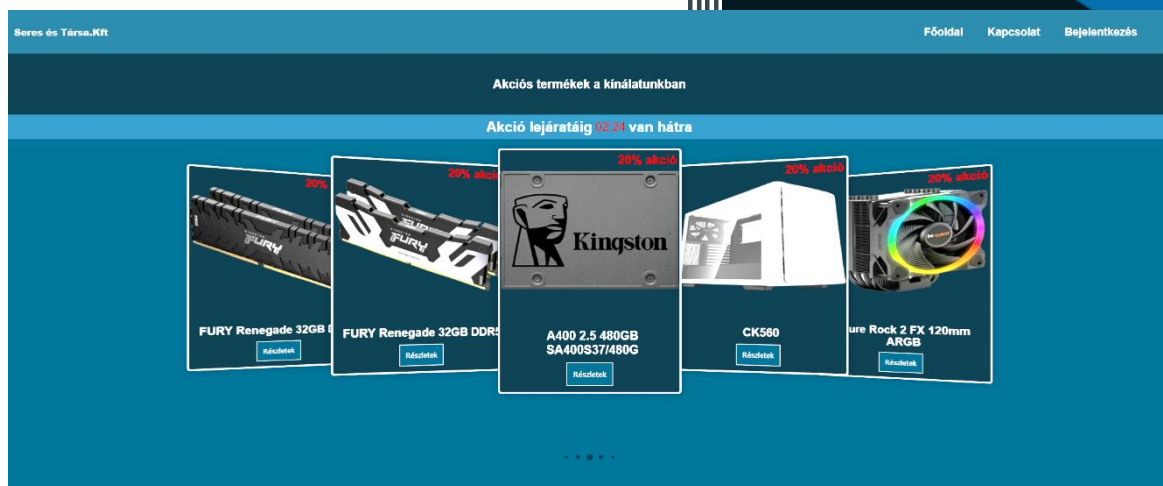


2023

SZÁMÍTÓGÉP ALKATRÉS Z WEBSHOP



Webshop alkalmazás adatbázissal, JavaScripttel és PHP backend-el

Laczka Adrián Zsolt és Seres Szabolcs
Nógrád Megyei Szakképzési Centrum
Szent-Györgyi Albert Technikum
szoftverfejlesztő és -tesztelő technikus
szakma azonosító: 5-0613-12-03
2023.04.25.

Tartalom

Bevezetés.....	4
1. Felhasználói dokumentáció.....	6
1.1. Kezdőoldal	6
1.2. Bejelentkezés oldal	8
1.3. Termékek a kategóriákon belül.....	9
1.4. További adatok az adott termékről	10
1.5. Kapcsolat menü.....	11
1.6. Kedvenc termékek menüpont	12
1.7. Kosár tartalmának megtekintése	13
1.8. Megrendelés menete	13
1.9. A rendelés-lezárás oldala	14
1.10. Adminisztrációs felület	15
1.10.1. Új gyártó hozzáadása	16
1.10.2. Új termék hozzáadása	17
1.10.3. Termék módosítása és törlése	18
1.10.4. Kedvelt termékek	18
1.10.5. Megrendelések megtekintése	19
1.10.6. Felhasználók kezelése	19
1.11. Illetéktelen belépés az admin oldalra.....	20
2. Fejlesztői dokumentáció	21
2.1. Felhasznált technológiák	21
2.1.1. Visual Studio Code.....	21
2.1.2. HTML5	21
2.1.3. CSS3	22
2.1.4. JavaScript.....	22
2.1.5. AJAX	22
2.1.6. PHP	22
2.1.7. XAMPP.....	23

2.1.8.	GitHub	23
2.1.9.	Composer	23
2.1.10.	CKEditor	23
2.1.11.	Dompdf	24
2.1.12.	WinSCP	24
2.2.	A weboldal programozása	24
2.2.1.	Felhasználók kitiltása.....	24
2.2.2.	A népszerű termékek megjelenítése a főoldalon	25
2.2.3.	Belépés.....	26
2.2.4.	Termékek felvitele az adatbázisba	30
2.2.5.	Termékek hozzáadása a kosárhoz, és mentése localStorage-be	30
2.2.6.	Adatok hozzáadása a kosárhoz	32
2.2.7.	A rendelés hozzáadása a kosárhoz.....	34
2.2.8.	Törlés a kosárból.....	35
2.2.9.	Számla kiállítása	36
2.2.10.	Kapcsolat	39
2.2.11.	Egyéb felhasznált eszközök	41
2.3.	Admin oldal	41
2.3.1.	Főoldal	41
2.3.2.	Új gyártó felvitele	44
2.3.3.	Új termék felvitele	45
2.3.4.	Népszerű termékek	47
2.3.5.	Megrendelések megtekintése	47
2.3.6.	Felhasználók kezelése	49
2.3.7.	Törlés	50
2.3.8.	Termék módosítása	51
2.3.9.	Részletek.....	51
3.	Adatbázis	52
3.1.	Kiinduló egyedek	52
3.1.1.	Kategória.....	52
3.1.2.	Gyártó	52
3.1.3.	Felhasználó	52

3.1.4.	Termék	52
3.1.5.	Felmerült kérdések.....	53
3.1.5.1.	Népszerű termékek létrehozása	53
3.1.5.2.	Az admin és a felhasználó egyed szétválasztásának kérdése	53
3.1.5.3.	A regisztrációs kötelezettség	53
3.1.5.4.	A gyártó és kategória kapcsolatának kérdésköre	53
3.1.5.5.	A lakcím – szállítási cím tárolásának kérdése	53
3.1	Kapcsolatok meghatározása.....	53
3.3.	Táblák.....	54
3.3.1.	A kategória tábla.....	54
3.3.2.	A gyarto tábla.....	55
3.3.3.	A gyartokategoria tábla.....	55
3.3.4.	A termék tábla.....	56
3.3.5.	A megrendeles tábla.....	57
3.3.6.	A kedvenctermekek tábla.....	58
3.3.7.	A telepulesek tábla.....	58
3.3.8.	A users tábla.....	59
3.3.9.	Adatbázis diagram	60
4.	Tesztelés.....	61
	Összefoglalás.....	59
	Források.....	63
	Ábrajegyzék.....	64

Bevezetés

Mára már szinte mindenkinek természetessé vált az internet napi használata. Sokan az okostelefonjukról olvassák a híreket, nem nyomtatott lapokból, és online rendelnek termékeket ahelyett, hogy kilométereket utaznának egy-egy speciális üzletig.

