $password = $_POST['password'];
  89     $cpassword = $_POST['cpassword'];
  90     $token = bin2hex(random_bytes(5));
  91     $status = 0;
  92     $jogosultsag_id = 1;
  93
  94     $sql = "SELECT email FROM felhasznalo WHERE email='$email'";
  95     $stmt = $conn->prepare($sql);
  96     $stmt->execute();
  97     $count = $stmt->fetchColumn();
  98
  99     if ($password !== $cpassword) {
 100       echo "
 101         <script src='
 102           https://cdn.jsdelivr.net/npm/sweetalert2@11.7.2/dist/sweetalert2.all.min.js
 103         '></script>
```

Az *adminVerify* metódus, ami egy adatbázis-kapcsolatot hoz létre és ellenőrzi az adminisztrátor által elküldött felhasználói adatokat. Az adminisztrátor által elküldött adatok közé tartozik a felhasználó azonosítója, az állapota (az adminisztrátor által jóváhagyott vagy sem) és az e-mail címe.

Ha a felhasználó állapota még nem jóváhagyott, az adminisztrátor dönthet arról, hogy elfogadja vagy elutasítja a kérelmet. Ha elfogadja, akkor a felhasználó állapota átállítódik, és e-mailt kap a jóváhagyásról. Ha az e-mail elküldése sikertelen, akkor az adminisztrátornak erről értesítést küld a rendszer.

Ha az adminisztrátor elutasítja a kérelmet, akkor a felhasználó e-mailt kap az elutasításról.

A metódus a PHPMailer nevű külső könyvtárat használja az e-mailek küldéséhez, és a SweetAlert nevű JavaScript könyvtárat használja az üzenetek megjelenítéséhez a felhasználói felületen. A metódus végén az AdminDao osztály egy másik metódusát hívja meg a tevékenységről történő naplázáshoz.

Az *osszesTermek* metódus a termékek lekérdezésére szolgál az adatbázisból. Az első három paraméter segítségével szűrhetők a termékek nem, kategória és márka szerint.

A metódus visszatérési értéke egy tömb, amely tartalmazza a lekért termékeket, a nemeket, a kategóriákat és a márkkákat is.

A *searchProduct* metódus a termékek keresésére szolgál az adatbázisban. Az eljárás először lekérdezi a keresőkifejezésre illeszkedő termékeket, majd ha talál ilyeneket, akkor a keresési feltételeket alkalmazva lekérdezi a teljes termékinformációt is. Ha nem talál egyezést, akkor egy JavaScript alapú üzenetet jelenít meg, amely arra kéri a felhasználót, hogy térjen vissza az előző oldalra.

Az *nemek_marka* metódus az adatbázisban található nemek és márkkák lekérdezésére szolgál. A metódus visszatérési értéke egy tömb, amely tartalmazza a nemeket és a márkkákat.

Az *productById* egy adatbázis lekérdezést végez az adott ID-jú termékről és annak többi adatairól (neve, cikkszáma, ára, stb.) a hozzá tartozó táblából, mint például a márka, a neme és a termék-kategória. A metódus visszaadja ezeket az adatokat egy tömbben, amely tartalmazza a termék, a neme, a termék-kategória és a márka részletes adatait is. Továbbá, a metódus betölti a termékhez tartozó képeket, amelyeket a *termek_kepek* táblában tárolnak.

Az *update* metódus egy adatbázis frissítést hajt végre az adott termékről az űrlapon megadott adatok alapján. Az adatbázisban található több táblában is módosításokat végez. A metódus az adatokat POST kérésen keresztül kapja meg, és azokat az adatbázisban tárolja. Az új kép feltöltésekor a metódus ellenőrzi, hogy a fájl megfelelő típusú és méretű-e, majd ha az általános ellenőrzések sikeresek, akkor a fájlt áthelyezi az App/images/Clothes/Men/ mappába és a termékhez kapcsolódó adatbázis rekordot is frissíti.

```
App > Model > AdminDao.php
738
739
740     public static function update() {
741         $dbObj = new Database();
742         $conn = $dbObj->getConnection();
743
744         $id = $_POST['id'];
745         $nev = $_POST['nev'];
746         $cikkszam = $_POST['cikkszam'];
747         $marka = $_POST['marka'];
748         $leiras = $_POST['leiras'];
749         $nem = $_POST['nem'];
750         $ar = $_POST['ar'];
751         $termek_kategoria = $_POST['termek_kategoria'];
752
753         if ($termek_kategoria != 1) {
754             $sql = "UPDATE termek_termek_kategoria SET kategoria_id='$termek_kategoria' WHERE termek_id='$id' AND";
755             $statement = $conn->prepare($sql);
756             $statement->execute();
757         }
758
759         $sql = "UPDATE `termek` SET nev='$nev',cikkszam='$cikkszam',marka_id='$marka',leiras='$leiras',nem_id='$nem',ar='$ar'";
760         $statement = $conn->prepare($sql);
761         $statement->execute();
762
763         $targetDir = "App/images/Clothes/Men/";
764         $kep = basename($_FILES['kep'][ "name"]);
765         $targetFilePath = $targetDir . $kep;
766         $extension = strtolower(pathinfo($kep, PATHINFO_EXTENSION));
767         $imgExtArr = ['jpg', 'jpeg', 'png'];
768         $sql = "SELECT * FROM termek WHERE id='$id'";
769         $statement = $conn->prepare($sql);
```

A *deleteImage* egy kép törlésére szolgál egy adatbázisból és egy mappából is. Az első lépés a kapcsolat létrehozása az adatbázissal, majd a megfelelő mappában található fájl elérési útjának meghatározása. Ezután a képet tartalmazó terméket keresik az adatbázisban, és megszámolják, hogy hány ilyen kép található. Ezután megnézik, hogy a kép a termékhez van-e csatolva. Ha igen, akkor az adatbázis frissítése, azaz a kép törlése az adott termékről. Ha nem, akkor a kép csak a képek táblázatából lesz törölve. Mindkét esetben a függvény visszajelzést ad az eredményről és frissíti az adminisztrátor naplóját a történésről. A visszajelzés egy SweetAlert ablakban jelenik meg, amelynek tartalma sikeres vagy sikertelen törlés esetén különbözik. Ha a felhasználó az OK gombra kattint, akkor visszatér a korábbi oldalra, különben az oldal előzményeire kattintva tér vissza.

A *newProduct* új termék hozzáadásához használunk az adatbázishoz. A függvény létrehoz egy kapcsolatot az adatbázissal, majd az űrlapból származó adatokat menti az adatbázisba. A feltölti a képet a mappába, majd az adatbázisban eltárolja a kép nevét és a termék azonosítóját. A függvény beállítja a termék-kategóriát és a méreteket is. Végül az adminisztrátor művelet naplózza a műveletet és egy üzenetet jelenít meg az eredményről.

A *deleteById* függvény törli az adatbázisból egy termékhez kapcsolódó összes adatot, beleértve a megrendelt termékeket, a termék méretekét, a termék kategóriáit és a termék képeit is. A függvény a megadott termék ID alapján dolgozik, és végrehajtja a megfelelő adatbázis műveleteket, majd a képeket is törli a szerverről. Végül egy sikeres üzenetet jelenít meg a felhasználónak.

A *adminProductCategories* függvény a termék kategóriák lekérdezésére szolgál az adatbázisból.

A *newProductCategory* függvény új termék kategóriát hoz létre az adatbázisban a felhasználó által megadott névvel, majd sikeres feltöltési üzenetet jelenít meg a felhasználónak.

A *deleteCategory* törli a megadott termékkategóriát az adatbázisból. A metódus először lekérdezi a kategória nevét és rögzíti az admin tevékenységi naplójában. Ezután törli az adatbázisból a termékkategória összes hivatkozását a termékekhez és magát a kategóriát is. Végül egy megerősítő üzenetet jelenít meg a felhasználónak és visszatér a korábbi oldalra.

A *productCategoryChange* lekérdezi az adatbázisból a megadott termékkategória nevét és visszaadja azt.

Az *updateCategory* frissíti a megadott termékkategória nevét az adatbázisban. A metódus először lekéri az új nevet a felhasználótól, majd frissíti az adatbázisban. Ezután rögzíti az admin tevékenységi naplójában a módosítást és megerősítő üzenetet jelenít meg a felhasználónak. Végül visszatér a korábbi oldalra.

Az *adminBrands* függvény egy adatbázis-lekérdezést hajt végre az összes márka lekérdezésére a márkok táblából, majd visszatér a lekérdezett adatokkal.

Az *newBrand* függvény egy új márka hozzáadását teszi lehetővé. Először csatlakozik az adatbázishoz, majd beszűr egy új rekordot a márkkák táblába a megadott név alapján. Ezután egy SweetAlert üzenetet jelenít meg az eredményről.

A *changeBrand* függvény egy márka adatainak szerkesztésére használatos. Az adott márka adatait kérdezi le a márkkák táblából az adott brandId-val megegyező ID alapján, majd visszatér a lekérdezett adatokkal.

Az *deleteBrand* függvény törli az adott deleteBrandId-vel megegyező ID-jú márkat a márkkák táblából, majd egy SweetAlert üzenetet jelenít meg az eredményről. A törlés előtt az adminisztrátor logolja az eseményt, beleértve a törölt márka nevét is.

Az *updateBrand* függvény frissíti az adatbázisban található márkkák nevét a kapott adatok alapján. Ehhez létrehoz egy adatbázis kapcsolatot, majd az id és nev POST változók értékét használva végrehajtja az SQL lekérdezést, majd egy adminisztrátori naplzási bejegyzést hoz létre és végül egy üzenetet jelenít meg a felhasználónak.

A *users* függvény lekérdezést hajt végre az adatbázisban található felhasználókról. A jogosultság azonosítóját használja, hogy szűrje az eredményeket, és a lekérdezett adatokat objektumok formájában tériti vissza.

A *deleteUser* függvény egy felhasználó törlésére szolgál az adatbázisból. A függvény létrehoz egy adatbázis objektumot, majd lekérdezi a felhasználó adatait az adatbázisból a felhasználó azonosítója alapján. Ha a felhasználó email címe "mester.harmas.webshop@gmail.com", akkor egy figyelmeztető üzenet jelenik meg a felhasználónak, és nem törli a felhasználó adatait. Ha a felhasználó email címe nem "mester.harmas.webshop@gmail.com", akkor a függvény elküld egy e-mailt a felhasználónak, hogy értesítse őket a fiókjuk törléséről, majd töri a felhasználó adatait az adatbázisból. A függvény végén egy üzenet jelenik meg a felhasználónak, hogy tájékoztassa őket a folyamat sikeres befejezéséről.

Az *adminOrders* függvény lekéri az összes megrendelést és annak termékeit az adatbázisból. A lekérdezés a megrendelt_termek, megrendeles, termek és meretek táblákból történik, és az eredményt az ORDER BY kulcsszóval rendezi dátum és státusz szerint.

