

BKSZC

Weiss Manfréd Technikum,
Szakképző Iskola és Kollégium



RIMKING.HU

Témavezető:

Kovács László

Készítette:

Pulszter Csaba

Budapest

2024

Tartalomjegyzék

1.Bevezetés	2
– 1.1. Köszönetnyilvánítás –	2
– 1.2. Témaválasztás indoklása –	2
– 1.3. Elméleti és gyakorlati lehetőség –	2
2.Fejlesztői dokumentáció	3
– 2.1. Az oldal témája –	3
– 2.2. Dizájn és elképzelés –	3
– 2.3. Fejlesztői környezet –	4
– 2.4. A weblapon használt adatszerkezet –	5
– 2.6. A user tábla –	6
– 2.7. A login táblája –	7
– 2.8. A termék tábla –	9
– 2.9. A kezdőlap –	10
– 2.9.1. A kezdőlapi kártya elkészítése –	13
– 2.10. Termék bázis –	16
– 2.10.1. Termék info léc –	17
– 2.10.2. Remove, clear all –	18
– 2.11. Hibaüzenet az oldalon –	20
3.Felhasználói dokumentáció	21
– 3.1 Hardware és szoftver igény –	21
– 3.2. A program készítőjének elérhetősége –	21
3.3. Használat és tájékozódás	21
4.Összefoglalás.....	24
– 4.1. A szakdolgozat célja –	24
– 4.2. Célkitűzések, amik nem sikerültek teljességgel –	24
5.Irodalomjegyzék	25
– 5.1. A szakdolgozathoz használt források –	25

1.Bevezetés

– 1.1. Köszönetnyilvánítás –

Ezúton szeretném megköszönni Kovács László Bálint tanár úrnak, Ervin Bruno Englert tanár úrnak az évek során nyújtott segítséget akik nélkül ez a szakdolgozat nem jöhetett volna létre. Hálás köszönet minden biztatásért és tudásért amit átadtak.

– 1.2. Témaválasztás indoklása –

Ahogy a diákok között állva, a webfejlesztés és a komplex programozás közötti különbséget érzem, bár mindkettő izgalmas terület, valahogy a webfejlesztés közelebb áll hozzám. Az interaktív weboldalak és az online alkalmazások létrehozása valósággal lenyűgöz. A weboldal létrejöttét több internetes forrás is segítette, köztük videós források is, sokat tanultam különböző YouTuberek videóiból. Az autók iránti szeretetem azonban valóban meghatározta témaválasztásomat. Ahogy elgondolkodom a kihívásokról és az új lehetőségekről, úgy érzem, hogy ez a pálya mindent tartogathat, amire vágyom.

– 1.3. Elméleti és gyakorlati lehetőség –

Ezen web programozáson belül megtudhatjuk miként lehet egy termék bázis felületet létrehozni és hogyan formázzunk meg és készítsünk el egy letisztult igényes kezdőlapot és még több hasznos dolgot. Megtudhatjuk miként működik egy ilyen fajta oldal adatbázisa és felépítése. Betekintést nyerünk az oldalon belül tartózkodó gombok, menü felületek, és dizájn / formázási rejtélyeibe és programon belüli megvalósításába. Többek között megismerjük a program mögötti elgondolást, tervet és elképzelést témaválasztás és munkamenet szempontjából is. Végülis tökéletesen tovább fejleszthető, formálható jellegzetességekkel rendelkezik így átlátható sablon funkciókat tartalmaz amivel további gyakorlati ismereteket lehet elsajátítani.

2.Fejlesztői dokumentáció

– 2.1. Az oldal témája –

Az oldal témája 100%-ig az autókra összpontosít és azon belül is főleg a felnikre ezért lett az oldal neve **RIM** King aminek angol megfelelője rim = felni és a **KING** = király ami az oldal dominanciáját tükrözi , hogy kitűnjön a piacról és egy bizonyos előnyből induljon. A fejleszthetősége miatt az oldal kompatibilis a termék bázis további készlet bővítésére alkatrész szempontból például: fényszórók, hátsó lámpák, optikai alkatrészek, motor tuning alkatrészek.

– 2.2. Dizájn és elképzelés –

Az oldal főképp a letisztultságon alapul, hogy átlátható legyen, könnyen kezelhető és mindenkinek felett praktikus elosztással rendelkezik. A felület egy stíluson alapszik betű típusait, felépítését és környezetét tekintve. Így gondolkodtam miközben próbáltam hangsúlyt helyezni az oldal dinamikusságán kezelhetőség szempontjából. A dizájn alapja ellentétes a témával amit képvisel ugyanis az autók mint járművek összetettek és bonyolultak aki ebben a világban jártas az tudhatja így törekvéseim ezért haladtak a könnyen kezelhetőség és átláthatóság irányába, úgy gondolom, hogy ez lehet a kulcs egy jó autós téma felépítésében. Jövőre tekintően is próbálkoztam gondolkodni, hogy miként lehet ezt a bonyolult rendszert bővíteni, szépíteni és persze gyarapítani eme témával szemben. A lehetőségek tömkelege jutott eszembe de a végeredmény magáért beszél.

– 2.3. Fejlesztői környezet –

A weboldal szerkesztéséhez és elkészítéséhez több programot és eszközt használtam segítségül amik nagyban megkönnyítették vagy hozzájárultak a munkámhoz. Ehhez egész pontosan **5** dolgot hívtam segítségül.

- 1.** Szükség volt **PHPMYADMIN**-ra egy lokális környezet kialakításához és adattároláshoz. Ezt persze egy úgynevezett **localhost**-on belül tudtam dinamikusan kezelni, nyomon követni és összekapcsolni a kódommal.
- 2.** A második program amit használtam a kód megírásához és amin igazából a nagy része alapszik a záródolgozatomnak nem más mint **Visual Studio Code** néven ismert. Itt rejlik a sok kódolás **php**-ban, **html**-ben, **css**-ben és persze **adatbázis**-ban. Igazából ő a fejlesztői környezet főszerep játszója.
- 3.** Harmadik helyre sorolom a **jegyzetömböt** ami arra szükségeltetett, hogy a kód írása és szerkesztése közben eltároljak begépelte formában szükséges, nem szükséges esetleg alternatív megoldással rendelkező kód részleteket. Ezt természetesen a leadott záródolgozat **nem tartalmazza**, inkább egy létfontosságú háttér segítségnek volt hasznomra.
- 4.** Negyediknek szükségem volt egy **internetes közeg**-re ahol letudtam futtatni a webes felületet és ezáltal tudtan ellenőrizni a haladásom folyamattát, kinézetét, és esetleges hibáit.
- 5.** Végül utolsó helyen az **internetes források** amik nagyban segítettek munkámat többek között írásos és videós formában is. Ezen források megtalálhatóak az **Irodalomjegyzék 5-ös pontjában** a dokumentáció végén.