Nem kell egy eladó véleményére hagyatkoznunk, hanem fórumokon tudunk tájékozódni akár egy termékről, vagy akár egy webshopról, hogy az mennyire megbízható. Bármilyen termékhez könnyedén hozzájuthatunk, legyen az a világ bármely pontján.

Ebből fakadóan szinte már mindenki rendelkezik különböző informatikai eszközökkel, amelyek, ahogyan telnek az évek, lassan elavulnak, vagy bővítésre szorulnak, a javításokról nem is beszélve. Ezért úgy gondoltuk, hogy egy igen széles vásárlói közönséget tudunk megszólítani azzal, ha egy informatikai alkatrészeket értékesítő online üzlet weboldalát készítjük el.

Az általunk elkészített webshopból kiválaszthatók a különböző termékkategóriák, mint például processzorok, vagy videókártyák, és ezeken a termékkategóriákon belül találhatóak a termékek. A fejlécben az akciós termékeket láthatja a felhasználó, amelyek időközönként változnak. A különböző termékkategóriákon belül keresni is tudnak a weboldalunkon.

A mi webshopunk abban különbözik a többitől, hogy ha egy felhasználó nem jelentkezett be, akkor a termékek alatt nem is jelenik meg a kosár gomb, csak a bejelentkezés után mivel úgyis csak regisztrált felhasználó tud rendelni. Ellenkező esetben kiválaszthatja, miből hány darab terméket szeretne rendelni. A rendelés végén a programunk egy számlát is generál részére. A vásárlók panasz esetén üzenetet is tudnak küldeni a weboldal felületéről, mely az adminisztrátorok e-mail címére kerül továbbításra.

Az oldalunkhoz létrehoztunk egy adminisztrációs felületet is, ahol az új termékeket fel tudjuk tölteni, illetve az adatait módosíthatjuk, vagy törölhetjük. Minden oldalon az adatok beolvasása az adatbázis különböző tábláiból történik. Amennyiben a felhasználó rendel valamilyen terméket, és a megrendelését megerősíti, azaz megadja a számlázási adatait, a megrendelése bekerül az adatbázisba további feldolgozásra. Továbbá az admin inaktíválhatja a felhasználói fiókokat.

Dolgozatunk első része a felhasználói dokumentáció, melyen belül részletesen kitérünk oldalaink menüpontjaira, hogyan tud könnyedén eligazodni a kezdő felhasználó is a webshopunkban.

A második fejezetben, a fejlesztői dokumentációban bemutatjuk a felhasznált technológiákat, az adatbázis-tervezés folyamatát, és az adatbázisunkat, majd ezt követően kiemeljük azokat a programrészleteket, amelyekre különösen büszkék vagyunk, hogy meg tudtuk oldani, mint például a számla kiállítása, vagy a bejelentkezés hibakezelése stb.

A harmadik fejezetben a tesztelés folyamatát mutatjuk be. Végezetül az összefoglalásban értékeljük a közös munkánkat, és sorra vesszük a továbbfejlesztési lehetőségeket, mint fizetési rendszer kialakítása, komplett számítógép tervezése.

1. Felhasználói dokumentáció

1.1. Kezdőoldal

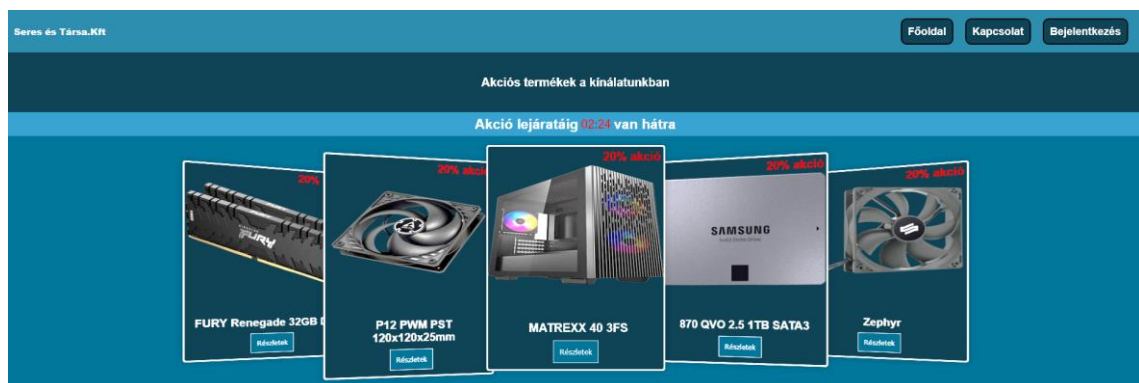
Weboldalunk kezdőoldalának felső sávjában helyezkedik el a navigációs menü, melyben a baloldalon a cég logója és 3 menüpont található a jobb oldalon. A „Főoldal”-ra kattintva bármely más oldalról visszatérhetünk a fő oldalra. A „Kapcsolat”-ra kattintva üzenetet tud küldeni a bejelentkezett felhasználó az adminnak. A „Bejelentkezés”-re kattintva a bejelentkező oldalra jutunk el, ahol regisztrálhatunk, vagy ha már van regisztrációs fiókunk, akkor bejelentkezhetünk, ekkor már több menüpont közül választhat a felhasználó.



1. ábra Menü belépett felhasználó esetében

Amint a fenti képen is látható, ez esetben a bejelentkezés gomb eltűnik, és a kijelentkezés jelenik meg helyette, ezzel jelezve, hogy be van lépve a felhasználó. Ezután található egy visszaszámláló, ami a random kiválasztott akciós termékek idejét mutatja. Ha lejár, akkor új akciós termékek jelennek meg. És végül a kosár gomb, ami jelzi, hogy hány terméket tett a kosarába a vásárló.