Az *orderById* függvény egy adott megrendelés adatait kéri le az adatbázisból az azonosító alapján. A lekérdezés a megrendeles és felhasznalo táblákból történik.

Az *orderProductById* függvény egy adott megrendelés termékeit kéri le az adatbázisból az azonosító alapján. A lekérdezés a megrendelt_termek, megrendeles, termek és meretek táblákból történik.

Az *orderSend* függvény lekéri az adatbázisból az adott azonosítójú rendelést és a hozzá tartozó termékeket. Ha a rendelés állapota "0" (még nem lett feldolgozva), akkor e-mailt küld a vevőnek és a webáruház tulajdonosának a rendeléssel kapcsolatos információkkal. Ha az e-mail küldése sikertelen, akkor egy figyelmeztető üzenet jelenik

meg a felhasználónak. Az adatbázisban frissíti a rendelés állapotát "1"-re (feldolgozva). A termékek eladásából származó jövedelmet hozzáadja a webáruház bevételéhez.

Az *orderDelete* funkció feladata egy megadott rendelés törlése az adatbázisból. Először létrehoz egy adatbázis kapcsolatot, majd lekérdezi a megrendelés adatait az AdminDao::orderById() metódus segítségével. Ha a megrendelés státusza 0 (nem teljesíthető), akkor e-mailt küld az ügyfélnek a PHPMailer használatával, majd a funkció törli a megrendeléshez kapcsolódó adatokat az adatbázisból és visszajelz a felhasználónak egy megerősítő üzenettel. Ha az e-mail küldése nem sikerült, akkor az üzenet helyett egy hibaüzenet jelenik meg a felhasználó számára. Végül az AdminDao::adminLog() metódus hívásával rögzíti az adminisztrátor tevékenységét az adatbázisban.

A *jogosultsagok* függvény lekérdezi az összes "jogosultságok" táblából származó rekordot az adatbázisból, és azokat visszatéríti.

Az *adminUserSearch* függvény, egy keresést végez a "felhasznalo" táblában a POST kérésben kapott lekérdezés alapján. Ha vannak találatok, akkor lekéri a felhasználók adatait a "jogosultsagok" táblából és azokat visszatéríti. Ha nincs találat, akkor egy SweetAlert hibaüzenet jelenik meg, és a felhasználó visszatérhet az előző oldalra.

Az *adminLog* függvény, egy új bejegyzést ad hozzá a "log" táblához a megadott felhasználói azonosítóval, log-kategóriával és időbélyeggel.

Az *adminLogs* visszaadja az összes naplóbejegyzést az adatbázisból a megadott kategóriában (ha van ilyen), és időrendi sorrendben rendezve.

Az *adminLogDelete* törli az adatbázisból a megadott naplóbejegyzést, majd visszatér az előző oldalra.

Az *adminLogCategoryDelete* metódus törli az összes naplóbejegyzést a megadott kategóriából az adatbázisból, majd visszatér az előző oldalra.

Az *adminLogCategories* egy adatbázis-kérést hajt végre, hogy az összes log kategóriát lekérdezze az adatbázisból. Az adatbáziskapcsolatot először inicializálja, majd lekérdezi az összes adatot a "log_kategoriak" táblából, majd visszatér az összes talált eredménnyel.

Az *incomeAdd* szintén inicializálja az adatbáziskapcsolatot, majd megvizsgálja a \$datum paraméter hónapját. Aztán lekérdezi az adatbázisból az adott hónapra vonatkozó bevételt az evi_bevetel táblából, majd hozzáadja az új bevételt a régi összeghez. Végül az új összeget frissíti az adatbázisban.

Az *incomePerMonth* visszaadja az év bevételét. Ehhez lekérdezi az adatbázisból a tavalyi évet és az idei évet, majd ha már túl vagyunk az előző éven, nullázza az idei évet az adatbázisban.

Az *adminAndUser* visszaadja az adminisztrátorok és vásárlók számát az adatbázisban.

1.5.7.2 A „CartDao.php” fájl

A *randomTermeket* metódus akkor fut, ha a kosárban vannak elemek. Először ellenőrzi, hogy a kosár nem üres (*isset(\$_SESSION['kosar'])* && *count(\$_SESSION['kosar'])>0*). Ha van termék a kosárban, akkor kapcsolódik az adatbázishoz, és lekérdezi az adatokat a termek, termek_meretek, meretek és markak táblákból, ahol a feltételnek megfelelőek. A feltétel a kosárban lévő elemeket ellenőrzi az id és meret attribútumok alapján. A visszatérési érték egy tömb lesz, amely tartalmazza a termékeket és azok darabszámát (*\$ret['termek']* és *\$ret['dbok']*). Ha a kosár üres, akkor visszatérési értéke false lesz.

```
App > Model > CartDao.php
1  <?php namespace home\mesterh2\public_html\App\Model;
2
3  require_once(ROOT . '/App/Lib/Database.php');
4
5  use home\mesterh2\public_html\App\Lib\Database;
6  use PHPMailer\PHPMailer\PHPMailer;
7  use PHPMailer\PHPMailer\SMTP;
8  use PHPMailer\PHPMailer\Exception;
9
10 class CartDao
11 {
12     public static function randomTermeket()
13     {
14         if (isset($_SESSION['kosar']) && count($_SESSION['kosar'])>0) {
15             $dbObj = new Database();
16             $conn = $dbObj->getConnection();
17             $felt='';
18             $dbok=[];
19             $n=count($_SESSION['kosar']);
20             foreach ($_SESSION["kosar"] as $kosar) {
21                 $id=$kosar["id"];
22                 $meretId=$kosar["meret"];
23                 $kulcs=$id.'_'.$meretId;
24                 $felt.= "(tm.termek_id='".$id' AND tm.meret_id = '$meretId') OR ";
25                 $dbok[$kulcs]=$kosar["db"];
26             }
27             $felt.= "0";
28             $sql = "SELECT t.*, m.megnevezes AS meret_nev, ma.markanev, m.id AS meret_id
29                   FROM `termek` AS t
30                   INNER JOIN `termek_meretek` AS tm ON tm.termek_id=t.id
31                   INNER JOIN `meretek` AS m ON m.id = tm.meret_id
32                   INNER JOIN `markak` AS ma ON ma.id=t.marka_id
33         }
34     }
35 }
```

A *cartPay* metódus a fizetés folyamatát valósítja meg. Először kapcsolódik az adatbázishoz, és lekéri a felhasználó szállítási adatait. Ha a szállítási adatok hiányoznak, akkor a felhasználót visszairányítja az előző oldalra, és megjelenít egy üzenetet, amely arra kéri a felhasználót, hogy töltse ki a szállítási adatokat. Ha a kosár üres, akkor hasonló üzenetet jelenít meg. Ha van elem a kosárban, akkor a teljes árat kiszámítja az elemek árai és darabszámai alapján. Ha az összeg nagyobb, mint 1 millió Ft, akkor megjelenít egy figyelmeztető üzenetet. Ha a kosár összege elfogadható, akkor a Stripe API-t használva létrehoz egy fizetési űrlapot.

A *cartSuccess* a megrendelés befejezésekor fut le, amely adatbázisba menti a megrendelést és a megrendelt termékeket.

```
App > Model > CartDao.php
177     }
178 }
179
180 public static function cartSuccess() {
181     $dbObj = new Database();
182     $conn = $dbObj->getConnection();
183
184     $id = $_SESSION['user_id'];
185     $sql = "SELECT * FROM felhasznalo_szallitasi_adatok WHERE felhasznalo_id='$id'";
186     $statement = $conn->prepare($sql);
187     $statement->setFetchMode(\PDO::FETCH_OBJ);
188     $statement->execute();
189     $szallitasiAdat = $statement->fetch();
190
191     $felhasznaloId = $szallitasiAdat->felhasznalo_id;
192     $szallitasiCim = $szallitasiAdat->szallitasi_cim1;
193     $varos = $szallitasiAdat->varos;
194     $iranyitoszam = $szallitasiAdat->iranyitoszam;
195     $orszag = $szallitasiAdat->orszag;
196     $telefonszam = $szallitasiAdat->telefonszam;
197
198     $sql = "INSERT INTO `megrendeles`(`felhasznalo_id`, `megrendeles_datuma`, `megrendeles_statusz`, `szallitasi_cim1`, `varos`, `iranyitoszam`, `orszag`, `telefonszam`, `date_default_timezone_set('Europe/Budapest')`, `datum` = date('Y-m-d'))";
199     $statement = $conn->prepare($sql);
200     $statement->execute();
201
202     date_default_timezone_set("Europe/Budapest");
203     $datum = date("Y-m-d");
204
205
206     $sql = "SELECT id FROM megrendeles WHERE felhasznalo_id='$felhasznaloId' AND megrendeles_datuma='$datum'";
207     $statement = $conn->prepare($sql);
208     $statement->execute();
```

A *deleteItem* függvény egy kosár elemből történő törlést hajt végre.

1.5.7.3 A „ProductDao.php” fájl

A *ferfiTermek* a férfi termékek keresésére szolgál. A keresési eredményekben szerepel a termék neve, cikkszáma, leírása, megnevezése, képének neve és ára. A megjelenítendő termékek kategóriája azonosító alapján van szűrve. A függvény visszatérési értéke egy tömb, amely a termék és a kategória eredményeit tartalmazza.