– 2.4. A weblapon használt adatszerkezet –

A weblapon használt adatszerkezetéért felelős amár említett **PHPMYADMIN**. A számítástechnikában az adatbázis elektronikusan tárolt és elérhető adatok szervezett gyűjteménye. Ezen belül a regisztrációnál **integer** és **string** típusú adatokat tárolunk. A weboldal fő adattárolásáért **integer**, **varchar**, **text**, **date** illetve **datetime** adattípusok felelnek, ezután a **login = bejelentkezés** a már eltárolt felhasználó adatainak mentése után külön táblába tárol a következő adattípusok alapján. Az adattípusok a következő sorrendben haladnak **lid**, **luid**, **ldatum**, **lip**. A másik fő tábla az úgynevezett **termek** tábla ami a termékek adatbázisát biztosítja.

Ezen belül főképpen a termékekhez hozzátartozó képeket is biztosítja amit egy **id**, **name = név**(termék neve), **price = ár** és **image = kép**(termék képe) adattípusokkal hoztam össze.

– 2.5. Az adatbázis különböző táblái és azon mezői –



The screenshot shows the phpMyAdmin interface for a database named 'pelda'. The left sidebar shows the database structure with 'pelda' selected. The main area displays a table list for 'pelda' with columns: Tábla, Művelet, Sorok, Típus, Illesztés, Méret, and Felülírás. The tables listed are login, naplo, termek, and user.

Tábla	Művelet	Sorok	Típus	Illesztés	Méret	Felülírás
login	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	0	InnoDB	utf8mb4_0900_ai_ci	16 KB	-
naplo	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	950	InnoDB	utf8mb4_0900_ai_ci	80 KB	-
termek	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	20	InnoDB	utf8mb4_0900_ai_ci	16 KB	-
user	Tartalom Szerkezet Keresés Beszúrás Kiürítés Eldobás	0	InnoDB	utf8mb4_0900_ai_ci	16 KB	-
4 tábla	Összesen	970	InnoDB	utf8mb4_0900_ai_ci	128 KB	0 B

Ez a **peldaként** elnevezett adatbázis szolgáltatja, működteti és tárolja az egész weblap adatainak összességét. Ezen belül több tábla is található amik tartalmazzák a különböző adat mezőket is. Az adatbázis végig fut a **Visual Studio Code**-on ami tulajdonképp feldolgozza, részletezi és megjeleníti végül az internetes felületen az itt tárolt tartalmat.

– 2.6. A user tábla –

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra	Művelet
1	uid	int(11)			Nem	Nincs		AUTO_INCREMENT	Módosítás Eldobás Több
2	unev	varchar(20)	utf8mb4_0900_ai_ci		Nem	Nincs			Módosítás Eldobás Több
3	umail	varchar(60)	utf8mb4_0900_ai_ci		Nem	Nincs			Módosítás Eldobás Több
4	upw	varchar(32)	utf8mb4_0900_ai_ci		Nem	Nincs			Módosítás Eldobás Több
5	udatum	datetime			Nem	Nincs			Módosítás Eldobás Több
6	uip	varchar(32)	utf8mb4_0900_ai_ci		Nem	Nincs			Módosítás Eldobás Több
7	ustatusz	varchar(1)	utf8mb4_0900_ai_ci		Nem	Nincs			Módosítás Eldobás Több

Művelet	Kulcsnév	Típus	Egyedi	Csomagolt	Oszlop	Számosság	Illesztés	Nulla	Megjegyzés
Módosítás Eldobás	PRIMARY	BTREE	Igen	Nem	uid	0	A	Nem	

A **USER** tábla a felhasználók adatainak tárolására szolgál. Az első adatmező nem más mint a **uid** ami a felhasználó **id** számát tárolja **integer**-ben és összesen **11** karakter lehet. Mivel ez a mező a legfontosabb és legmeghatározóbb és kapcsolatban más táblák esetleges mezőivel ezért **elsődleges kulcsként** (angolul **primary keyként**) van megadva. Természetesen megvan adva az úgynevezett

AUTO_INCREMENT ami lehetővé teszi, hogy egy egyedi számot generáljunk egy új rekord hozzáadásakor. Az egyediséget úgy generálja, hogy nem mi töltjük ki az értékét és mindig az előző rekord értékénél eggyel nagyobbat generál. Ezután következik az **UNEV** nevű mező ami a felhasználó nevét tárolja el amit saját maga ad meg. Ez a mező **varchar**-ban tárolja el a neveket ami majdnem úgy működik mint egy **string** típusú adattípus, rövidítés a "**variable character**" kifejezésből származik, és lehetővé teszi a karakterláncok tárolását változó hosszúságban. A **unev**-nél **20** karakter hosszú nevet tudunk megadni. Az **umail** mező szintén a felhasználó által kell legyen megadva, ez a mező szolgáltatja az **email** cím adatának tárolását. Az **umail** szintén **varchar**-ban van megadva annyi különbséggel, hogy itt **60**

karakter hosszú email címet tudunk megadni. Az adatbázis táblájának közepén található az **upw** nevű mező ami a **user password** miatt kapta a nevét tehát ez a tábla **jelszavakat** fog eltárolni ami szintén és konstansan felhasználó által lesz megadva. Adattípusa **varchar** és **32** karakter hosszú lehet maximális esetben. **udatum** egyik legegyszerűbb mezők egyike ami a felhasználó **bejelentkezési dátumát** tárolja és figyeli. Ennek adattípusa már nem **varchar** hanem **datetime**-ban van megadva tehát nem csak dátumot tárol hanem pontos időpontot is az esetleges

bejelentkezés/regisztrációról. Itt karakter megadásra **nincs/nem volt** szükség. **uip**, rövidítése **user ip (ip cím)**. Ezen mező tárolja a felhasználó **ip címét**, ami egy kifejezetten fontos és elengedhetetlen része egy regisztrációs felületnek. Adattípusa **varchar** és mint a jelszónál **32** karakterben van megadva. Végül az **ustatusz** ami a **user** tábla legegyszerűbben kifejezhető mezője, **user státusz** ami a felhasználó **státuszát** adja meg **varchar**-ban és mindössze **1** darab karakterből áll. Ez lenne hát adatbázisom első táblája, ugorjunk egyből a következőre.

– 2.7. A login táblája –

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra	Művelet
1	lid	int(11)			Nem	Nincs		AUTO_INCREMENT	Módosítás Eldobás Több
2	luid	int(11)			Nem	Nincs			Módosítás Eldobás Több
3	ldatum	datetime			Nem	Nincs			Módosítás Eldobás Több
4	lip	varchar(32)	utf8mb4_0900_ai_ci		Nem	Nincs			Módosítás Eldobás Több