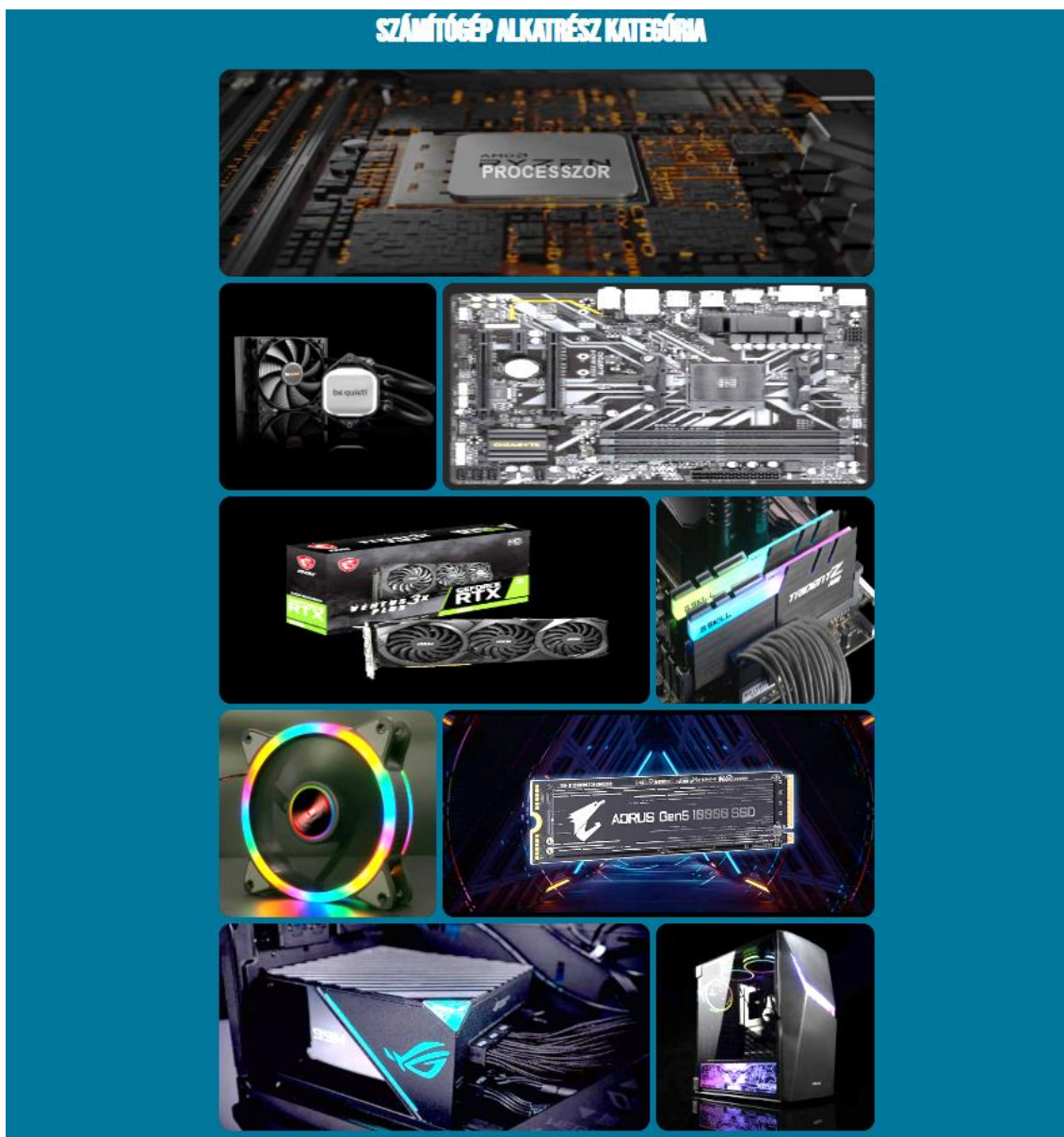
A menü alatt az akciós termékek jelennek meg horizontálisan mozgatható képkockákban, ahol a „részletek” gombra kattintva a kiválasztott termék oldala jelenik meg az akciós árral együtt.



2. ábra Menü és akciós kategóriák carousel

Ezekbe a kártyákba azok a termékek kerültek bele, amelyeket a program random 2 percenként választ ki, és 20%-kal áraz le. Az egyszerűbb szemléltetés végett csak 2 percet állítottunk be a váltásra. Nyilván ez nem életszerű, és éles weboldal esetében, majd azokat a termékeket árazzuk le, amelyeket üzleti szempontból érdemes lesz.

Az akciós termékek alatt a kategóriák fotói jelennek meg, melyekre, ha rátoljuk az egeret, akkor megjelenik a kategória neve, mint például processzor, processzorhűtő, alaplap, videókártya, memória, rendszerhűtő, SSD, merevlemez, tápegység, számítógépház. Ha valamelyikre rákattintunk, akkor elnavigál az adott termékkategóriába tartozó termékekhez.



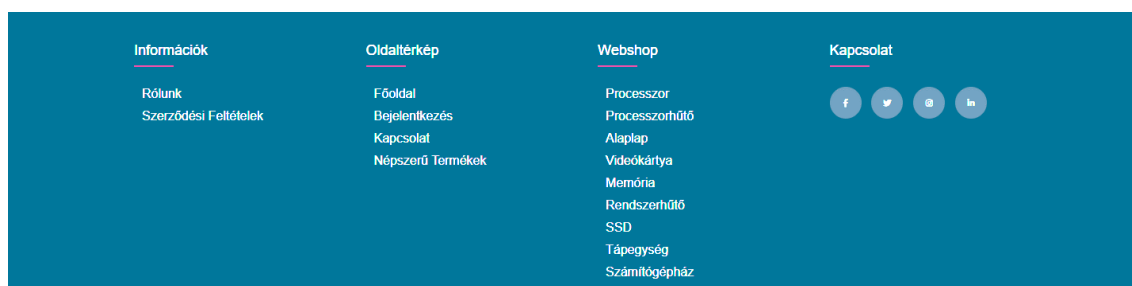
3. ábra Termékkategóriák a kezdőoldalon

Tovább görgetve lefelé az oldalon leválogatásra kerültek azok a termékek, amelyekre a legtöbb kedvelést adták a vásárlók. Itt csak a 15 legkedveltebb termék jelenik meg, amelyeket az alattuk található csúszkával tudunk mozgatni.



4. ábra Kedvelt termékek a kezdőoldalon

Az oldal alján a lábléc található, amelyben a kapcsolati adatok, valamint közösségi oldalakra vezető ikonok jelennek meg. Ezekhez nem rendeltünk hozzá az adott oldalakra mutató linket, mivel még nem hoztuk létre a webshop Facebook, Twitter, Instagram és LinkedIn oldalait.