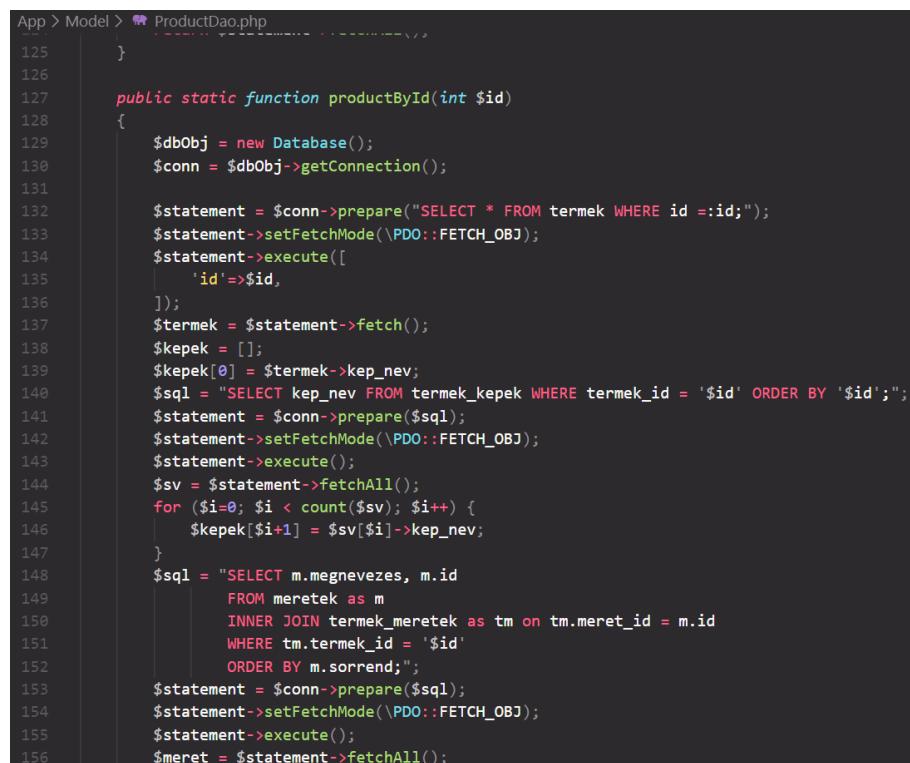
```
App > Model > ProductDao.php
1  <?php namespace home\mesterh2\public_html\App\Model;
2
3  require_once(ROOT . '/App/Lib/Database.php');
4
5  use home\mesterh2\public_html\App\Lib\Database;
6  use PHPMailer\PHPMailer\PHPMailer;
7  use PHPMailer\PHPMailer\SMTP;
8  use PHPMailer\PHPMailer\Exception;
9
10 class ProductDao
11 {
12     public static function ferfiTermek($kategoria_id)
13     {
14         $dbObj = new Database();
15         $conn = $dbObj->getConnection();
16         if ($kategoria_id == 0) {
17             $felt = "";
18         } else {
19             $felt = " AND tk.id = '$kategoria_id'";
20         }
21         $sql = "SELECT DISTINCT t.id,t.nev,t.cikkszam,t.leiras,tn.megnevezes,t.kep_nev,t.ar
22             FROM termek as t
23             INNER JOIN nem as tn ON t.nem_id=tn.id
24             INNER JOIN termek_kategoria as ttk on ttk.termek_id = t.id
25             INNER JOIN termek_kategoria as tk on tk.id = ttk.kategoria_id
26             WHERE tn.megnevezes = 'férfi' $felt;";
27         $statement = $conn->prepare($sql);
28         $statement->setFetchMode(\PDO::FETCH_OBJ);
29         $statement->execute();
30         $termek = $statement->fetchAll();
31         $sql = "SELECT DISTINCT tk.id, tk.nev
32             FROM termek_kategoria as tk"
```

A *noiTermek* a női termékek keresésére szolgál. A keresési eredményekben szerepel a termék neve, cikkszáma, leírása, megnevezése, képének neve és ára. A megjelenítendő termékek kategóriája azonosító alapján van szűrve. A függvény visszatérési értéke egy tömb, amely a termék és a kategória eredményeit tartalmazza.

A *gyermekTermek* az adatbázisból kérdez le adatokat, amelyek gyermektermékeknek felelnek meg. A metódus egy kategória azonosítót kap paraméterként, amely alapján az adatokat lekérdezi. Ha a kapott kategória azonosító értéke 0, akkor az összes gyermekterméket lekéri, ha nem 0, akkor csak az adott kategóriába tartozó gyermektermékeket. Az adatbázisból lekérdezett adatok két tömbben kerülnek visszaadásra a metódus által: termek és kategoria.

A *ajanlottTermek* véletlenszerűen választ ki 7 terméket az adatbázisból, majd ezeket a termékeket adja vissza.

A *productById* az adatbázisból lekérdez egy terméket a megadott id alapján, majd visszaadja az összes információt erről a termékről, beleértve a termék nevét, cikkszámát, leírását, képeit és méreteit is. Az információkat tömbben adja vissza.



```
App > Model > ProductDao.php
125     }
126
127     public static function productById(int $id)
128     {
129         $dbObj = new Database();
130         $conn = $dbObj->getConnection();
131
132         $statement = $conn->prepare("SELECT * FROM termek WHERE id =:id;");
133         $statement->setFetchMode(\PDO::FETCH_OBJ);
134         $statement->execute([
135             'id' =>$id,
136         ]);
137         $termek = $statement->fetch();
138         $kepek = [];
139         $kepek[0] = $termek->kep_nev;
140         $sql = "SELECT kep_nev FROM termek_kepek WHERE termek_id = '$id' ORDER BY '$id'";
141         $statement = $conn->prepare($sql);
142         $statement->setFetchMode(\PDO::FETCH_OBJ);
143         $statement->execute();
144         $sv = $statement->fetchAll();
145         for ($i=0; $i < count($sv); $i++) {
146             $kepek[$i+1] = $sv[$i]->kep_nev;
147         }
148         $sql = "SELECT m.megnevezes, m.id
149               FROM meretek as m
150                 INNER JOIN termek_meretek as tm on tm.meret_id = m.id
151                   WHERE tm.termek_id = '$id'
152                     ORDER BY m.sorrend;";
153         $statement = $conn->prepare($sql);
154         $statement->setFetchMode(\PDO::FETCH_OBJ);
155         $statement->execute();
156         $meret = $statement->fetchAll();
```

A *search* a felhasználó által megadott keresési lekérdezést használja a termékek keresésére az adatbázisban. Visszaadja az összes találatot, ha van találat, vagy egy "Nincs ilyen" üzenetet, ha nincs találat.

1.5.7.4 A „UserDao.php” fájl

A *register* függvény a felhasználó regisztrációját kezeli. Létrehoz egy adatbázis-kapcsolatot és beolvassa a regisztráció során megadott felhasználói adatokat. Az adatok ellenőrzése után ellenőrzi, hogy a két megadott jelszó megegyezik-e, és ha nem, akkor hibaüzenetet jelenít meg. Ha a megadott e-mail cím már létezik az adatbázisban, akkor is hibaüzenet jelenik meg. Ha minden adat helyes, akkor egy megerősítő e-mailt küld az adott e-mail címre, amely tartalmaz egy megerősítő linket. A függvény végén egy üzenet jelenik meg a felhasználónak, hogy a megerősítő e-mail elküldve, majd az oldal visszairányítja a felhasználót a bejelentkezési oldalra.

```
App > Model > UserDao.php
1  <?php namespace home\mesterh2\public_html\App\Model;
2
3  require_once(ROOT . '/App/Lib/Database.php');
4  require_once(ROOT . '/App/Model/AdminDao.php');
5
6  use home\mesterh2\public_html\App\Lib\Database;
7  use home\mesterh2\public_html\App\Model\AdminDao;
8  use PHPMailer\PHPMailer\PHPMailer;
9  use PHPMailer\PHPMailer\SMTP;
10 use PHPMailer\PHPMailer\Exception;
11
12 class UserDao
13 {
14     public static function register() {
15         require 'vendor/autoload.php';
16
17         error_reporting(0);
18
19         $dbObj = new Database();
20         $conn = $dbObj->getConnection();
21
22         $full_name = $_POST["signup_full_name"];
23         $email = $_POST["signup_email"];
24         $password = $_POST['signup_password'];
25         $cpassword = $_POST['signup_password2'];
26         $token = bin2hex(random_bytes(5));
27         $status = 0;
28         $jogosultsag_id = 2;
29
30         $sql = "SELECT email FROM felhasznalo WHERE email='$email'";
31         $stmt = $conn->prepare($sql);
32         $stmt->execute();
```

A *login* függvény ellenőrzi a felhasználói adatokat a bejelentkezéshez, majd ha a megadott jelszó helyes, akkor bejelentkezteti a felhasználót, eltárolja az id-ját a session változóban, naplózza az adminisztrátori tevékenységet, majd átirányítja a felhasználót a termékek oldalra. Ha a bejelentkezési adatok helytelenek, akkor egy figyelmezhető üzenetet jelenít meg a felhasználónak, majd átirányítja őt a bejelentkezési oldalra.

A *verify_email* függvény ellenőrzi a regisztrációkor megadott e-mail címmel kapott ellenőrző token-t, majd ha az megfelelő, akkor a felhasználó sztátuszát frissíti az adatbázisban, eltárolja az id-ját a session változóban és átirányítja a felhasználót a termékek oldalra. Ha nincs érvényes session változó vagy token, akkor a felhasználó a bejelentkezési oldalra irányítódik.

A *logout* függvény kijelentkezteti a felhasználót, naplózza az adminisztrátori tevékenységet, majd visszairányítja a felhasználót a termékek oldalra.

A *forgotPassword* függvény ellenőrzi, hogy az űrlap a jelszóvisszaállító gombbal van-e elküldve, majd ellenőrzi az adatbázisban, hogy az adott e-mail cím regisztrált-e. Ha igen, akkor e-mailben elküldi a jelszóvisszaállító linket, ha nem, akkor hibaüzenetet jelenít meg.

A `resetPassword` függvény az e-mailben kapott jelszóvisszaállító link segítségével új jelszót állít be. Először ellenőrzi, hogy a két jelszó mezőben megadott jelszó megegyezik-e, majd ha igen, a felhasználó jelszavát hash-eli és frissíti az adatbázisban. Ha sikeres a művelet, a felhasználót a bejelentkezési oldalra irányítja át. Ha nem egyeznek meg a jelszavak, akkor hibaüzenet jelenik meg.

A `userToken` függvény egy felhasználóhoz tartozó tokent használva visszaadja az összes olyan rekordot a felhasználó táblában, amelynek a token értéke megegyezik a megadott `userToken` paraméterrel.

A `loggedinUser` függvény ellenőrzi, hogy a felhasználó be van-e jelentkezve, és ha nincs, akkor igaz értéket ad vissza, egyébként hamisat.

A `userData` függvény ellenőrzi, hogy a felhasználó be van-e jelentkezve, és ha igen, akkor visszaadja az összes olyan rekordot a felhasználó táblában, amelynek az azonosítója megegyezik az aktuális felhasználó azonosítójával és az állapota "1" (aktív). Ha a felhasználó nincs bejelentkezve, akkor nullát ad vissza.

A `userChange` függvény az adatbázisban felhasználó adatainak frissítését végzi el. Az új adatokat POST kéréssel kapja meg, majd ellenőrzi, hogy a megadott telefonszám helyes-e, majd ennek függvényében végrehajtja a frissítést. Végül egy SweetAlert üzenetet jelenít meg a sikeres vagy sikertelen mentésről.

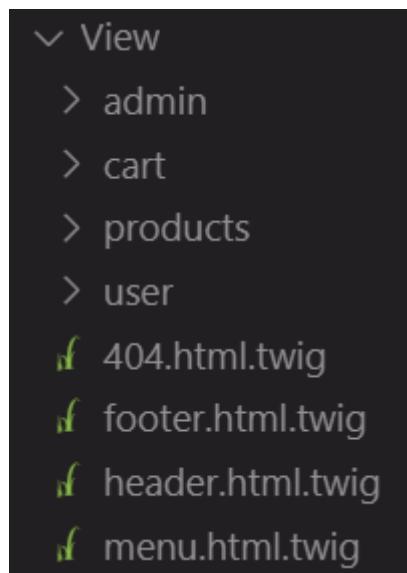
A `userShippingData` függvény a felhasználóhoz tartozó szállítási adatokat kérdezi le az adatbázisból a felhasználó id-ja alapján, majd ezeket az adatokat adja vissza.