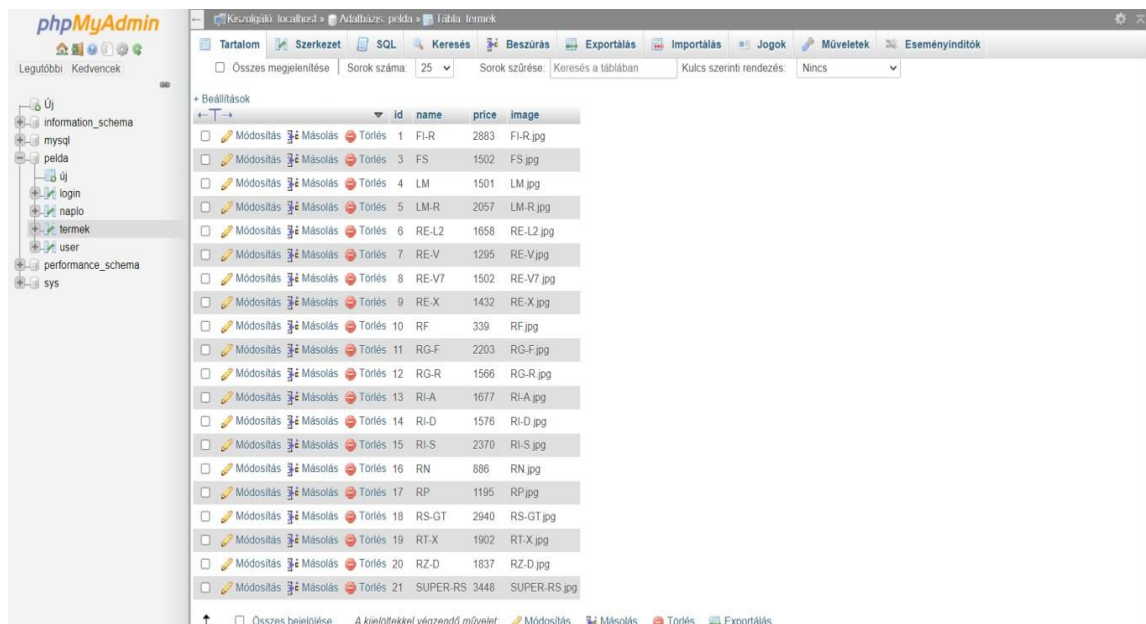
A **login** tábla felelősségteljes munkája nagy szerepet játszik a bejelentkezéshez szükséges adatok tárolásában miután megtörtént a regisztráció. Ezen tábla **négy** darab különböző mezőt tartalmaz **három** különböző adatípussal. Menjünk is végig rajta egyesével úgy gondolom. Az első amit megfigyelhettünk már a dokumentáció során, hogy minden mező neve **le** van rövidítve egy **kulcsszóra** ami tartalmazza a mező **funkciójának lényegét**. Ebben az esetben az **L** az az **login** névre fog hallgatni minden mező nevének legelején. Legelső táblánk az **lid** névre hallgat

ami röviden annyit tesz egyértelműen, hogy **login id**, tehát itt kapjuk meg a bejelentkezéskor szükséges **felhasználói id**-t ami megkülönbözteti a felhasználókat. **lid** egy **integer** adattípusban van megadva. Amit fontos megjegyezni ha eddig nem említettem volna, hogy **az integer egy olyan adattípus az**

adatbázisrendszerekben, amely egész számokat tárol. Általában az "integer" típus fix méretű egész számok tárolására szolgál, amelyek általában 32 vagy 64 bit hosszúságúak. Az "integer" típusú oszlopokban csak egész számokat lehet tárolni, például -2, -1, 0, 1, 2 stb. Ez esetben az lid maximális karakterszáma a 11 és elsődleges kulcsként (primary key-ként) van megadva amiről már hallottunk ebben a dokumentációban. Emellett AUTO_INCREMENT van beállítva hogy a felhasználó automatikusan más id számot kapjon mint a többi felhasználó, így ezáltal lehet a legjobban megkülönböztetni őket. A második mezőnk az luid (login user id = login felhasználói id) ami az állandó és sajátos id száma lesz a felhasználónak. Természetesen ez is integer adattípusban van megadva és ugyanúgy 11 karakter hosszú lehet. Harmadik helyen az ldatum a bejelentkezéskor használt dátum és idő, hogy pontosan megállapítsuk, mikor is volt bejelentkezés a felhasználó által. Rövidítése a login dátum, date = dátum.

Adattípus szempontjából nem mást adtam meg mint a **datetime**-ot és **nincs/nem volt szükséges** ilyenkor megadni **karakter hosszúságot**. Végül a tábla utolsó mezője az **lip, login ip = bejelentkezési ip cím**. Itt az adattípus természetesen **varchar** és **32** karakter hosszúság lett megadva.

– 2.8. A termék tábla –



	id	name	price	image
<input type="checkbox"/>	1	FI-R	2883	FI-R.jpg
<input type="checkbox"/>	3	FS	1502	FS.jpg
<input type="checkbox"/>	4	LM	1501	LM.jpg
<input type="checkbox"/>	5	LM-R	2057	LM-R.jpg
<input type="checkbox"/>	6	RE-L2	1658	RE-L2.jpg
<input type="checkbox"/>	7	RE-V	1295	RE-V.jpg
<input type="checkbox"/>	8	RE-V7	1502	RE-V7.jpg
<input type="checkbox"/>	9	RE-X	1432	RE-X.jpg
<input type="checkbox"/>	10	RF	339	RF.jpg
<input type="checkbox"/>	11	RG-F	2203	RG-F.jpg
<input type="checkbox"/>	12	RG-R	1566	RG-R.jpg
<input type="checkbox"/>	13	RI-A	1677	RI-A.jpg
<input type="checkbox"/>	14	RI-D	1576	RI-D.jpg
<input type="checkbox"/>	15	RI-S	2370	RI-S.jpg
<input type="checkbox"/>	16	RN	886	RN.jpg
<input type="checkbox"/>	17	RP	1195	RP.jpg
<input type="checkbox"/>	18	RS-GT	2940	RS-GT.jpg
<input type="checkbox"/>	19	RT-X	1902	RT-X.jpg
<input type="checkbox"/>	20	RZ-D	1837	RZ-D.jpg
<input type="checkbox"/>	21	SUPER-RS	3448	SUPER-RS.jpg

Ez itt végül a legfontosabb de majdnem a legegyszerűbb tábla a **termek** nevet kapta mivel ő szolgál/felel a **termék bázis** adat tárolásáért. Habár egyszerűnek tűnik feladata rettentő fontos, nézzük hát meg, hogy mit találunk itt. Eme táblán belül összesen **20** egymástól különböző mezőt láthatunk. Mind a termék bázison belül található képeket szolgálják/tárolják, hogy a felhasználó vizuálisan láthassa, mit is vesz pontosan. Ez a tábla a többitől teljesen eltérő. Minden mező **jpg** képet tartalmazó adatot kapott. **Mi is az a jpg? A "jpg" rövidítés egyike a képformátumoknak, és a JPEG**

(Joint Photographic Experts Group) formátumot jelöli. A JPEG egy elterjedt képformátum, amelyet főként digitális fényképek tárolására és megosztására használnak. Itt nagy hangsúlyt fektettem a mezőket megformáló **oszlopokra** amik lényegesen és működő képessé teszik a táblát. Csöppenjünk hát bele és nézzük meg miként működik.