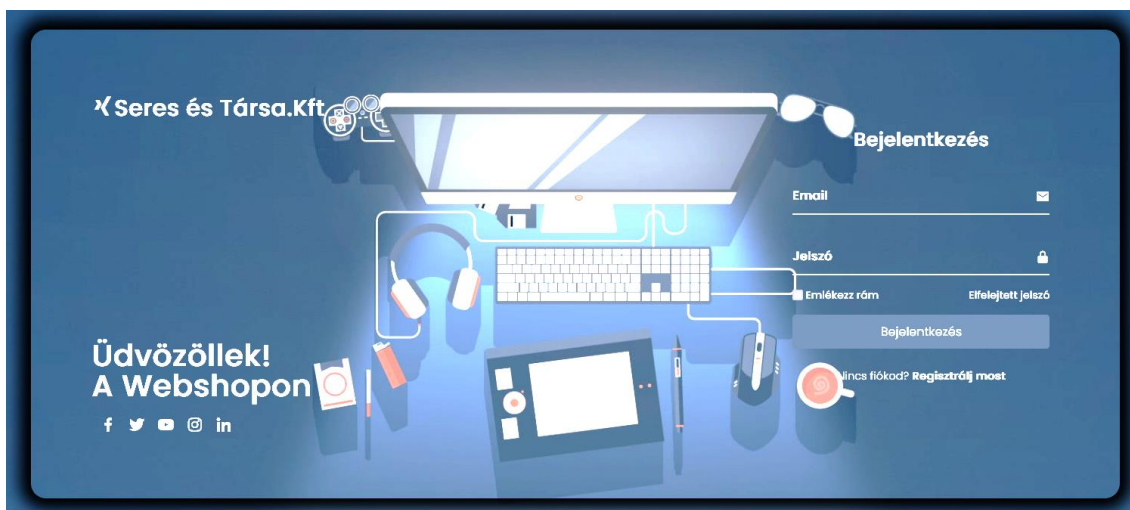
5. ábra Lábléc

1.2. Bejelentkezés oldal

A bejelentkező oldalon a felhasználó be tud lépni a saját e-mail címe és jelszava használatával. Ha esetleg a már regisztrált vásárló elfelejtette volna a jelszavát, akkor az „Elfelejtett jelszó” szövegre kattintva majd új jelszót kérhet, de ennek megvalósítása már a továbbfejlesztési terveink között szerepel. Ha még nincsen fiókja, akkor a regisztráció gomb megnyomására megjelenik a regisztrációs űrlap. Csak a belépett felhasználók esetében jelenik meg a termékek alatt a kosár gomb, és a darabszám kiválasztása.

A regisztrálónak meg kell adnia a vezetéknevét, az e-mail címét, a jelszavát, melyet megerősítés céljából ismételtten be kell gépelnie az űrlapra. Alatta található egy választónégyzet, amivel elfogadjuk az általános szerződési feltételeket (ÁSZF), ami kötelező.

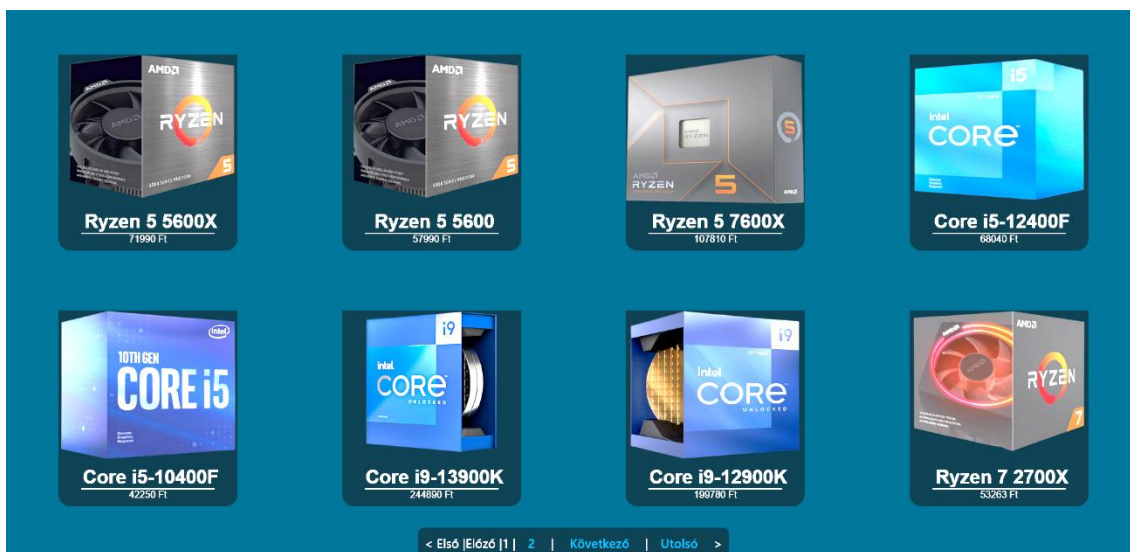
Ha a regisztrációnál kihagy egy űrlapmezőt a felhasználó, akkor a regisztrálót figyelmezteti, hogy melyik mezőt nem töltötte ki.



6. ábra Bejelentkezés oldal

1.3. Termékek a kategóriákon belül

Ha kattintunk egy termékkategóriára, akkor megjelennek az adott csoporthoz tartozó termékek.



7. ábra Termékek, ha a felhasználó nem lépett be az oldalra

A termékeknel található alul egy lapozó, amely a fenti képen is látható. A weboldalak hőtérvkép vizsgálatai alapján megállapították, hogy felhasználók többsége gyakran nem görget az oldal legaljára. Mivel a webshopunkban sokat kellene görgetni lefelé, hogy az összes termék megjelenjen a felhasználó előtt, ezért jobb megoldásnak tartottunk egy lapozó elkészítését. Egy oldalon 8 termék jelenik meg, amely kitölti a teljes képernyőt, és a lapozóval tud a következő oldalakra lépni a felhasználó. Az oldal alján megtalálható a minden oldalon feltűnő lábléc is.

Ha a felhasználó nem lépett be az oldalra, akkor csak a termék neve és ára jelenik meg. Ha bejelentkezett felhasználóként tekinti meg a termékek oldalát, akkor a termék fotója alatt egy szív ikon, a termék neve, ára és alatta a választható darabszám mellett a kosár gomb is megjelenik.



8. ábra Termékek megjelenése a belépett felhasználó esetében

Ha egy termék megtetszik egy vásárlónak, akkor a termék fotó alatti szív ikonra kattintva adhat le egy szavazatot a termékre. Szavazás után ez piros színnel jelenik meg, mint ahogyan a fenti képen is látható. Minél több szavazat érkezik egy termékre, annál előbb kerül besorolásra a fő oldalon a népszerű termékek közé.

Ebben az esetben már megjelenik egy mező, ahol megadhatjuk a megvásárolni kívánt termék mennyiségét. Mellé a „Kosárba” gombot helyeztük el, amelyre ha rákattintunk, a termék a kosárba kerül. Ha szeretné megtekinteni a vásárló a kosara tartalmát, akkor a menüben a jobb felső sarokban lévő kis kosár ikonra kell navigálnia. Ha egy adott termék már nincs készleten, akkor figyelmezteti a felhasználót a termék neve alatt, hogy elfogyott és nem tud belőle már vásárolni.