A `shippingChange` függvény egy adatbázis kapcsolatot hoz létre, majd POST kérések alapján változtatja vagy beszúrja a szállítási adatokat az adatbázisba. A függvény továbbá ellenőrzi a megadott adatok helyességét, és ha szükséges, hibaüzenetet jelenít meg a felhasználónak. Ha minden adat helyes, akkor sikeres mentésről ad visszajelzést.

```
App > Model > UserDao.php
    ...
533     }
534
535     public static function shippingChange() {
536         $dbObj = new Database();
537         $conn = $dbObj->getConnection();
538
539         $id = $_POST["id"];
540         $szallitasi_cim1 = $_POST["szallitasi_cim1"];
541         $szallitasi_cim2 = $_POST["szallitasi_cim2"];
542         $varos = $_POST["varos"];
543         ($int)$iranyitoszam = $_POST["iranyitoszam"];
544         $orszag = $_POST["orszag"];
545         $telefonszam = $_POST["telefonszam"];
546
547         $sql = "SELECT * FROM felhasznalo_szallitasi_adatok WHERE felhasznalo_id= '$id'";
548         $statement = $conn->prepare($sql);
549         $statement->setFetchMode(PDO::FETCH_OBJ);
550         $statement->execute();
551
552         if ($statement->fetchColumn()) {
553             if (is_numeric($telefonszam) && strlen($telefonszam) == 9 && (mb_substr($telefonszam, 0, 2) == 30 || 
554                 if (is_numeric($iranyitoszam) && strlen($iranyitoszam) == 4) {
555                     $sql = "UPDATE felhasznalo_szallitasi_adatok SET szallitasi_cim1='$szallitasi_cim1', szallitasi_cim2='$szallitasi_cim2', varos='$varos', iranyitoszam=$iranyitoszam, orszag=$orszag WHERE felhasznalo_id= '$id'";
556                     $statement = $conn->prepare($sql);
557                     $statement->execute();
558                     echo "
559                         <script src='
560                             https://cdn.jsdelivr.net/npm/sweetalert2@11.7.2/dist/sweetalert2.all.min.js
561                         '></script>
562                         <script src='/App/js/jquery-3.6.3.min.js'></script>
563                         <script type='text/javascript'>
564                             $(document).ready(function(){
```

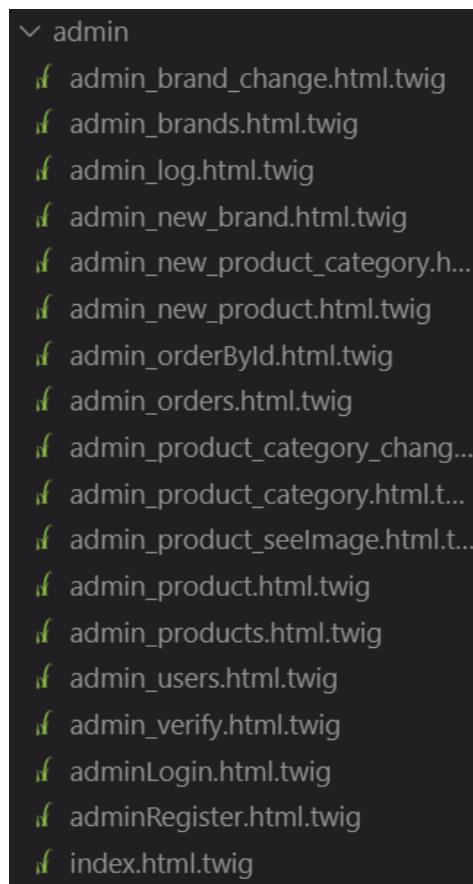
1.5.8 A „View” mappa

A *View* mappa az összes felhasználói felületet (UI) tartalmazza.



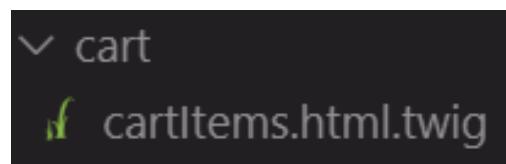
1.5.8.1 Az „admin” mappa

Az *admin* mappa tartalmazza az összes adminisztrációs felületet.



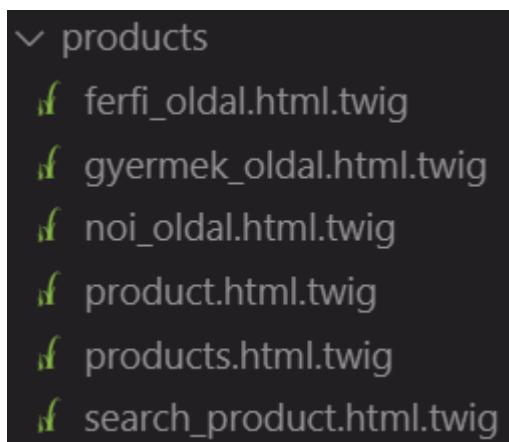
1.5.8.2 A „cart” mappa

A *cart* mappa a kosár oldalakat tartalmazza.



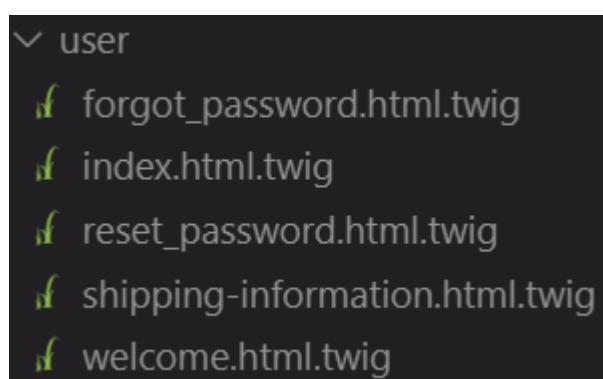
1.5.8.3 A „products” mappa

A *product* mappa a termékoldalakat tartalmazza.



1.5.8.4 A „user” mappa

A *user* mappa pedig az összes felhasználói fiókhoz kapcsolódó oldalt tartalmazza.



1.5.8.5 A „404.html.twig” fájl

A *404.html.twig* egy hibaoldal, amely akkor jelenik meg, ha az alkalmazás egy adott oldalt nem talál.

```
App > View > 404.html.twig
1  ✓ [% include 'header.html.twig' %]
2  ✓     <div class="container mt-5">
3  ✓         <div class="alert alert-danger text-center" role="alert">
4             |     Sajnáljuk, nincs ilyen oldal :(
5             |     </div>
6         </div>
```

1.5.8.6 A „*footer.html.twig*”, és „*header.html.twig*” fájlok

A *footer.html.twig* és *header.html.twig* fájlok a felhasználói felületek alján és tetején látható állandó tartalmakat tartalmazzák.

```
App > View > footer.html.twig
1   <footer class="footer" id="footer">
2     <div class="footer_container">
3       <div class="row">
4         <div class="footer-col">
5           <h4>cégeink</h4>
6           <ul>
7             <li><a href="#">about us</a></li>
8             <li><a href="#">our services</a></li>
9             <li><a href="#">privacy policy</a></li>
10            <li><a href="#">affiliate program</a></li>
11          </ul>
12        </div>
13        <div class="footer-col">
14          <h4>segítség</h4>
15          <ul>
16            <li><a href="#">FAQ</a></li>
17            <li><a href="#">shipping</a></li>
18            <li><a href="#">returns</a></li>
19            <li><a href="#">order status</a></li>
20            <li><a href="#">payment options</a></li>
21          </ul>
22        </div>
23        <div class="footer-col">
24          <h4>online vásárlás</h4>
25          <ul>
26            <li><a href="#">watch</a></li>
27            <li><a href="#">bag</a></li>
28            <li><a href="#">shoes</a></li>
29            <li><a href="#">dress</a></li>
30          </ul>
31        </div>
32        <div class="footer-col">
```

```
App > View > header.html.twig
1   <!DOCTYPE html>
2   <html lang="hu">
3     <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta
7         name="viewport" content="width=device-width, initial-scale=1.0">
8       <!-- Swiper CSS -->
9       <link
10      rel="stylesheet" href="/App/css/swiper-bundle.min.css"/>
11      <!-- CSS -->
12      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-DRoQcVZlNqIwK+ZyL2f4ZJgQHnGjZD9WwZo0kqnpZQvHvJ" crossorigin="anonymous">
13      <link href="https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css" rel="stylesheet"/>
14      <link rel="stylesheet" href="/App/css/index.css"/>
15      <link rel="stylesheet" href="/App/css/slider.css"/>
16      <link rel="stylesheet" href="/App/css/footer.css"/>
17      <link rel="icon" href="/App/images/logoTitle.png">
18      <link href="https://cdn.jsdelivr.net/npm/sweetalert2@11.7.2/dist/sweetalert2.min.css" rel="stylesheet">
19      <link rel="stylesheet" href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">
20      <title>Mester hármas webshop</title>
21    </head>
22    <body>
23      {% set menu = loginHeader %}
24      {% include 'menu.html.twig' with {'login': menu, 'activePage': activePage} %}
25    </body>
26  </html>
27
```

1.5.8.7 A „menu.html.twig” fájl

A *menu.html.twig* fájl pedig a weboldal navigációs menüjét tartalmazza.

```
App > View > menu.html.twig
1   <section class="navbar_section">
2     <nav id="navbar_top" class="navbar navbar-expand-lg bg-light">
3       <div class="container-fluid">
4         <a class="navbar-brand" href="/products"></a>
5         <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedCor
6           aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
7             <span class="navbar-toggler-icon"></span>
8           </button>
9           <div class="collapse navbar-collapse" id="navbarSupportedContent">
10             <ul class="navbar-nav me-auto mb-2 mb-lg-0 mx-auto">
11               {% if activePage == "products" %}
12                 <li class="nav-item">
13                   <a class="nav-link active" aria-current="page" href="/products">Főoldal</a>
14                 </li>
15               {% else %}
16                 <li class="nav-item">
17                   <a class="nav-link" aria-current="page" href="/products">Főoldal</a>
18                 </li>
19               {% endif %}
20
21               {% if activePage == "ferfi" %}
22                 <li class="nav-item dropdown-navbar">
23                   <a class="nav-link active" href="/ferfiOldal" role="button">
24                     Férfi
25                   </a>
26                 </li>
27               {% else %}
28                 <li class="nav-item dropdown-navbar">
29                   <a class="nav-link" href="/ferfiOldal" role="button">
30                     Férfi
31                   </a>
32                 </li>
```

1.6 Tesztelés

Tesztelésre azért van szükség, hogy a szoftver termékben meglévő hibákat még az üzembe helyezés előtt megtaláljuk, ezzel növeljük a termék minőségét, megbízhatóságát. Abban szinte biztosak lehetünk, hogy a szoftverben van hiba, hiszen azt emberek fejlesztik és az emberek hibáznak. Tehát abban szinte biztosak lehetünk, hogy tesztelés előtt van hiba, abban viszont nem lehetünk biztosak, hogy tesztelés után nem marad hiba. A tesztelés után azt tudjuk elmondani, hogy a letesztelt részekben nincs hiba, így nő a program megbízhatósága. Ez azt is mutatja, hogy a program azon funkcióit kell tesztelni, amiket a felhasználók legtöbbször fognak használni.