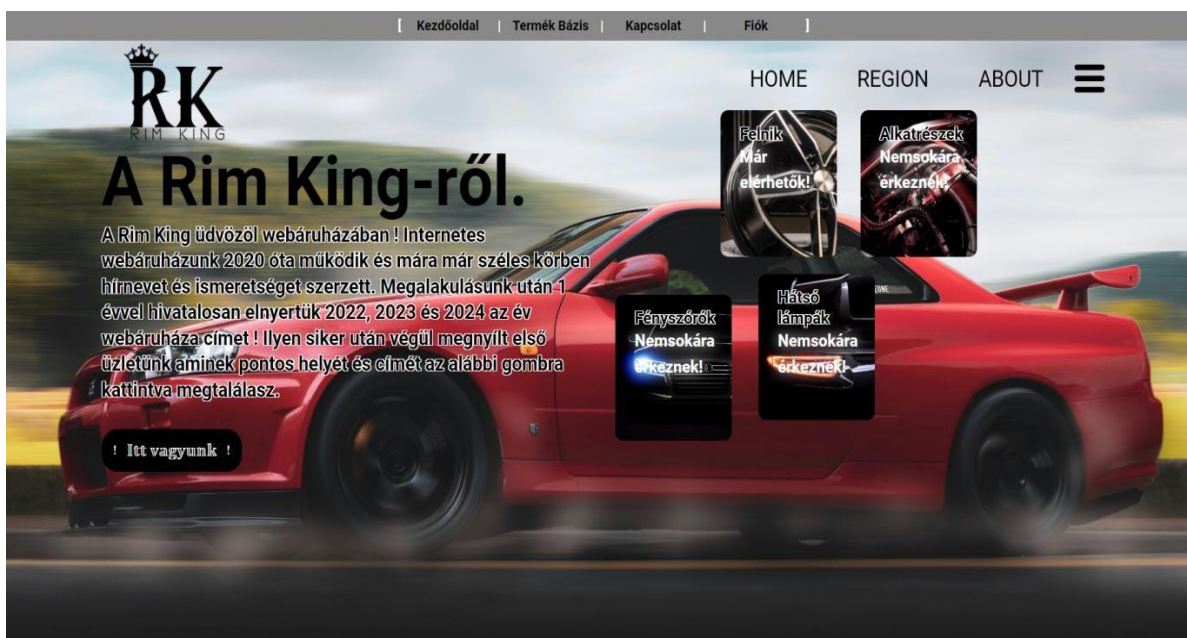
Összesen **4** darab oszlopot láthatunk minden mezőnek van egy egyedileg megadott **id** száma amiket saját kezűleg írtam be az különbözteti meg főképp a termékeket egymástól mikor betesszük őket a kosárba és így tudja a weblap a legkönnyebben feldolgozni mikor kiválasztásra kerülnek a termékek. A

második oszlop a **name** oszlop ugye **name = név** itt adtam meg a termékek pontos nevét amik a weblapon is termékeként megjelennek, nyilván ezzel is alaposabban meg tudjuk őket

különböztetni egymástól mondhatni. Így maradt még két létfontosságú oszlop az egyik nem más mint a **price** nevű oszlop **price = ár** ami az oldalon feltüntetett **árak adatait** biztosítja **dollárban \$** így látják a felhasználók, hogy mi mennyibe is kerül. Végül érkezik az utolsó oszlop ami az **image** nevet képviseli **image = kép** rövidítése lehetne **img** is. Eme oszlopon belül tároltam el a termékekről készült képeket amit a feldolgozó program amár sokszor említett **Visual Studio Code** majd megjelenít a weblapon.

– 2.9. A kezdőlap –

Az adatbázis ismertetése után jöhet a kezdőlap ami már a kódolások soraival, kódolások tömkelegével fog folytatódni. A weblap kezdőoldala angolul a **home/home page** a legelső lap ahová kerülünk miután megnyitottuk az interneten. Vizuális szempontból egész pontosan így néz ki.



A kezdőlapon megfigyelhető egy dupla fejléces menü központ az egyik a szürkével jelölt részen található a másik egyből alatta 4 apró gomb formájában. Az alsó fejléc menü pontjai tovább fejleszthetők ezért nincs teljes körű funkciójuk. Lássuk mit találhatunk még a kezdőlapon belül, látunk egy logót amit én saját kezűleg

készítettem internetes kép szerkesztőben igyekeztem egy letisztult és értelmezhető formát/külalakot adni neki. Egy egyszerű címsorral rendelkezik amit egy `<h1>` taggal hoztam létre és alatta egy `<p>` taggal létre hozott kis leírás helyezkedik el. Nagyon fontos megjegyezni, hogy a kezdőlap elkészítésében eltér a többitől ugyanis HTMLben készült. Mi is az a HTML? A "HTML" vagy a "Hypertext Markup Language"

(Hiperszöveges Jelölőnyelv) egy olyan jelölőnyelv, amelyet weboldalak létrehozására használnak. Az HTML olyan strukturált kódokból áll, amelyek meghatározzák egy weboldal tartalmát és szerkezetét. A HTML kódot böngészők olvassák és megjelenítik, hogy a felhasználók számára látható és interaktív weboldalakat hozzanak létre. Most, hogy ezt tudjuk visszatérhetünk a felfedezésünkhöz ugyanis az említett `<p>` tag alatt van egy gomb amire rákattintva elvisz minket a kapcsolat nevű oldalra/menü ponthoz. Itt össze kellett kapcsoljam a html gombot angolul button-t a kapcsolat oldallal ami már egy php-ban megírt oldal és egy index.php nevű kódlaapon keresztül kellett megoldanom.

Itt látható ahogy egy `<a>` hivatkozással (link-el) létrehoztam egy tag-et és a href segítségével összekapcsolat az index.php-val

```
</p>
<a href="?p=kapcs"><button type="button"> Itt vagyunk ! </button></a>
</div>
<div class="col">
  <div class="card card1">
```

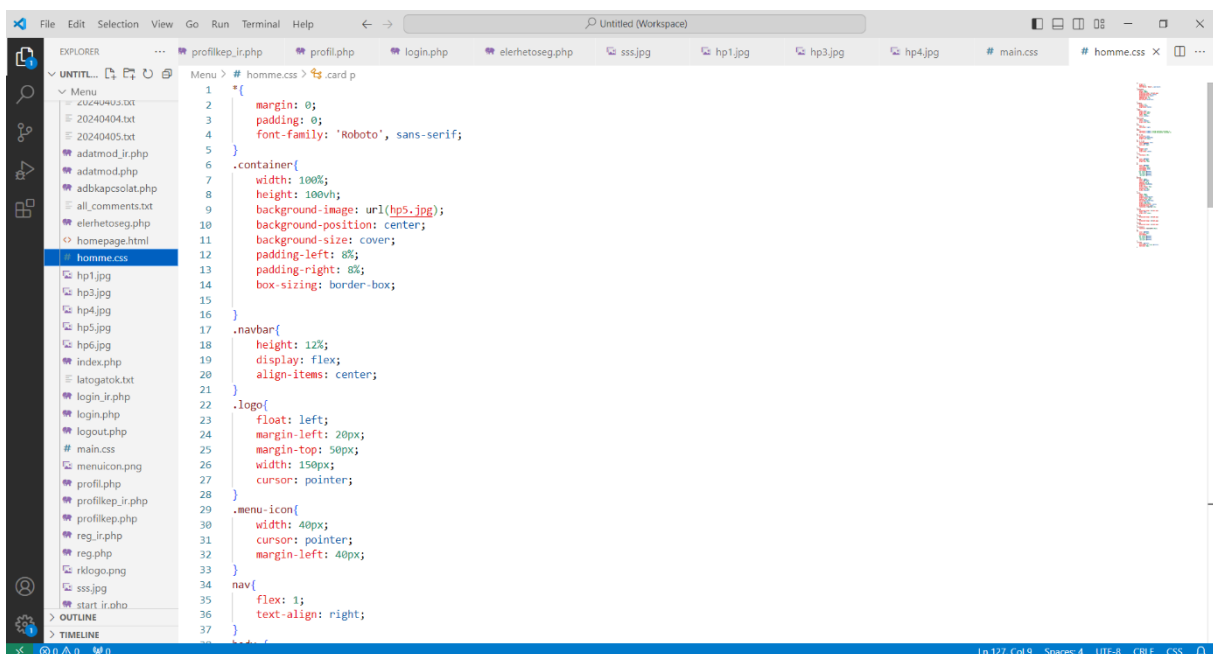
Ezután az index.php-n belül egy case-el megadtam a kapcsolat nevű oldal jelölését és egy include segítségével ahol megvan adva a kapcsolat lap pontos fájl neve tovább irányít minket a megfelelő cél helyre.

```
113         break;
114     case "kapcs":
115         include("elerhetoseg.php");
116         break;
```