1.4. További adatok az adott termékről

Ha kattintunk egy termékre, akkor egy új oldal jelenik meg, ahol egy rendezett táblázatos formában annak valamennyi tulajdonságát megjelenítettük.

Seres és Társa.Kft

Főoldal Kedvenc termékek Kapcsolat Latiza Kijelentkezés 0

be quiet! Pure Rock 2 (BK007)

Típus	Aktív hűlő
Hűlő típusa	Processzor
Ventilátor átmérője	120 mm
Ventilátor fordulatszáma	1500 rpm
Maximális zajszint	26.8 dB
Levegőtáplálás	87 CFM
LED megvilágítás	Nincs
Méretek	120 x 120 x 25 mm
Tömeg	575 g

Eredeti ár: 46 170 Ft
Új ár: 12 936 Ft

9. ábra Bővebb információk egy kiválasztott termékről

Ha a termék épp leárazásra került, akkor a részleteknél és a termékeknél is az akciós árat jeleníti meg, mint ahogyan a fenti képen is látszik.

1.5. Kapcsolat menü

Üzenet küldése

Teljes Név*:
Ruzsinszki Zita

E-mail*:
rweb.guide@gmail.com

Ön által jelentett probléma*:
Még nem érkezett meg a rendelésem, meg tudnák írni mikorra várható a processzor, amelyet rendeltem? Válaszokat előre is nagyon szépen köszönöm!!

A * -al jelölt mezők kitöltése kötelező!

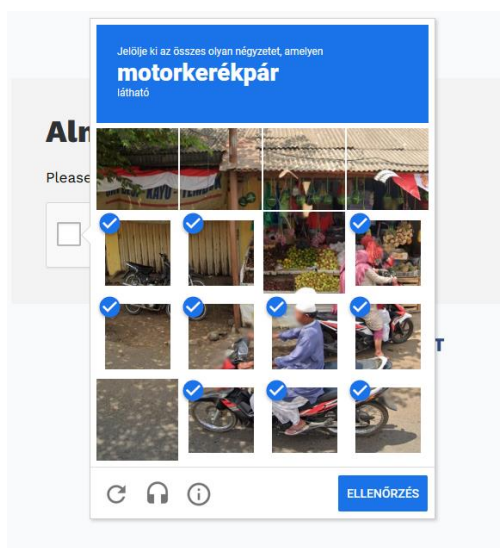
Vissza

Elküldés

11. ábra Kapcsolati űrlap

A Kapcsolat űrlapot azért hoztuk létre, hogy a felhasználóink tudjanak nekünk visszajelzést küldeni a termék állapotáról, esetleg kérdéseket vethetnek fel, amikre mi szívesen válaszolunk.

A visszajelzéseket e-mailben fogadjuk. A felhasználónak meg kell adni kötelezően a nevüket, e-mail címüket és az üzenet szövegét. A Vissza gombra kattintva törlődnek a felhasználó által kitöltött adatok. Az Elküldés gombra kattintva először egy „Captcha” üzenet jelenik meg, amelybe ha belekattintunk, generál egy megoldandó feladatot. Ha ezt



10. ábra Captcha kód generátor

sikeresen megoldja a felhasználó, akkor továbbításra kerül az üzenetük az e-mail címünkre, a képernyőn pedig megjelenik egy köszönő üzenet.

PcWebshop visszajelzés [Beérkező levelek x](#)

FormSubmit <submissions@formsubmit.co>
címezett: én ▼

Someone just submitted your form on <http://www.projectmunka.nhely.hu/>.

Here's what they had to say:

Name	Value
name	Ruzsinszki Zita
email	itweb.guide@gmail.com
message	Üdvözlöm! Még nem érkezett meg a processzor amit rendeltem. Meg tudnák mondani mikorra várhatom? Előre is köszönöm szépen a válaszukat! Cikk szám: 7

Submitted at Thu, Apr 20, 2023 6:59 AM (UTC)

12. ábra Kapott e-mail, a felhasználó által küldött üzenetről

1.6. Kedvenc termékek menüpont

A navigációs menüben a „Főoldal” és a „Kapcsolat” között találhatjuk meg a „Kedvenc termékek” menüpontot, ez csak akkor jelenik meg, ha a felhasználó bejelentkezett a fiókjával. A menüre, ha rákattintunk, akkor egy új oldalon megjelennek az adott felhasználó kedvelt termékei, amelyeket megjelölt. A termékek egy táblázatos formában jelennek meg, jelezve az adminoknak is egyúttal, hogy milyen termékeket kedvelnek a felhasználók, esetleg melyikből érdemes többet rendelni a raktárba. A táblázatban tudja a felhasználó törölni termékeit, amelyeket mégsem kedvel. Egy felhasználó egy termékre csak egy szavazatot tud leadni.

Főoldal			
Az Ön által kedvelt termék(ek):			
Fotó	Termék neve	Termék ára	Művelet
	Radeon RX 7900 XT 20G	375200	Törölés
	Radeon RX 6800 Fighter 16GB	217700	Törölés
	ARC A770 16GB DDR6	155750	Törölés

13. ábra Felhasználó által kedvelt termékek

1.7. Kosár tartalmának megtekintése



14. ábra Kosár tartalmának megtekintése

A kosár gombra kattintva egy felugró ablakban megjelennek a kosárba helyezett termékek. Ha több terméket rendel egy vásárló, akkor görgethetővé válik az ablak.

A táblázatban elhelyezett termékek közül tud törölni termékfajtánként is, illetve az „összes törlése” gombra kattintva ki tudja üríteni a kosarát. A kis ablak tetején a jobb felső sarokban található a bezárás gomb, amellyel be tudja zárni ezt a felugró ablakot. A “Megrendelés elküldése” gombra kattintva véglegesítheti a rendelését a felhasználó. Ha a vásárló rákattint, figyelmeztető üzenet jelenik meg, hogy „Biztosan megrendeli a termékeket?”. Ha az „Igen, megrendelem”-re kattint, akkor megjelenik a következő üzenet: „Átírányítjuk az adatok megadása oldalra”. Ha eltelt 3 másodperc, akkor átirányítja a vásárlót egy új oldalra, ahol meg kell adnia az adatait.

1.8. Megrendelés menete

A megjelenő űrlapon a felhasználónak meg kell adnia az érvényes adatait a rendelés véglegesítéséhez, mint vezetéknév, keresztnév, a bankkártya adatai (bankkártya szám, CVV kód), szállítási cím és telefonszám.