Néhány tesztelési alapelvek:

- A tesztelés képes felfedni a hibákat, de azt nem, hogy nincs hiba. Ugyanakkor a szoftver minőségét és megbízhatóságát növeli.
- Általában csak a fontosabb részeket érdemes tesztelnünk, hisz nem lehet minden egyes bemeneti kombinációt tesztelni.
- Érdemes a tesztelést az életciklus minél korábbi szakaszában elkezdeni, mert minél hamar találunk meg egy hibát (mondjuk a specifikációban), annál olcsóbb javítani.
- A tesztelésre csak véges időnk van, ezért a tesztelést azokra a modulokra kell koncentrálni, ahol a hibák a legvalószínűbbek, illetve azokra a bemenetekre kell tesztelnünk, amelyre valószínűleg hibás a szoftver
- A hibátlan rendszer téveszméje: Hiába javítjuk ki a hibákat a szoftverben, azzal nem lesz elégedett a megrendelő, ha nem felel meg az igényeinek. Azaz használhatatlan szoftvert nem érdemes tesztelni.

Tesztelési technikák:

- Feketedobozos (black-box) vagy specifikáció alapú, amikor a specifikáció alapján készülnek a tesztelések.
- Fehérdobozos (white-box) vagy strukturális teszt, amikor a forráskód alapján készülnek a tesztelések.

Tesztelési technikák:

- komponensteszt,
- integrációs,
- rendszerteszt,
- átvételi teszt

<https://aries.ektf.hu/~gkusper/SzoftverTeszteles.pdf>

1.6.1 Tesztesetek

Regisztráció esetén kettő visszajelző üzenetet kaphatunk. Az egyik esetben, ha a megadott email címmel létezik már fiók az adatbázisban. A második esetben, ha a két jelszó nem egyezik és úgy rákattint a regisztrációs gombra. Mindkettő esetben egy felugró hiba üzenet figyelmezteti, hogy miért is nem sikerült a regisztráció. A bejelentkezés szint úgy alertes hibát dob, ha a jelszó és email nem egyezik esetleg nincs ilyen email. Ellenkező esetben, ha minden jól töltöttünk ki, akkor alertes üzenetben tájékozódhatunk róla a sikeres regisztrációról vagy bejelentkezésről.

A weboldalon kedvünkre tekinthetünk körbe a különböző termékek közül és mindegyiket megtekinthetjük külön weboldalon, ahol elolvashatjuk a leírásukat, viszont ahhoz, hogy a terméket kosárhoz adjuk bekell jelentkezni fiókunkkal illetve megkell adni a különböző profil adatokat mint pl szállítási adatok. Ellenkező esetben alertes hiba üzenetet kell dobjon az oldal, amiben tájékoztatja a felhasználót a hiba okáról. Ha bevagyunk jelentkezve, akkor tájékoztatnia kell a programnak, hogy sikeresen beletettük a kosárba. Ez két féleképpen történik. Egyszer a jobb kosár ikonnál megkell, hogy jelenjen egy szám, amely azt jelképezi, hogy hány db termék van a kosárban, illetve alertben tájékoztatjuk az egyes termékeknél, hogy sikeresen beletettük a kosárba a kiválasztani kívánt termékét.

A kosár ikonra rákattintva megtekinthetjük kosarunk tartalmát. Láthatjuk mely termékekből, milyen méretűeket illetve hány db -ot tettünk a kosárba. Ha szeretnénk átállíthatjuk, hogy melyik termékből hány db maradjon kosarunkba egy csuszka segítségével. Netán betettünk olyan terméket, amit végül mégse szeretnénk megvásárolni, az esetben törölhetjük a kuka gombra kattintva, aminek következménye, hogy azt a terméket törli kosarunkból illetve levonja az árat az összesített árból. Ha kosarunk üres azt jelzi az oldal is számunkra, illetve üres kosárral nem tudjuk vélegesíteni a rendelést.

Ahhoz, hogy vélegesítsük megrendelésünket, kikell fizetni a terméket. A program leellenőrzi, hogy helyes, adatokat adott meg, ha nem kikell, hogy írja a hibát. Ha sikeres a ez a folyamat is, abban az esetben üzenetet kapunk, a sikeres megrendelésről illetve emailt is.

Admin felületen nyomon követhetjük termékeinket, szűrhetünk rájuk illetve vihetünk fel új termékeket, amelyek megkell, hogy jelenjenek a listázásnál. Nem csak új terméket vihetünk fel. Lehet ez új márka is vagy éppen új termék kategória.

A sikeres rendeléseknek megkell, hogy jelenjenek az admin felületen és azokat admin tudja kezelni különböző módokon. Tudja törölni vagy elküldeni.

Az admin tud felhasználókat is törölni, akiknek emailt küldve értesítjük, hogy fiókját törölték.

Az admin felületen nyomon követhetjük a különböző eseményeket, mint például bejelentkezés, termék módosítás, vagy új termék hozzáadása a webshophoz.

1.7 Továbbfejlesztési lehetőségek

A szoftverünk tervezése során rengeteg tovább fejlesztési lehetőség jutott eszünkbe melyekből csak a számunkra fontosabbakat fogjuk megemlíteni felsorolásképp.

- **Jogosultságok:** A rendszerünkben egyelőre csak egy féle admin jogosultság található, aki kezel, illetve lát minden. Ezt a későbbiekben szeretnénk szétosztani különböző részekre.
- **Szűrés:** Több féle szűrés kezelést tudjon használni a felhasználó, amikor a termékek között keress.
- **Telefon alkalmazás:** Figyeltünk a responzivitásra, de a legtöbb weboldalnak manapság van alkalmazása mobiltelefonra, amit szintén lelehet tölni telefonunkra.
- **Statisztika:** Még több statisztika megjelenítése az admin felületen.
- **Kinézett:** Fejlesztésünk során sok energiát fektettünk a kinézetbe, hiszen ez manapság nagyon fontos, de mindig lehet szebbnél szebb oldalakat csinálni.
- **Szállítás:** A megrendelt termék kiszállítása a vásárlóhoz.
- **Fizetés véglegesítése.**
- **Megrendelt csomag nyomon követése.**
- **Raktár alkalmazás a weboldalhoz,** ahol a termékek raktározását, beszerzését, elküldését nyomon lehet követni.

2 Felhasználói dokumentáció

2.1 Alkalmazás rövid ismertetése

Ez egy webes alapú webshop, mely egy webshopnak megfelelő funkciókat tartalmaz. Gondolok itt a termékek megtekintéséhez, férfi, női, gyermek nemként egyaránt, termék kosárba helyezése, vásárlása, bejelentkezés, kijelentkezés, illetve az oldalhoz tartozik egy admin felület is. Egyenlőre a programunkat kisebb vállalkozóknak szántuk, későbbiekben amint a tovább fejlesztésben leírtakat megvalósítottuk akkor céloznánk meg nagyobb cégeket.

2.2 Rendszerkövetelmények

A programunk Windows 10, IOS illetve Android operációs rendszerrel rendelkező eszközökön futtatható, melyeken internetkapcsolat található. Ahhoz, hogy elérjük lokálisan a forráskódot, akkor a következő követelmények mellett lehetjük meg:

- Apache web szerver 2.4.46 verzió
- PHP 8.0.2 verzió
- MySQL ami 3306-os porton fussion a 'mester_harmas_webshop_db' nevezetű adatbázissal.

A program úgy indítható el, hogy parancssorból bele kell lépnünk a program mappájába és a következő parancssorral indítható el: php -S localhost:8000, amint ezt megtettük elérhető lesz a következő linken: <http://localhost:8000>.

Ajánlott böngészők az alkalmazásunk használatához, melyekkel kiválóan működik:

- Google Chrome
- Firefox

2.3 Alkalmazás telepítése, szükséges beállítások

A webes alkalmazáshoz szükséges kiegészítő a *composer*, amit telepítenünk kell, Ezt a következő linken getcomposer.org található utasításokat követve végezhetjük el a telepítést.

2.4 Alkalmazás használata

A webes alkalmazás használata rendkívül egyszerű, hisz minden egyes gomb magától értetődő. Kizárolag internetkapcsolat mellett működik, mivel egy online adatbázisban ellenőrzi az adatokat a program.

2.4.1 Bejelentkezés/Regisztráció

Ahhoz, hogy egy terméket kosárba tudjunk helyezni, be kell jelentkezni az oldalra. Bejelentkezés nélkül nem lehet rendelést leadni, profil adatokat megadni. Az oldalon csak megtekinteni tudjuk így a termékeket, de profilt bárki tud létrehozni az oldalon a regisztrációra klikkelve, meg kell adnia a teljes nevét, email címét valamint a jelszavát kétszer.

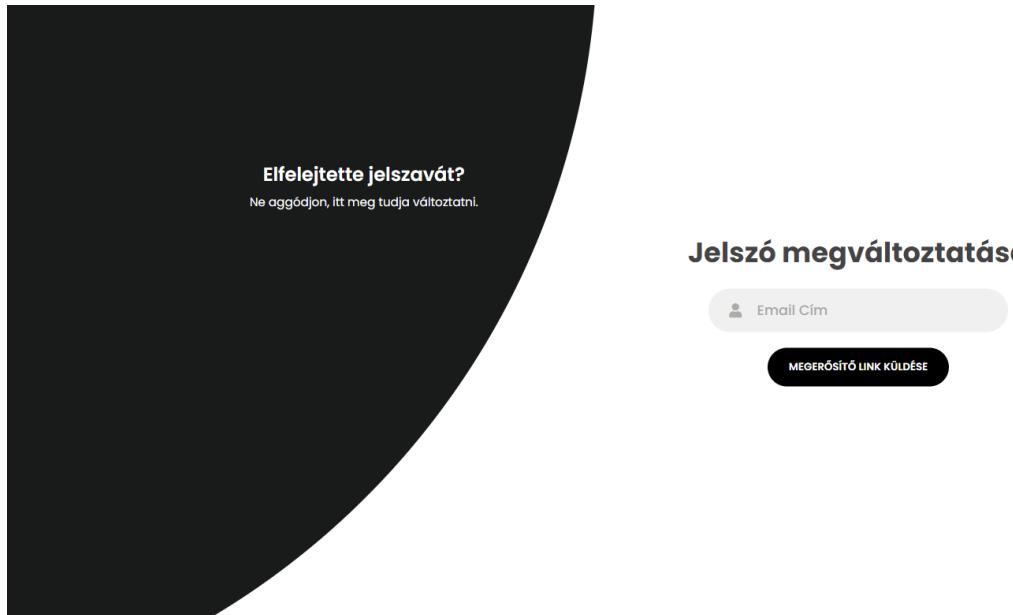
The image shows two overlapping web pages. On the left, a registration form titled 'Regisztráció' is visible. It contains four input fields: 'Teljes Név' (Full Name), 'Email Cím' (Email Address), 'Jelszó' (Password), and 'Jelszó megerősítése' (Password Confirmation). Below these fields is a checkbox labeled 'Elfogadom a feltételeket.' (I accept the terms and conditions) and a 'REGISZTRÁCIÓ' button. At the bottom, there is a link 'Vissza a főoldalra.' (Back to the main page). On the right, a login page titled 'Bejelentkezés' is partially visible. It also has fields for 'Email Cím' and 'Jelszó', a 'BEJELENTKEZÉS' button, and a link 'Vissza a főoldalra.'