Miután ez megvan és megtudtuk miként jutunk el A-ból B-be egyetlen fontos dolog maradt le amit nem vettünk még szemügyre. Az oldal jobb oldalán elhelyezkedő négy kis kártya amik dizájn szempontból érdekesek lehetnek, ám de sok funkcionalitást nem adnak a dologhoz. A kártyákon látjuk milyen termékekhez juthatunk hozzá a weblapon belül és a kártyák is bizonyítják az oldal továbbfejleszthetőségét mivel fehérén kiemelt színnel fel van valami tüntetve.

Egész pontosan az, hogy az oldalon egyelőre csak felnik érhetőek el és a többi jövőre látóan majd érkezik de még nincs raktáron. Így jelzem a

vásárlók/felhasználók felé, hogy mindeképp legyenek résen a jövőben hogyha szükségük lesz valamire. Tehát ebben a részben megismerkedtünk a HTML-el viszont kimaradt a style.css ami a HTML dizájn meg formázására alkalmas. Mi is az a css pontosan szakmailag? A CSS (Cascading Style Sheets) egy stílusleíró nyelv a webfejlesztésben, amelyet arra használnak, hogy megjelenítési jellemzőket, mint például a színek, betűtípusok, elrendezés és egyéb vizuális tulajdonságokat határozzanak meg a HTML és XML alapú dokumentumokhoz. A CSS segítségével elválaszthatjuk a dokumentum tartalmát és szerkezetét a megjelenésétől, így lehetővé téve a különféle stílusok könnyű alkalmazását egy weboldalon vagy akár az egész webhelyen. Egy kis betekintő a CSS-ről :

A screenshot of a code editor interface, likely Visual Studio Code, showing a file named 'homme.css' open in the editor. The file explorer on the left shows a project structure with various files including 'Menu', 'adatmod.php', 'adkapcsolat.php', 'all_comments.txt', 'elerhetoseg.php', 'homepage.html', 'hp1.jpg', 'hp3.jpg', 'hp4.jpg', 'hp5.jpg', 'hp6.jpg', 'index.php', 'latogatok.txt', 'login.php', 'logout.php', 'main.css', 'menuicon.png', 'profil.php', 'profilkep.php', 'reg.php', 'reg_ikr.php', 'rklgo.png', 'sss.jpg', 'start.ir.oho', and 'TIMELINE'. The editor window shows the following CSS code:

```
1 {
2     margin: 0;
3     padding: 0;
4     font-family: 'Roboto', sans-serif;
5 }
6 .container{
7     width: 100%;
8     height: 100vh;
9     background-image: url(hp5.jpg);
10    background-position: center;
11    background-size: cover;
12    padding-left: 8%;
13    padding-right: 8%;
14    box-sizing: border-box;
15 }
16 }
17 .navbar{
18     height: 12%;
19     display: flex;
20     align-items: center;
21 }
22 .logo{
23     float: left;
24     margin-left: 20px;
25     margin-top: 50px;
26     width: 150px;
27     cursor: pointer;
28 }
29 .menu-icon{
30     width: 40px;
31     cursor: pointer;
32     margin-left: 40px;
33 }
34 nav{
35     flex: 1;
36     text-align: right;
37 }
```

Ezen belül találkoztunk még különböző tag-ekkel de mik is azok a tag-ek a HTML-ben? Az "HTML tag-ek" azok a részek, amelyek az HTML dokumentumok struktúráját és tartalmát határozzák meg. Ezek a tag-ek a böngészőknek és más HTML-feldolgozó eszközöknek segítenek megérteni, hogy hogyan kell megjeleníteni vagy feldolgozni az adott tartalmat. Az HTML tag-ek két részből állnak: nyitótag és zárótag, amelyek között helyezkedik el a tartalom. Elöljáróban még beszéljünk az elhangzott php-ről amit az index.php-ban találhatunk meg és, hogy értsük is mi az a php. A PHP egy szerveroldali szkriptnyelv, amelyet dinamikus weboldalak és webalkalmazások létrehozására használnak. A PHP betűszó az "Hypertext Preprocessor" rövidítése. A PHP-t arra tervezték, hogy beágyazható legyen az HTML-be, ami azt jelenti, hogy a PHP kód közvetlenül a HTML kóddal együtt található meg, így dinamikus tartalmat lehet generálni az oldalakon.

Most, hogy ebben a részben megsértettünk egy csomó dolgot mehetünk tovább a következő fontos témákra.

– 2.9.1. A kezdőlapi kártya elkészítése –

A kezdőlapi kártyák már említve voltak ez előbbi pontban és sok mindent megtudtunk róla. De a legfontosabb dolog mögé nem nyertünk még betekintést ami alatt a kódolásra gondolok. Lássuk hát, hogy miként készültek el és milyen kódolás áll az effektek mögött amikor rámegyünk az egérrel, vágjunk hát bele. Ehhez szintén HTML-t és CSS-t használtam és legjobban úgy érhetjük meg ha képekkel demonstrálok és közben adok rá magyarázatot.

Első sorban a homepage.html fájlban belül készítetek egy `<div>`-et aminek adok egy “col” osztály jelölőt, majd ezen belül készítettem négy másik különböző `<div>`-et és mind a négynek más osztály jelölő nevet adtam meg, `card card1`, `card card2`, `card card3`, `card card4`. Miután ez megvan mindnek adtam egy `<h5>`-ös cím sort ahova a kártyák címét/nevét adtam meg, majd alájuk `<p>` tagekkel egy kis leírást. A `<div>`-en belül használt osztály jelölők azért nagyon fontosak mert ezekkel tudjuk megkülönböztetni majd a CSS-ben a formázható HTML kódokat. Az eddig elmondottak valahogy így néznek ki kódban.

```
33 </div>
34 <div class="col">
35   <div class="card card1">
36     <h5>Felnik</h5>
37     <p>Már elérhető!</p>
38   </div>
39   <div class="card card2">
40     <h5>Alkatrész</h5>
41     <p>Nemsokára érkeznek!</p>
42   </div>
43   <div class="card card3">
44     <h5>Fényszórók</h5>
45     <p>Nemsokára érkeznek!</p>
46   </div>
47   <div class="card card4">
48     <h5>Hátsó lámpák</h5>
49     <p>Nemsokára érkeznek!</p>
50   </div>
```