Termék(ek) megrendelésének véglegesítése

Keresztnév * Zita

Vezetéknév * Ruzsinszki

Bankkártyaszám * 123456789-123456789-123456789

CVV Kód * 286

Telefonszám * 06302432455

Irányítószám: * Salgótarján (3100)

Utca név: * Munkás

Házzám: * 6

☒ Adatok elmentése(?)
Az adatok elmentésére rákattintva el tudja menteni az adatait ez által nem kell kitöltenie az újbóli vásárlása során!

Rendelés véglegesítése

Megrendelés törlése

15. ábra Rendelés megerősítése, adatok megadása

Az érvényes adatok megadása után a rendelése feltöltésre kerül az adatbázisba, és a felhasználó átirányításra kerül a rendelés lezárása oldalra, ahol letöltheti a vásárlási számláját. Ha valaki újra ugyanazzal a felhasználói fiókkal szeretne vásárolni, akkor az adatokat nem kell újra begépelnie, a program automatikusan megjeleníti azokat az űrlapon, amelyeket módosíthat. Adatmódosítás után a mentés gombra kattintva felülírásra kerülnek az adatai az adatbázisban.

Ezen a ponton fontos megemlítenünk, hogy amikor a felhasználó véglegesíti a rendelését, akkor a raktárkészleten lévő termékek darabszámából is kivonásra kerül a megrendelt termékek darabszáma.

1.9. A rendelés-lezárás oldala

Itt egy üzenet jelenik meg: „Köszönjük rendelését”. Alatta három gomb látható. Az egyik PDF-et generál, vagyis egy A4-es méretű számla jelenik meg. A másik gombbal kérheti a kinyomtatott PDF mellé postai számla kiküldését is. Az utolsó gombbal visszatérhetünk a főoldalra.

A PDF-ben, amit letöltöttünk, annak jobb felső sarkában a saját webshopunk elérhetőségei jelennek meg, pontosabban egy telefonszám és egy e-mail cím. Alatta a „Számla bizonylat”, ezt követi a számla sorszáma. A következő szakaszban beolvasásra kerülnek a céges adatok, valamint a vásárló adatai is.

Bizonylat sorszáma:		SZ00010/2023	
Kinyomtatás dátuma: 2023.04.26 12:03:42			
Cég	Vásárló		
Seres és Társa KFT	Zita Ruzsinszki		
Bankszámlaszám: 123456789-123456789-123456789	Bankszámlaszám: 123456789-123456789-123456789		
Helyszín: 1022, Budapest I.kerület Domb utca 11.	Helyszín: 3100, Salgótarján, Munkás utca 6		
E-mail cím: onlinePcWebshop2023@gmail.com	E-mail cím: itweb.guide@gmail.com		

Termék cikkszama	Termék neve	Darab	Eredeti Ár	Akciós Ár	Akció
415	ASUS TUF GAMING B660M-PLUS D4	1	58 740 Ft	58 740 Ft	0 %
416	ASUS TUF GAMING B560M-PLUS WIFI	1	51 190 Ft	40 952 Ft	20 %
Végösszeg					99 692 Ft

16. ábra Letölthető PDF számla

Ezek alatt egy táblázat került felsorolásra az összes megrendelt termékkel együtt. A táblázat tartalma: termék azonosítója, maga a termék neve, darabszáma, hogy miből mennyit rendelt a vásárló, a termék ára és végül az akció mértéke. Az utolsó termék alatt található a végösszeg.

1.10. Adminisztrációs felület




Az adminisztrációs felületre kizárólag csak az oldal tulajdonosa tud belépni. Ez már a felhasználó számára nem látható. Itt már a menünél található piros szöveggel megjelenik az „admin oldal” felirata, valamint a bejelentkezett admin neve, ha az oldal tulajdonosa lépett be.

Főoldal	Kapcsolat	DrSeres	Admin oldal	Kijelentkezés
----------------	------------------	----------------	--------------------	----------------------

17. ábra Adminisztrátor számára megjelenő menü

Az admin oldalra kattintva megjelenik az összes termék egy táblázatban, amelyben dinamikusan tudunk keresni. Itt kereshetünk kategóriára, gyártóra, vagy akár terméknevre is, azonnal azok a termékek kerülnek leválogatásra, és csak azok jelennek meg a böngészőben.

A táblázat 7 oszlopból áll. Az első oszlopban a termék képe, a másodikban a kategória neve, a harmadikban a gyártó neve, a negyedikben a termék konkrét típusa, az ötödikben az ára forintban, a hatodikban a termékből hány darab van raktáron, az utolsóban pedig 3 gomb található, melyek a törlés, módosítás és részletek jelennek meg. Ha egy adott termék elfogyott, akkor piros mezőnevekkel figyelmezteti az adminisztrátort ennek tényéről.

TERMÉKEK						
<div> Új gyártó hozzáadása Új termék hozzáadása Rendszeri termékek Megrendelések megtekintése Felhasználók kezelése </div> <div>Vissza a webshopra</div> <div>Keresés</div>						
Kép	Kategória	Gyártó	Termék	Ár	Raktáron	Műveletek
	processzor	AMD	Ryzen 3 1200	11 000 Ft	5 darab	Törölés Módosítás Részletek
	processzor	AMD	Ryzen 5 5600	57 990 Ft	29 darab	Törölés Módosítás Részletek
	processzor	AMD	Ryzen 5 7600X	107 810 Ft	32 darab	Törölés Módosítás Részletek

18. ábra Termékek az admin oldalon

1.10.1. Új gyártó hozzáadása

Felvihetünk új gyártókat, ahol választó menüből kiválaszthatjuk a forgalmazott termékkategóriát, majd megadhatjuk a gyártó nevét, valamint alul lenyitható elemként megtekinthetjük a már felvitt gyártókat.

Új gyártó hozzáadása

Kategória kiválasztása*: Alaplap ▼

Márka / Gyártó*:

A*-gal jelölt mezők kitöltése kötelező.

Rendben

Vissza az oldalra

▼ Meglévő táblák megtekintése

Gyártó	Kategória
AMD	processzor
INTEL	processzor

19. ábra Új gyártó felvitele

Ha a rendben gombra kattintunk, akkor az adatbázisban is mentésre kerül az új gyártó, majd visszatérhetünk a főoldalra.