Amint ez megtörtént a program küld egy megerősítő emailt a megadott email címre, amit ha megerősítünk, be tudunk jelentkezni az oldalra, amint megadja az email címét és jelszavát.

The image shows two overlapping web pages. On the left, a registration confirmation message is displayed. It says 'Új itt?' (New here?) and 'Az alábbi gombra kattintva tud regisztrálni.' (Clicking the following button you can register). Below this is a 'REGISZTRÁCIÓ' button. On the right, a login confirmation message is shown. It says 'Bejelentkezés' (Login) and 'Elfelejtette jelszavát?' (Forgot password?). Below this is a 'BEJELENTKEZÉS' button and a link 'Vissza a főoldalra.'

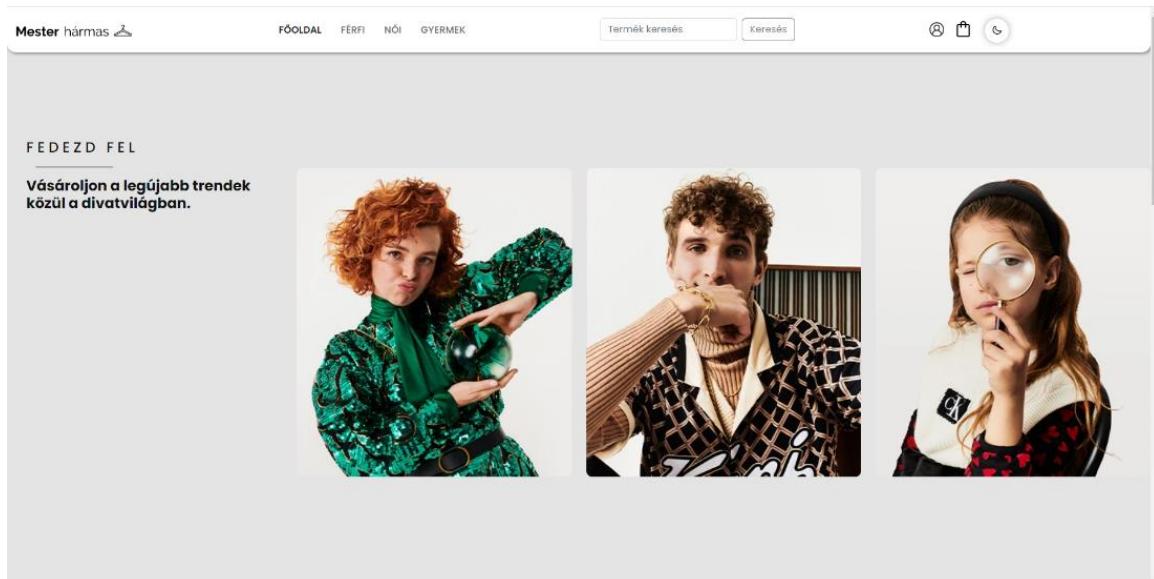
2.4.2 Elfelejtett jelszó

Abban az esetben, ha a felhasználó elfelejtette jelszavát, meg tudja változtatni, csak az email címét kell megadnia, majd az email címén keresztül egy linket, ahol tudja módosítani jelszavát.



2.4.3 Főoldal

Ez az oldal a programunk fő oldala, ahol láthatja a felhasználó az oldalon megtalálható legkiemelkedőbb ruha márkákat, valamint néhány ajánlott terméket ki listáz az oldal a felhasználó számára. Az oldal tetején található egy navbar, melyben a menüpontok között válogathatunk, terméket kereshetünk, a kosarat tudjuk megtekinteni és be tudunk jelentkezni, illetve sötét módra tudjuk rakni az oldalt.



Mester hármas 

FÓOLDAL FÉRFI NŐI GYERMEK

Termék keresés Keresés

Válogass világhírű márkká között



Nike
TOMMY HILFIGER
adidas
Levi's
Calvin Klein
Ralph Lauren

"A MAGABIZTOSÁGOD A DIVATBÓL FAKAD."

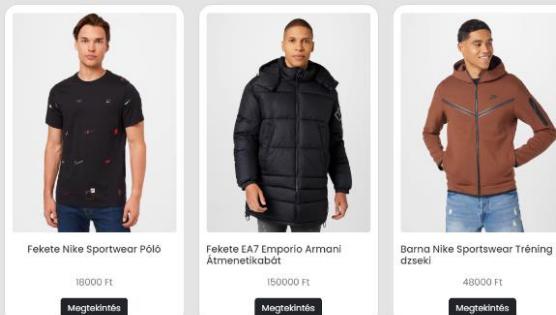
Ajánlott termékek

Mester hármas 

FÓOLDAL FÉRFI NŐI GYERMEK

Termék keresés Keresés

Ajánlott termékek



Fekete Nike Sportswear Póló 18000 Ft Megtekintés

Fekete EA7 Emporio Armani Átmeneti kabát 15000 Ft Megtekintés

Barna Nike Sportswear Tréning dzseki 48000 Ft Megtekintés

2.4.4 Férfi/Női/Gyermek oldalak

Az oldalon található férfi, női, gyermek ruházat. A felhasználó a fent található menüpontoknál tud választani, melyik nem ruházata között szeretne válogatni. A program mindenkorban az elérhető termékeket fogja listázni.

Mester hármas 

FÓOLDAL FÉRFI NŐI GYERMEK

Termék keresés Keresés

Férfi termékek

Osszes termék



Fekete Nike Sportswear Póló 18 000 Ft Megtekintés

Fekete EA7 Emporio Armani Átmeneti kabát 150 000 Ft Megtekintés

Fekete Calvin Klein Ing 38 000 Ft Megtekintés

Barna Nike Sportswear Tréning dzseki 48 000 Ft Megtekintés

Mester hármas

FŐOLDAL FÉRFI NŐI GYERMEK Termék keresés Keresés Összes termék

Női termékek

Szürke Melir GUESS Póló 14 000 Ft Megtékin्�tes	Fekete ICHI szoknya 23 990 Ft Megtékin्�tes	Fekete Nike Sportswear Átmeneti Dzseki 59 000 Ft Megtékin्�tes	Fekete ADIDAS PERFORMANCE Sport top 14 990 Ft Megtékin्�tes
---	---	--	---

Mester hármas

FŐOLDAL FÉRFI NŐI GYERMEK Termék keresés Keresés Összes termék

Gyermekek termékek

Éjkék Champion pulóver 9 592 Ft Megtékin्�tes	Under Armour Szabadidő felső 15 990 Ft Megtékin्�tes	Tommy Hilfiger Póló 7 890 Ft Megtékin्�tes	Nike Sportnadrág 15 990 Ft Megtékin्�tes
---	--	--	--

2.4.4.1 Szűrés ruha kategóriák szerint

Mindegyik oldalon található egy szűrő, amely az aktuális oldalon lévő termékek kategóriájára tud szűrni. Azok a kategóriák láthatók, amelyből található termék az oldalon.

Mester hármas

FŐOLDAL FÉRFI NŐI GYERMEK Termék keresés Keresés

Női termékek

Szürke Melir GUESS Póló 14 000 Ft Megtékin्�tes	Fekete ICHI szoknya 23 990 Ft Megtékin्�tes	Fekete Nike Sportswear Átmeneti Dzseki 59 000 Ft Megtékin्�tes	Fekete ADIDAS PERFORMANCE Sport top 14 990 Ft Megtékin्�tes
---	---	--	---

Miután kiválasztotta a kívánt kategóriát, a weboldal így listázza a felhasználónak a termékeket:

The screenshot shows a website header with navigation links: Mester hármas, FÓOLDAL, NŐI, GYERMEK, Termék keresés, Keresés, and shopping cart icons. Below the header, a search bar contains the text "Pulóverek és melegítőfelsők". The main content area is titled "Női termékek". It displays two product cards. The first card features a woman in a pink ribbed pullover and blue jeans, labeled "Levis Pulóver" with a price of 27 990 Ft and a "Megtekintés" button. The second card features a woman in a beige long-sleeved top and matching pants, labeled "Polo Ralph Lauren Pulóver" with a price of 137 990 Ft and a "Megtekintés" button.

2.4.4.2 Kiválasztott termék oldal

Amint kiválasztottunk egy terméket, a következő oldal fogadja a felhasználót:

The screenshot shows a website header with navigation links: Mester hármas, FÓOLDAL, NŐI, GYERMEK, Termék keresés, Keresés, and shopping cart icons. The main content area displays a product page for an "Adidas Originals Póló". It features two images of a man wearing the white polo shirt. The right image is a close-up of his face. The product title is "Adidas Originals Póló", the price is "11 490 Ft", and there is a dropdown menu for "Válassz méretet" set to "XS". A quantity selector shows "Mennyiség" with value "1" and a "Kosárba tesz" button. Below the button, product details are listed: "Póló, Szín: fehér, Ujjhossz: Negyedes ujj, Hossz: Normál hosszságú, Fazon: normál fazon, Anyag: 100% Pamut". At the bottom of the page, there is a footer with links: Cégünk, Segítség, Online Vásárlás, and Készítette.

Ezen az oldalon, hogyha meg szeretnénk vásárolni a terméket ki tudjuk választani milyen méretben szeretnénk megvásárolni és hány darabot. Amint ez megtörtént a 'Kosárba tesz' gombra kattintva, amennyiben a felhasználó be van jelentkezve, a termék a kosárba kerül.

2.4.5 Termék keresés

Az oldal tetején található egy kereső, ahova be tudja írni a felhasználó, mire szeretne keresni.