A következő folyamatok már CSS-ben folytatódnak főként. Először is a fő `<div>`-ben megadott “col” osztály jelölőnek adtunk egy úgynevezett flex-basis nevű CSS elrendezési tulajdonságot amivel létrejön egy üres kártya felület és ezt beállítottam 50%-ra.

```

57     .col{
58     |     flex-basis: 50%;
59     }

```

Ezután megfigyelhetjük, hogy mind a négy al `<div>`-ben és az azon belüli osztály jelölőben kettő `card` van. Az egyik sima `card` a másik pedig mondjuk `card1`, ebből az első mindig a főosztály jelölő amivel először megformáztam a kártyát CSS-ben a megfelelő paraméterek szerint a megfelelő kóddal. Megadtam méretet meghatározó paramétereket, formát meghatározó paramétereket, cursor megjelenítő paramétereket és ami az effektér felel az úgy nevezett `transition` amikor rámegyünk a kis kártyára egy 0.5 másodperces intervallumban elvégzi az effekt mozgását. Ez sz egész pontosan így lett megjelenítve/megvalósítva.

```

hp3.jpg      88     .card{
hp4.jpg      89         width: 150px;
hp5.jpg      90         height: 170px;
hp6.jpg      91         display: inline-block;
index.php    92         border-radius: 10px;
latogatok.txt 93         padding: 15px 25px;
login_ir.php 94         box-sizing: border-box;
login.php    95         cursor: pointer;
logout.php   96         margin: 10px 15px;
main.css     97         background-position: center;
menuicon.png 98         background-size: cover;
              99         transition: transform 0.5s;
              100    }

```

Végül de nem utolsó sorban megcsináltam a négy kártyát `card1`, `card2`, `card3` és `card4`-et amihez csak annyit kellett tennem, hogy megadom a kártyákhoz tartozó képeket. A `card1`-nél annyi eltérés van, hogy minimálisan el kellett mozgatnom, hogy igényesen a helyükön legyenek ehhez pedig használtam egy `float` és egy `margin-left` nevű helymeghatározó CSS kódot.

```

)          101     .card1{
)          102         background-image: url(hp1.jpg);
)          103         float: left;
hp         104         margin-left: 150px;
ok.txt     105     }
.php       106     .card2{
hp         107         background-image: url(hp6.jpg);
.php       108     }
ss         109     .card3{
con.png    110         background-image: url(hp3.jpg);
hp         111     }
hp         112     .card4{
hp_ir.php  113         background-image: url(hp4.jpg);
hp.php     114     }

```

A kártya véglegesítéséhez végül egyetlen dolog maradt mégpedig az, hogy megadjuk pontosan hány pixelt mozduljon el az effekt mikor rávisszük a

kurzort. Ehhez használt kód a `card: hover` és ebbe tettem bele a `transform: translateY(-10px);` kódot tehát az effekt -10px-et (pixelt) mozdul el a feladat/effekt elvégzésekor.

```
5  .card: hover{
5  |      transform: translateY(-10px);
7  |  }
```

Nos a kártyák elkészítését, formázását, méretezését és azok effektjeinek kódjába mostmár bele láttunk és megértettük viszont még egy apró dolog hátra van. Ami nem más mint az azon belül elhelyezkedő `<h5>`-ös cím tagek és az alatta tartózkodó `<p>` tagek hiszen őket is meg kell formáljuk. Kezdjük egyből a `<h5>` tagel amit nagyon egyszerű módon szerkesztettünk. Először is adtunk neki egy alap színt a `color` segítségével majd ezután megadtuk a méretét a `font-size` segítő kezével ami 20px lett végül készítettem neki egy fényes hatású külalak kiemelést a `text-shadow`-al. Íme.

```
118  h5{
119  |      color: ■ #000000;
120  |      font-size: 20px;
121  |      text-shadow:
122  |      -1px -1px 0 □ #ffffff,
123  |      1px -1px 0 □ #ffffff,
124  |      -1px 1px 0 □ #ffffff,
125  |      1px 1px 0 □ #ffffff;
126  |  }
```

Legvégül a kártyák legegyszerűbb része a `<p>` tag aminek formázása/szerkesztése igen egyszerű. Nyilvánvaló, hogy neki is adtam egy alap színt a `color` segítségével ami ez esetben fehér színű (nagyon fontos megjegyezni hogy CSS-ben a színeket színekódok alapján adjuk meg aminek elején mindig egy hastag # áll és ezek a színekódok állhatnak betűkből és számokból egyaránt). Az alapszín megadása után neki is megadtam egy fehér színű, fényes hatású külalak kiemelést a `text-shadow` segítségével. majd egy betű méretet a `font-size` segítségével.


```
127  .card p{
128  |      color: □ #ffffff;
129  |      text-shadow: 0 0 15px □ #ffffff;
130  |      font-size: 20px;
131  |  }
```


– 2.10. Termék bázis –

A termék bázis egy rettentő bonyolult felépítésű része a záródolgozatomnak. Mindez php-ban készült úgy, hogy elérhető legyen az index.php-ból és ezzel együtt össze legyen kötve a hozzátartozó adatbázissal is. Az index.php egyébként a központja az egész kódnak minden abból indul ki és minden azon keresztül megy át így ő elengedhetetlen a termék bázis működéséhez. A termék bázis felületét nagyon egyszerűre és könnyen kezelhetővé akartam készíteni, hogy a felhasználó elképesztő könnyedséggel tudja megtekinteni, hogy mit is akar venni és mennyiért. Itt tulajdonképpen kettő darab oszlop van az oldalon belül. AZ egyik a termékek nevet képviseli a másik pedig a kiválasztott termékek nevet tudhatja magáénak. Ez egy görgethető oldal mert a húsz különböző felnit nem akartam beléd nyomorgatni egy pici négyzet alakú helyre. Mindnek saját neve, saját ára és saját képe van így a felhasználó/vásárló egy kisebb /nagyobb választékból tudja eldönteni, hogy mi lesz a megfelelő választás végül. A termékek alatt mind rendelkezik egy gombbal amivel kosárba tudjuk őket tenni és egy egyedileg készített darab számlálóval ahol a vásárló el tudja dönteni, hogy pontosan melyikből mennyit szeretne, hány darabot szeretne megvásárolni. A jobb oldali oszlop címe alatt tartózkodik egy info léc ami megjeleníti a termék id-jét, a termék nevét, a termék darabszámát, végösszegét és hogy melyik terméken végzett kiválasztást a felhasználó az adatbázisból. Miután a vásárló kiválasztotta a számára megfelelő terméket ezután instant átkerül a termék a balból a jobb oldali oszlopba. Több termék kiválasztása esetén esetleg amikre nincs szükségünk azonnal el tudjuk távolítani vagy az esetben ha nincs szükségünk semmire az úgynevezett clear all gombbal az összeset egy gombnyomással el tudjuk távolítani a kiválasztott termékekből. A gombok 2 különböző színnel vannak megjelölve/kiemelve így nem téveszthetjük el, hogy mi is pontosan a szándékunk.