1.10.2. Új termék hozzáadása

Felvihetünk új termékeket is, ahol választó menüből ki kell választanunk a termék kategóriáját, és meg kell adnunk a termék nevét. A részletes leírásnál beépítettük a ckeditort, hogy táblázatot is be tudjunk illeszteni, vagyis már eleve megformázott szöveget is akár, amely így eleve táblázatban, vagy a kiválasztott formátumban fog megjelenni a termék részleteinél. Ezt követően meg kell adnunk az árát, hány darab van belőle, majd fotót is szükséges feltölteni hozzá az adott termékről, a rendszerünkben a PNG, JPEG és GIF formátumot engedélyeztünk. Ezek megadása után menthetjük el az adatokat, azaz bekerülnek az adatbázisba. Ha valamilyen adatot nem töltünk ki, a program figyelmeztető üzenetet küld.

20. ábra Új termék felvitele

1.10.3. Termék módosítása és törlése

A termékek sorában 3 műveletet végezhetünk el: a törlést, a módosítást, és a részleteket tekinthetjük meg. Ha a törlésre kattintunk, az adott termék azonnal törlésre kerül az adatbázisból. A módosítás során, az adott termék minden adata beolvasásra kerül az adatbázisból, így nem kell újra megadni azokat, elég csak a szükséges módosításokat elvégezni, melyek a mentés után azonnal felülírásra kerülnek az adatbázisban. A részleteknél ugyanúgy táblázatos formában jelenítjük meg a termék összes adatát, mint ahogyan a felhasználó látja egy adott termékkel kapcsolatosan.

1.10.4. Kedvelt termékek

Visszatérve az admin főoldalára, ha a népszerű termékek gombra kattintunk, táblázatos formában megjelennek azok a termékek, amelyekre az ügyfelek szavaztak, vagyis kattintottak a szív ikonra. Első oszlopban a termék fotója jelenik meg, a második oszlopban a termék neve, a következőben pedig, hogy az adott termékből hány darab van raktáron, a negyedikben a termék nettó ára található forintban, és az utolsóban található meg, hogy az adott terméket hányan kedvelték. Ha egy raktáron lévő termék elfogyott,

pirosra válnak az adott terméknel az oszlopok, ezzel jelezve az adminisztrátornak, hogy az adott termékből nincsen készleten.

Kép	Termék neve	Raktáron	Ár	Kedvelések száma
	ARC A770 16GB DDR6	40	155750	2

21. ábra Kedvelt termékek

1.10.5. Megrendelések megtekintése

Ezen az oldalon a vásárlók eddig megrendelt termékeit tudjuk megtekinteni. A megrendelők nevei legördülő listában jelennek meg, így ki tudjuk könnyedén választani, kinek a megrendeléseit szeretnénk megnézni.

Megrendelések kezelése						
<div> <div>Latiza (Zita Ruzsinszki)</div> <div>▼</div> </div>						
Sorszám	Termék neve	Mennyiség	Statusz	Címzett	Összesen	Számlázás
1	TUF GAMING B660M-PLUS D4	1 darab	Kész	2015 Szigetmonostor	58 740 Ft	Számlázva
2	Liquid Freezer 280	2 darab	Kész	2015 Szigetmonostor	93 180 Ft	Számlázva

22. ábra Megrendelések megtekintése

1.10.6. Felhasználók kezelése

Ezen az oldalon megjelenítésre kerül az összes regisztrált felhasználó, ahol látható a jogosultsági szintje, amely vásárló esetén a user, admin esetén az admin felirat jelenik meg a felhasználó neve mellett. A művelet alatt lévő gombokkal lehetőségünk van kitiltani a felhasználókat, illetve újra engedélyezni.

Felhasználók kezelése		
Felhasználó neve	Jogosultsági szint	Művelet
DrSeres	admin	<button>Kitiltás</button>
user	user	<button>Engedélyezés</button>
teszt	user	<button>Kitiltás</button>

23. ábra Felhasználók kezelése

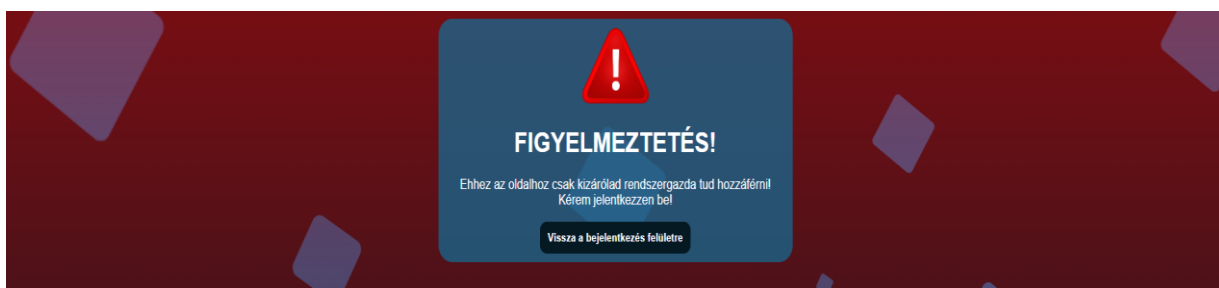
Azok a felhasználók, akiket kitiltunk, a következő alkalommal már nem tudnak bejelentkezni, hanem a kitiltott felhasználó oldalra kerülnek, ahol egy figyelmeztető üzenetet kapnak.



24. ábra Kitiltott felhasználó

1.11. Illetéktelen belépés az admin oldalra

Ha egy látogató megpróbálna belépni az admin felületre, a képen is látható üzenet fogadja villogó háttérképpel.



25. ábra Jogosulatlan belépési próbálkozás figyelmeztető üzenet

2. Fejlesztői dokumentáció

Dolgozatunk második fejezetében bemutatjuk a felhasznált technológiákat, majd a programozási munkánk folyamatát, hogy milyen módszerekkel oldottuk meg a különböző feladatokat, milyen akadályokba ütköztünk, és hogyan oldottuk meg ezeket.

2.1. Felhasznált technológiák

A tanévben mind a frontend, mind a backend programozást tanultuk, mint HTML5, CSS3, JavaScript, jQuery, Bootstrap, Angular.js, PHP, és PHP Cake programozási nyelveket, keretrendszereket. Mivel a záró projektünk tervezését már decemberben, a programozását pedig már januárban elkezdtük, azért választottuk a natív PHP programozási módszert, mivel ez alatt a rövid idő alatt ezen a területen tudtunk magabiztos tudást szerezni.