Ebben az esetben pólóra keresett a felhasználó:

The screenshot shows the Mester hármas website's search results page for 'póló'. At the top, there is a navigation bar with categories: FŐOLDAL, FÉRFI, NŐI, GYERMEK. Below the navigation bar is a search bar containing the word 'póló'. To the right of the search bar are icons for user profile, shopping cart, and a circular arrow. The main content area is titled 'Termék keresés' (Product search). It displays four product cards:

- Fekete Nike Sportwear Póló** - Price: 18000 Ft - [Megtekintés](#)
- Szürke Melír GUESS Póló** - Price: 14000 Ft - [Megtekintés](#)
- Kék Polo Ralph Lauren Póló** - Price: 70990 Ft - [Megtekintés](#)
- Kék, Világoskék Polo Ralph Lauren Ing** - Price: 50990 Ft - [Megtekintés](#)

2.4.6 Sötét mód

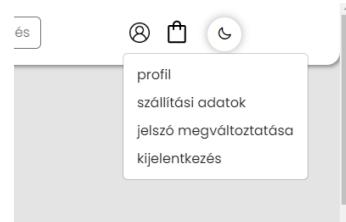
Az oldalon a jobb felső sarokban található egy hold ikon, amire ha kattint a felhasználó át vált az oldal egy sötétebb kinézetre.

The screenshot shows the Mester hármas website in dark mode. The header and navigation bar are visible at the top. A circular icon with a crescent moon symbol is located in the top right corner. The main content area features a dark banner with the text 'FEDEZD FEL' and 'Vásároljon a legújabb trendek közül a divatvilágban.' Below this, there are three product images: a woman in a green sequined jacket holding a green sphere, a man in a patterned blazer, and a girl looking through a magnifying glass.

2.4.7 Profil adatok

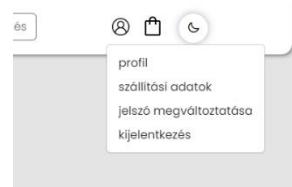
A felhasználó itt tudja a profil adatait megadni, amint a profil gombra klikkel.

Az email címen nem lehet változtatni, amivel regisztrált a felhasználó azt fogja látni. Ezek az adatokat meg kell adnunk mindenkiéppen, hogyha rendelést szeretnénk leadni. Változtatni tudunk felhasználónéven, telefonszámon és a felhasználó nevén bármikor.



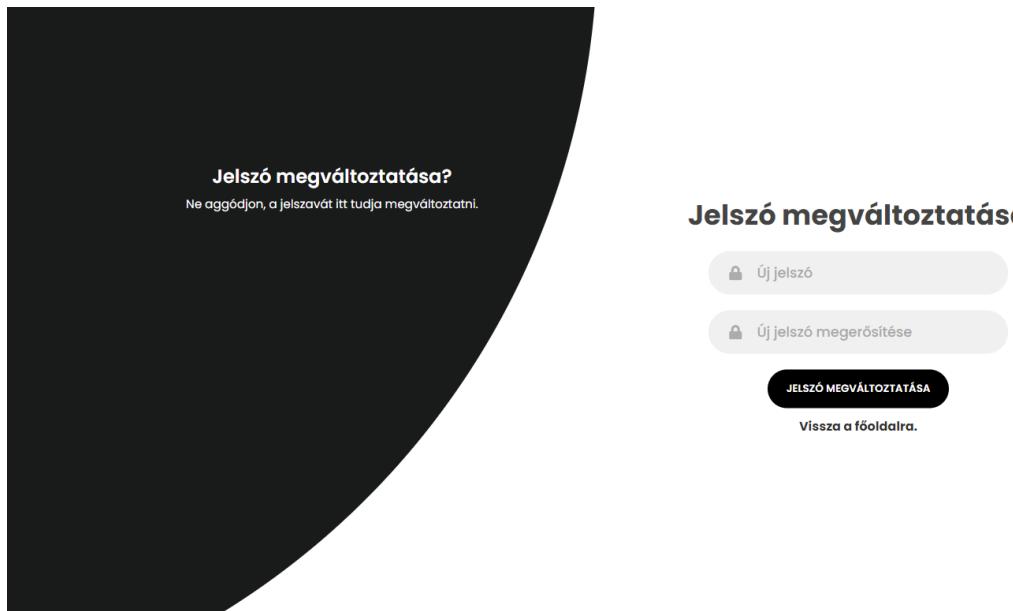
2.4.8 Szállítási adatok

A szállítási adatokat kiválasztva a felhasználó meg tudja adni hova szeretné, hogy a megrendelt terméke érkezzen. Az adatok megadása nélkül nem tud rendelni a felhasználó.



2.4.9 Jelszó megváltoztatása

Amennyiben a felhasználó meg szeretné változtatni jelszavát, a jelszó megváltoztatása gombra kattintva megteheti.



2.4.10 Kosár

Abban az esetben, ha a felhasználó kiválasztotta a megvásárolni kívánt termékeket, a kosár ikonra kattintva tudja megtekinteni kosarát. Itt tudja módosítani még a termék mennyiségét a slider segítségével, valamint el is tud távolítani terméket. Ha a felhasználó fizetni szeretne, akkor a 'Menj a fizetéshez' gombra kattintva megteheti.

The screenshot shows a shopping cart page with two items:

- Kosár**:
 - Calvin Klein**: Fekete Calvin Klein Ing. 35 990 Ft/db. Méret: XL. Mennyiség: 1 db. A slider is set to 1 db. There is a delete icon (trash bin) next to the quantity input field.
 - Polo Ralph Lauren**: Kék Polo Ralph Lauren Póló. 70 990 Ft/db. Méret: M. Mennyiség: 2 db. A slider is set to 2 db. There is a delete icon (trash bin) next to the quantity input field.
- Összegzés**: A termék összege 177 970 Ft. A 'Menj a fizetéshez' button is present.

2.4.11 Fizetés véglegesítés

A felhasználó itt tudja a fizetést végrehajtani amint megadta email címét és a kártya adatait. Bal oldalon láthatja milyen terméket, hány darabot és hogy mennyit kell fizetnie.

The screenshot shows a payment page from stripe.com. On the left, there's a summary of items:

Fekete Calvin Klein Ing	141 980,00 Ft
Mennyiség: 2	70 990,00 Ft darabjáért
Kék Polo Ralph Lauren Póló	35 990,00 Ft
Mennyiség: 1	

The total amount is **177 970,00 Ft**. The right side of the screen shows a card input form with fields for E-mail-cím, Kártyaadatok (with sample card number 1234 1234 1234 1234), A kártyán szereplő név, Ország vagy régió (Magyarország), and a checkbox for Adataim biztonságos mentése az egykattintásos fizetéshez. Below the card input is a blue button labeled **Fizetés**.

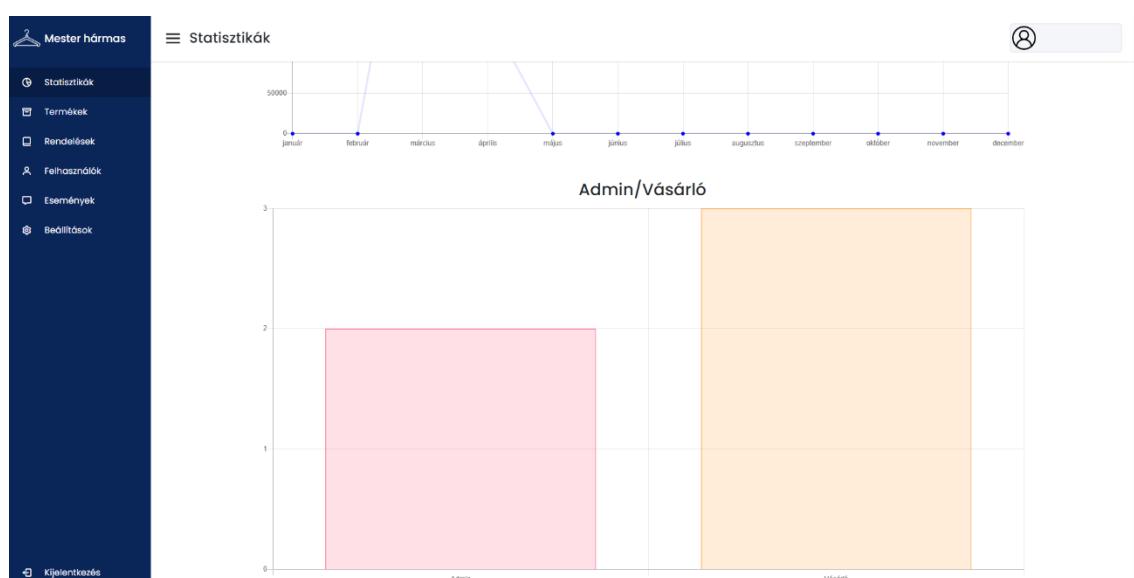
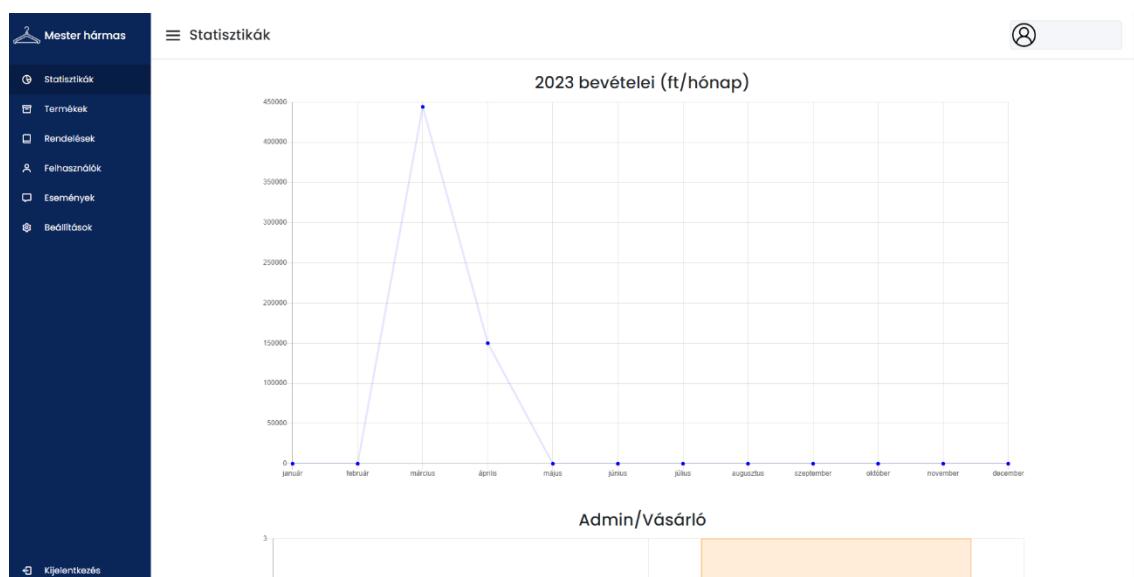
2.4.12 Admin felület

2.4.12.1 Bejelentkezés/Regisztráció

Az admin felület használatához be kell jelentkezni. Amennyiben egy új admin fiókot szeretnénk létrehozni, regisztrálnunk kell. Ha a regisztráció sikeres volt, küld az adminoknak egy admin kérelmet emailben. Az emailben található linken az admin el tudja dönteni, hogy elfogadja, vagy elutasítja a kérelmet. Mindkét esetben tájékoztatva lesz emailben a regisztrálni próbált felhasználó.