[Kezdőoldal | Termék Bázis | Kapcsolat | Fiók]


Termékek



FI-R

\$2,883.00


Kosárba



FS

\$1,502.00


Kosárba



LM


\$1,501.00

Kosárba




LM-R

\$2,057.00



RE-L2

\$1,658.00



RE-V

\$1,295.00

Kiválasztott Termékek

ID	Termék neve	Termék ára	Termék db szám	Végösszeg	Action
3	FS	1502	1	\$1,502.00	3 Remove
1	FI-R	2883	1	\$2,883.00	1 Remove
6	RE-L2	1658	1	\$1,658.00	6 Remove
				Végösszeg	6,043.00 Clear All

– 2.10.1. Termék info lécs –

A termék info lécrehozás php-ban valósult meg. A kiválasztott termékek cím alá készült igen érdekes megvalósításban. Ezeket szintén leírással és képekkel fogom tudni alaposan és érthetően bemutatni. Az első és legfontosabb, hogy nyitunk egy <php> taget, majd létrehozunk bizonyos output-okat. Miután létrehoztuk az outputokat nyitunk egy table = asztal taget ami ezt a lécszerkezetet fogja biztosítani nekünk és a table tag-en belül adjuk meg osztályjelölővel a formázását hogy milyen vékony legyen és hogy milyen elosztásúak az oszlopok. Az oszlopokat a table-ön belül <tr> és <th> tagekkel valósítottam meg.

```

103 <?php
104
105 $total = 0;
106
107 $output = "";
108
109 $output .= "
110 <table class='table table-bordered table-striped'>
111     <tr>
112         <th>ID</th>
113         <th>Termék neve</th>
114         <th>Termék ára</th>
115         <th>Termék db szám</th>
116         <th>Végösszeg</th>
117         <th>Action</th>
118     </tr>
119 ";

```

Mindezek után nyitunk egy if-el kezdődő session-t, hogy elvégezhessük a dolog logikai részét. A session-ön belül elnevezzük a táblát cart néven ezzel is egyfajta különböztetést adunk a folyamatnak, hogy legyen mi alapján dolgozni. Nagyon fontos a \$key és a \$value mert ezek segítségével tudjuk megadni az oszlopoknak a logikai kapcsolatát. De mivel is kapcsoljuk össze őket? Hát az

oldal bal oldalán tárolt és kiválasztott termékekkel. Nyitunk egy újabb output-ot majd `<td>` tag-es elrendezésben helyezzük el őket és mindezek után mindegyiknek adunk egy logikai értéket az az `$ value-t`.

```
121 |         if (!empty($_SESSION['cart'])) {
122 |             foreach ($_SESSION['cart'] as $key => $value) {
123 |                 $output .= "
124 |                 <tr>
125 |                     <td>".$value['id'].</td>
126 |                     <td>".$value['name'].</td>
127 |                     <td>".$value['price'].</td>
128 |                     <td>".$value['quantity'].</td>
129 |                     <td>".$value['quantity'].</td>
130 |                     <td>".$value['quantity'].</td>
131 |                     <td>".$value['quantity'].</td>
132 |                     <td>".$value['quantity'].</td>
133 |                     <td>".$value['quantity'].</td>
```

Végül az info lécből egy dolog hiányzik ami magyarázatra vár. Az egyik amikor egy termékből több darab számot választunk ki és össze kell adja és a total-t ami a végösszeget jelenti magyarul az összes kiválasztott termék árát összeadja darabszámmal együtt. Az a darabszámok összege amit beledobtunk egy úgynevezett `number_format`-ba majd adtunk neki egy értéket ez esetben a `price = ár`. Végül ezt az árat vagyis `price-t` szoroztuk meg a mennyiséggel ami a kódban `quantity` néven van fent.

```
<td>".$value['price'] * $value['quantity'],2).</td>
<td>".$value['id'].</td>
```

Végül a végösszeg ahol csináltunk egy `$total` nevű értéket összeadtuk a mennyiséggel (`quantity`-vel) majd végül ezt megszoroztuk a `price` nevű értékkel és ezt egyenlővé tettük a `$total`-al ami a végösszeget jelenti.

```
18 |         ";
19 |         $total = $total + $value['quantity'] * $value['price'];
20 |     }
21 | }
```

– 2.10.2. Remove, clear all –

Ez a része a programnak bonyolultnak tűnhet de egyébként egész egyszerű lássuk hogy készültek és miként működnek. A bemutatása és értelmezése a megszokott módon fog haladni most is leírás és kép. Kezdjük a `remove`

gombbal aminek elkészítéséhez nem kellett sokat programozni mindössze egy `<a>` tag-en belül összekapcsoltuk a termékek php-val ami ugyebár az `index.php`-n keresztül fut amit már korábban említettem. Majd kapott egy `$_GET` értéket az `action` oszlopba, hogy működjön de ahhoz, hogy megjelenjen készítenem kellett egy gombot a `<button>` tag-el. Nyilván itt is elengedhetetlen az osztály jelölő és végül a tag bezárása előtt a neve vagyis a `remove` ami angolul annyit tesz, hogy eltávolítás. Ezt aztán beledobtunk egy `<td>` tag-be.

```
<td>
    <a href='?p=termek&id=$_GET[id]&action=remove&id='.$value['id'].'>
        <button class='btn btn-danger btn-block'>Remove</button>
    </a>
</td>";
```

Ennek van egy működtetési kódja mikor rányomunk elvégezze a feladatát. If-el nyitunk majd kap egy `$_GET` értéket összekapcsolja egy session-al majd a `cart`-on belül véghez viszi a törlést/eltávolítást.

```
42     }
43     if ($_GET['action'] == "remove"){
44
45         foreach ($_SESSION['cart'] as $key => $value){
46
47             if ($value['id'] == $_GET['id']){
48                 unset($_SESSION['cart'][$key]);
49             }
50         }
51     }
52 }
53 ?>
```

Ha megvan a `remove` jöhet a `clear all` ami angolul annyit tesz, hogy összes eltávolítása/összes törlése. Szokásos `<a>` tag , `index.php`-n keresztül való elérés egy `$_GET` értékkel megadva egy gomb az az `<button>` tag a hozzátartozó osztály jelölő végül a `<td>` tag-es elrendezés.

Működése hasonlóképpen jár el mint a `remove`-nál. Itt session helyet egy if-es isset-el indítunk amihez hozzájárul egy `$_GET` érték az `action`-höz amivel egyenlővé is tesszük egy másodlagos if-el, aztán egy `unset`-el lezárjuk. Így már működőképes a `clear all` gombunk is.

```
37     ?>
38     <?php
39         if (isset($_GET['action'])) {
40             if ($_GET['action'] == "clearall"){
41                 unset($_SESSION['cart']);
42             }
43         }
```

Így készültek el és eme kódok alapján váltak működőképesé ezek a gombok.

– 2.11. Hibaüzenet az oldalon –

Egyetlen egy hibaüzenettel rendelkezik az oldal ami figyelem felkeltést vet bármilyen rossz húzásunkra felhasználói szempontból. Ez nem más mint a sokak által ismert 404. A kódja az index.php fő fájlban belül található meg. Mivel ez egy úgynevezett alap eszköz egy weblap életében ezért a hozzátartozó kód jelentése sem tesz másképp. Először default-al (default = alapértelmezett) megadunk printbe hozzá egy szöveget amit a break-el (brake = fék/megtörni) zárunk.

```
65 | | | default:
66 | | |     print("404 - Nincs ilyen oldal");
67 | | |     break;
68 | | | }
69 | | ?>
```

Majd nem sokkal lejjebb helyezkedik el ugyanez a default és break parancs csak egy 404-es <h1> tag-ben.

```
135 | | | default:
136 | | |     print("<h1>404</h1>");
137 | | |     break;
138 | | | }
```

3.Felhasználói dokumentáció

– 3.1 Hardware és szoftver igény –

Webes felület révén szükség lesz egy eszközre amivel képesek vagyunk internetezni ilyen például egy laptop vagy esetleg egy asztali számítógép. Továbbá szükségünk van egy tárhelyre például egy szerverre, hogy majd tudjuk publikálni a felületet. Ehhez hozzátartozik, hogy szükségünk van internetre és bármilyen böngészőre az alábbiak közül.

- Google Chrome
- Firefox
- Internet Explorer

– 3.2. A program készítőjének elérhetősége –

A weblapon belül létezik egy Kapcsolat nevű oldal ahol látható a készítő térképen vizuálisan látható elhelyezkedése, ezen felül a címe, email címe és a telefonszáma, hogy bármilyen uton módon képesek legyünk kontaktba lépni a készítővel az az **VELEM**. Bármilyen észrevétellel tud hozzám fordulni a felhasználó.

3.3. Használat és tájékozódás

Mint azt már tudjuk fontos egy weblapról tudni, hogy milyen módon van elkészítve mert ezzel is elősegíti egy felhasználó számára a megfelelő tájékozódást és a könnyű használatot. A weblapon belül kiinduló pontként a fiók szekció fog kelleni nekünk. A fiók szekcióban a felhasználó betudja küldeni a regisztrációját saját felhasználó névvel, email címmel és jelszó megadásával amit a megerősítés miatt kétszer kell megadni. Ezután a küldés gomra nyomva leadjuk a regisztrációt amit automatikusan lement az oldal az adatbázisba mint egy új adat oszlop.

Regisztráció

[Vissza a belépési felületre](#)

A regisztráció befejezése után rányomunk a „Vissza a belépési felületre” gombra és ott megadva az email címet vagy a felhasználó nevet és természetesen a jelszót a küldés gombra kattintva már nem csak regisztrálva vagyunk hanem bejelentkezve is.

Fiók

Ezután az oldal automatikusan átdob minket a termék bázis nevű felületre ahol kezdetét veszi a termékek utáni böngészés. Ha véletlen nem a termék bázis felületre való eljutás volt a célunk akkor a fent megtalálható menü pontok közül eljuthatunk máshova is de egyenlőre maradjunk itt. A termék bázison a felhasználó a görgő segítségével tud tájékozódni és fel le haladni az oldalon áttekintve az összes terméket. Ha valamire szükségünk van csak rákattintunk a „KOSÁRBA” nevű gombra és az általunk kiválasztott termék máris átkerül a lap jobb szegletébe ahol a már „kiválasztott termékek” vannak a hozzájuk szükséges összes információval. A kosárba gomb felett találunk egy sokszorozó rendszert aminek segítségével eldönthetjük, hogy az adott termékből mennyit/hány darabot szeretnénk megvásárolni, természetesen az áru darabszámát az oldal jobb oldala változatlanul feltünteti. A felület jobb oldalán ahol már a kiválasztott termékek várnak elönthetjük melyiket akarjuk véglegesen megtartani. Kitörölhetünk egy adott terméket a remove gomb segítségével és ha vásárlói kedvünk úgy tartja akár egy gomb nyomással eltudjuk távolítani az összeset amihez a clear all nevű gomb lesz segítségünkre.

Termékek



FS
\$1,502.00

Kosárba



LM
\$1,501.00

Kosárba

Kiválasztott Termékek

ID	Termék neve	Termék ára	Termék db szám	Végösszeg	Action
4	LM	1501	1	\$1,501.00	4 Remove
7	RE-V	1295	1	\$1,295.00	7 Remove
6	RE-L2	1658	4	\$6,632.00	6 Remove
8	RE-V7	1502	4	\$6,008.00	8 Remove
Végösszeg				15,436.00	Clear All

A menü pontok között még láthatunk egy Kapcsolat nevű oldalt ahol láthatjuk a készítő térképes elhelyezkedését és több fajta elérhetőségét pl. telefonszám vagy email cím.

A kapcsolat menü pontra nem csak manuálisan juthatunk el hanem a kezdőoldalon található gomb is elvezet minket oda ami „Itt vagyunk” néven található meg. Ha már szóba jött a kezdőoldal menüpont tekintsük meg azt is. A kezdőlap az oldal szíve dizájn és kezdés szempontjából és itt megtudhatunk kicsit többet a weblap történetéről. Itt helyezkednek el a weblap termék katalógusáról készült előre jegyzések kis mozgó kártyák formájában.



Így lesznek képesek a felhasználók kezelni és tájékozódni a weboldalon belül amihez hozzájárult egy letisztult formavilág és egy gondosan megtervezett oldal és persze annak megvalósítása.

4.Összefoglalás

– 4.1. A szakdolgozat célja –

A célja az volt, hogy készüljön egy letisztult könnyen kezelhető termék bázis alapú rendszer amit a felhasználó könnyedén átláthat. Ez tartalmazzon egy login-t és egy kezdőlapot amik ugyanúgy letisztult kontextusban helyezkednek el.

– 4.2. Célkitűzések, amik nem sikerültek teljességgel –

Az oldalon belül találhatóak olyan gombok, kártyák, felületek, menüpontok amik funkcionálisan nem 100%-osan működnek. Mint például a kezdőlapon a menüsor felül a HOME, REGION, ABOUT és a lenyíló menü szimbólum gombja. Ezekhez pl. volt tervem amit az időtartam miatt nem sikerült tovább fejleszteni / megvalósítani (A vizsgálóhoz adott határidő időpontja miatt / időtartama miatt). Szintén a kezdőlapon helyezkedik el a négy kártya amit tovább akartam fejleszteni hogy ha rányomunk elvigyen minket más más lapokra a webes felületen belül. Végül ami hátramaradt termék bázison belül kimaradt az úgynevezett tovább a fizetéshez gomb Amivel a termékek kiválasztása után fizetni tudtunk volna, ez azért nem valósult meg mert a weblap nincsen szerzőtetésben egyetlen bankkal sem a vásárláshoz.

5. Irodalomjegyzék

– 5.1. A szakdolgozathoz használt források –

A szakdolgozathoz használt források:

1. <https://www.w3schools.com/> A szakmai igazításokhoz (apróságokért)
2. <https://www.youtube.com/> Nehezebb kódok megvalósításához
3. <https://infojegyzet.hu/> Ötletek gyűjtéséhez
4. <https://helyesiras.mta.hu/helyesiras/default/suggest> Helyesírás javítás
5. <https://stackoverflow.com/> Pár ismeretlen kód megértéséhez

A fejlesztő elérhetősége:

- email : pulszterkartya@gmail.com
- telefon : 06301012171