2.4.12.2 Statisztikák

Ha sikeresen bejelentkeztünk, a statisztikák oldal fogad minket, ahol láthatjuk, hogy melyik hónapban mennyi volt a bevétel és hogy hány admin és hány regisztrált vásárló van az oldalon.



2.4.12.3 Termékek

A termékek oldalon találjuk az oldalon lévő összes terméket, itt tud az admin módosítani, törlni, vagy felvinni új terméket.

ID	Név	Cikkszám	Márkanév	Nem	Kategória	Ár	Módosítás	Töröl
4	Fekete Nike Sportwear Póló	01000123	Nike	Férfi	Pólók	18000	Módosítás	Töröl
5	Fekete EA7 Emporio Armani Átmeneti kabát	00123467	EA7 Emporio Armani	Férfi	Kabátok	150000	Módosítás	Töröl
6	Fekete Calvin Klein Ing	01237864	Calvin Klein	Férfi	Ingek	35990	Módosítás	Töröl
8	Szürke Melir GUESS Póló	01111562	Guess	Női	Pólók	14000	Módosítás	Töröl
11	Barna Nike Sportswear Tréning dzseki	1984001	Nike	Férfi	Tréningruhák	48000	Módosítás	Töröl
12	Kék Polo Ralph Lauren Póló	003453215	Polo Ralph Lauren	Férfi	Pólók	70990	Módosítás	Töröl
13	Fekete Nike Sportswear Nadrág	3214210	Nike	Férfi	Tréningruhák	13990	Módosítás	Töröl
14	Kék Világoskék Polo Ralph Lauren Ing	0178889	Polo Ralph Lauren	Férfi	Ingek	50990	Módosítás	Töröl
15	Fekete ICHI szoknya	1000653	ICHİ	Női	Szoknyák	23990	Módosítás	Töröl
17	Fekete Nike Sportswear Átmeneti Dzseki	NIS3956001000001	Nike	Női	Kabátok	59000	Módosítás	Töröl
18	Fekete ADIDAS PERFORMANCE Sport top	AD19404002000001	Adidas	Női	Tréningruhák	14990	Módosítás	Töröl
19	Sötét-Rózsaszín TOMMY HILFINGER Blézer	THS9bo0010000001	Tommy Hilfiger	Női	Blézerek	119000	Módosítás	Töröl
20	Ejkék Champion pulóver	CHP3948001000001	Champion	Gyermek	Pulóverek és melegítőfelsök	9592	Módosítás	Töröl

Termék márkká gombra kattintva tudja az admin megnézni milyen márkkák vannak a rendszerben, itt tud új márkkát hozzáadni, törlni, módosítani.

ID	Név	Módosítás	Töröl
1	Nike	Módosítás	Töröl
2	Adidas	Módosítás	Töröl
3	Calvin Klein	Módosítás	Töröl
4	Guess	Módosítás	Töröl
5	Tommy Hilfiger	Módosítás	Töröl
6	EA7 Emporio Armani	Módosítás	Töröl
7	Hunkemöller	Módosítás	Töröl
8	Polo Ralph Lauren	Módosítás	Töröl
9	ICHİ	Módosítás	Töröl
10	CK	Módosítás	Töröl
13	Champion	Módosítás	Töröl
14	Levis	Módosítás	Töröl
15	G-STAR RAW	Módosítás	Töröl
16	Under Armour	Módosítás	Töröl

A termék kategóriák gombra kattintva pedig az összes kategóriát tudja az admin megtekinteni, módosítani, újat felvinni és törölni is.

Id	Név	Módosítás	Töröl
2	Pólók	Módosítás	Töröl
3	Pulóverek és melegítőfelsők	Módosítás	Töröl
4	Rövidnadrágok	Módosítás	Töröl
5	Tréningruhák	Módosítás	Töröl
6	Mezék	Módosítás	Töröl
7	Kabátok	Módosítás	Töröl
8	Farmerek és nadrágok	Módosítás	Töröl
9	Leggingsék	Módosítás	Töröl
10	Fehérneműk	Módosítás	Töröl
12	Ingek	Módosítás	Töröl
13	Alsóneműk	Módosítás	Töröl
14	Szoknyák	Módosítás	Töröl
15	Blézerek	Módosítás	Töröl

2.4.12.4 Rendelések

A rendelések oldalon találhatjuk az összes leadott rendelést. Látható, melyik rendelés van folyamatban és melyik elküldve.

Termék név	Cikkszám	Méret	Mennyiség	Ár	Rendelés dátuma	Státusz	Elküldés	Részletek	Töröl
Fekete Calvin Klein Ing	01237864	M	1	35990	2023-04-03	Folyamatban	Elküld	Részletek	Töröl
Kék, Világoskék Polo Ralph Lauren Ing	0178889	M	1	50990	2023-04-03	Folyamatban	Elküld	Részletek	Töröl
Levis Pulóver	DF4433H	S	2	55980	2023-04-03	Folyamatban	Elküld	Részletek	Töröl
Kék Polo Ralph Lauren Póló	003453215	L	1	70990	2023-04-03	Folyamatban	Elküld	Részletek	Töröl

Amikor a rendelést elküldte az admin, a vásárló kap erről egy emailt, és az admin felületen is elküldve státusz lesz látható.

Termék név	Cikkszám	Méret	Mennyiség	Ár	Rendelés dátuma	Státusz	Elküldés	Részletek	Töröl
Fekete Calvin Klein Ing	01237864	M	1	35990	2023-04-03	Elküldve	<button>Elküld</button>	<button>Részletek</button>	<button>Töröl</button>
Kék, Világoskék Polo Ralph Lauren Ing	0178899	M	1	50990	2023-04-03	Elküldve	<button>Elküld</button>	<button>Részletek</button>	<button>Töröl</button>
Levis Pulóver	DF443H	S	2	55980	2023-04-03	Elküldve	<button>Elküld</button>	<button>Részletek</button>	<button>Töröl</button>
Kék Polo Ralph Lauren Póló	003453215	L	1	70990	2023-04-03	Elküldve	<button>Elküld</button>	<button>Részletek</button>	<button>Töröl</button>

A részleteknél pedig meg tudja nézni az admin a rendelő adatait.

Felhasználónév	Vezetéknév	Keresztnév	Szállítási cím	Település	Irányítószám	Ország	Telefonszám
webshop	Szászovszki	Kevin	Petőfi utca 1	Vác	2600	Magyarország	06702759888

2.4.12.5 Admin felület felhasználók oldal

A felhasználók nevű fülönél nyomon követhetjük azokat a felhasználókat, akik rendelkeznek fiókkal. Láthatjuk különböző adataikat és jogosultságukat. Szűrhetünk a jogosultságokra, amely lehet admin vagy vásárló. A felhasználókat tudjuk törölni is, kivéve az alapértelmezett admint, aki a webshop nevű felhasználó.

The screenshot shows the 'Felhasználók' (Users) section of the admin interface. On the left is a sidebar with navigation links: Statistikák, Termékek, Rendelések, Felhasználók (selected), Események, and Beállítások. The main area has a search bar and a table with columns: Felhasználónév, Vezetéknév, Keresztnév, Telefonszám, Jogosultság, Email, and Töröl (Delete). The table contains five rows of user data:

Felhasználónév	Vezetéknév	Keresztnév	Telefonszám	Jogosultság	Email	Töröl
webshop	Szozovszki	Kevin	06702759888	Admin	mester.harmas.webshop@gmail.com	Töröl
Fenyvesi Ferenc	Ferenc	Ferenc	06304404798	Vásárló	fenyvesi.ferenc02@gmail.com	Töröl
Belők Emese				Vásárló	mse.belok@gmail.com	Töröl
RG	Regűs	Gábor	06307582195	Admin	regugabor2001@gmail.com	Töröl
SzK	Szozovszki	Kevin	06702746764	Vásárló	szozovszkikevin@gmail.com	Töröl

2.4.12.6 Admin felület események oldal:

Az események nevű fülönél nyomon követhetjük azokat az eseményeket, hogy mely felhasználó mikor milyen eseményt követett el. Az eseményekre tudunk szűrni, mint például, hogy bejelentkezés, termék módosítás.

The screenshot shows the 'Események' (Events) section of the admin interface. On the left is a sidebar with navigation links: Statistikák, Termékek, Rendelések, Felhasználók, Események (selected), and Beállítások. The main area has a search bar and a table with columns: Esemény, Azonosító, Időpont, and Töröl (Delete). The table contains a list of events:

Esemény	Azonosító	Időpont	Töröl
bejelentkezés	-	2023-04-03 12:48:55	Töröl
rendelés küldése	webshop	2023-04-03 12:47:30	Töröl
bejelentkezés	-	2023-04-03 12:44:41	Töröl
kijelentkezés	-	2023-04-03 12:44:33	Töröl
bejelentkezés	-	2023-04-03 12:42:07	Töröl
bejelentkezés	-	2023-04-02 10:19:13	Töröl
bejelentkezés	-	2023-04-02 10:00:48	Töröl
bejelentkezés	-	2023-04-02 10:00:37	Töröl
Fenyvesi Ferenc	bejelentkezés	2023-04-02 10:00:21	Töröl
Fenyvesi Ferenc	kijelentkezés	2023-04-02 09:59:17	Töröl
Fenyvesi Ferenc	bejelentkezés	2023-04-02 09:54:39	Töröl
Fenyvesi Ferenc	bejelentkezés	2023-03-31 22:00:20	Töröl
webshop	bejelentkezés	2023-03-31 13:48:51	Töröl
webshop	bejelentkezés	2023-03-31 13:45:40	Töröl

3 Irodalomjegyzék

3.1 Online tartalmak

Bootstrap: <https://getbootstrap.com/>

Stackoverflow: <https://stackoverflow.com/>

Sdemy: <https://www.udemy.com/>

W3schools: <https://www.w3schools.com/>

PHP: <https://www.php.net/>

Twig: <https://twig.symfony.com/>

JavaScript: <https://www.javascript.com/>

jQuery: <https://jquery.com/>

Composer: <https://getcomposer.org/>

MySQL: <https://www.mysql.com/>

3.2 Könyvek

Robert C. Martin: Tiszta kód

Steve McConnell: Code Complete (second edition)

William Sanders: Learning PHP Design Patterns

Junade Ali: Mastering PHP Design Patterns

Peter Moulding: PHP Black Book

Steve McConnell: Code Complete

4 Mellékletek

4.1 Webes alkalmazás

Webes alkalmazás elérése:

<https://mesterharmaswebshop.hu/>

Admin felület elérése:

<https://mesterharmaswebshop.hu/admin>

Github elérés:

https://github.com/regosgabor2001/mester_harmas_webshop