2.1.1. Visual Studio Code

A Visual Studio Code az egy olyan fejlesztői környezet, amely a számos hozzáadható moduljának köszönhetően rendkívül megkönnyíti és meggyorsítja a programozás folyamatát. Ilyen pl. az Emmet, amely kiválóan kezeli a rövidítéseket, ha pl. egy `<div>` konténert szeretnénk beszúrni, elég az osztály nevét megadunk pl. „`container>.box*2`” ezzel bekerül a teljes HTML kódja, és a két gyermek eleme, `box` osztálynévvel ellátva. Az „autoclose tag”, automatikusan kiírja a záró tag-eket is, ha a kezdő tag-et elkezdjük átírni, de számos egyéb olyan modul is hozzáadható, amely nemcsak a HTML, hanem a JavaScript és PHP programozást is segíti, pl. azonnal jelöli az esetleges hibákat.

2.1.2. HTML5

A HTML (vagyis HyperText Markup Language) az egy általános jelölőnyelv a weboldal elkészítéséhez. Két fő része a head és a body. A head részben adhatjuk meg a különböző meta tag-eket és az oldal címét. A meta tag-ek között megadjuk a kulcsszavakat is, amelyek alapján a kereső motorok könnyebben megtalálják az oldalunkat. Itt csatoltuk a stílusformázásokat tartalmazó stíluslapokat is a dokumentumhoz. A body az a terület, amelyet a böngésző megjelenít. A HTML5 már biztosítja a szemantikus elemek használatát is, mint pl. a header, main vagy footer, amelyeket mi is felhasználtunk a weboldalunk felépítéséhez. Ezek olyan elemek, amelyek a dokumentum különböző részleteit jelentéssel ruházzák fel, amelyeket a kereső motorok

értelmezni tudnak, tisztább felépítést biztosítanak, ezzel is egy lépést téve a kereső optimalizálás felé.

2.1.3. CSS3

A CSS (Cascading Style Sheets) azaz lépcsőzetes stíluslap Ennek segítségével határoztuk meg a weboldalunk elrendezését és az elemek megjelenését. Itt adhatunk meg háttérképet vagy háttérszínt, a szöveg színét és betűtípusát. Használatának előnyei, hogy a különböző HTML elemeket csoportosítva tudjuk formázni, ennél fogva később könnyebb lesz az oldal karbantartása, és módosítása. Nem kell a HTML tag-ek között keresni a formázási tulajdonságokat és értékeket. Használtunk media query-eket, amelyekkel valóban reszponzívvá tudtuk tenni a weboldalunkat, tehát mobil nézetben is kényelmesen olvashatóan jelenik meg minden. Szintén fontos szempont az is, hogy a css eltárolásra kerül a cash-ben, és az újabb megnyitáskor már sokkal gyorsabban töltődik be az oldal.

2.1.4. JavaScript

A JavaScripttel tehetjük igazán látványossá és dinamikussá az oldalunkat. Számításokat végezhetünk, űrlapokat ellenőrizhetünk, újabb elemeket hozhatunk létre a weboldalunkon a segítségével. Dolgozatunk nagyon fontos eleme, hiszen a termékek kosárba helyezését is JavaScripttel oldottuk meg.

2.1.5. AJAX

Az AJAX (Asynchronous JavaScript and XML) egy webfejlesztési technika, amely interaktív webalkalmazások létrehozására szolgál. Az AJAX lehetővé teszi, hogy a weboldal újra betöltése nélkül kérjünk le adatokat egy szerverről és megjelenítsük azokat a böngészőben. Ez növeli a honlap interaktivitását, és a felhasználói élményt, mivel nem kell várakozni a felhasználónak arra, hogy betöltődjének a lekért adatok.

2.1.6. PHP

A PHP szerveroldali programozási nyelv, széles körben használják. A kódot a webszerver PHP-feldolgozó modulja értelmezi, ezzel dinamikus weboldalakat hozva létre. A tanulást a procedurális PHP programozással kezdtük, mivel ez a módszer állt közelebb az eddig tanult technikákhoz, így szinte azonnal használatba tudtuk venni. Az objektum-orientált PHP programozás kb. 10 éve kezdett elterjedni. Ezt később kezdtük el tanulni, így ezt a módszert is alkalmaztuk helyenként a dolgozatunkban.

2.1.7. XAMPP

Egy nyílt forráskódú, platform-független webservert-szoftvercsomag, amelynek legfőbb alkotóelemei az Apache webservert és MariaDB, illetve Php-t is tudjuk kezelni. Ezzel az alkalmazással az adatbázisunkat kezeljük. Ebben tudunk létrehozni táblákat, ebben tudunk beolvasni és feltölteni adatokat a tábláinkba a phpMyAdmin felület elérésével. Az adatokat adatbázisban tároljuk, és azokat futtatjuk az oldalunkon a PHP programozási nyelv segítségével.

2.1.8. GitHub

A GitHub bemutatását a Git-el kell kezdenünk. A Git egy nyílt forráskódú verziókezelő rendszer, amelyet Linus Torvalds kezdett el fejleszteni - ugyanaz a személy, aki létrehozta a Linuxot. Amikor a fejlesztők létrehoznak egy alkalmazást, akkor folyamatosan változtatnak a kódon. A verziókezelő rendszerek ezeket a módosításokat egy központi adattárban tárolják. Ez lehetővé teszi a fejlesztők számára az egyszerű együttműködést, mivel letölthetik a szoftver új verzióját, módosíthatják és feltölthetik a legújabb verziót. Minden fejlesztő láthatja ezeket az új változásokat, letöltheti és közreműködhet. A GitHub pedig az a felület, ahol ezeket a kódokat tárolhatják a fejlesztők és megoszthatják egymással. Mi is ide töltöttük fel a projektünket.

2.1.9. Composer

A Composer egy alkalmazásszintű függőségkezelő a PHP programozási nyelvhez, amely szabványos formátumot biztosít a PHP szoftverek és a szükséges könyvtárak függőségének kezelésére. Nils Adermann és Jordi Boggiano fejlesztette ki, akik továbbra is irányítják a projektet. A fejlesztést 2011. áprilisában kezdték el, és először 2012. március 1-jén adták ki.

A Composer a parancssorból fut, és függőségeket (pl. könyvtárakat) telepít egy alkalmazáshoz. Automatikus betöltési lehetőségeket is biztosít azon könyvtárak számára, amelyek automatikus betöltési információkat határoznak meg a harmadik féltől származó kódok használatának megkönnyítése érdekében. A pdf formátumú számlageneráláshoz volt rá szükségünk.

2.1.10. CKEditor

A CKEditor egy WYSIWYG azaz „azt kapod, amit láatsz” szerkesztő. A szövegbeviteli mezőkhöz illesztettük hozzá abból a célból, hogy ugyanúgy formázhassa

a dokumentumot olyan felhasználó is, akinek nincsenek komolyabb webszerkesztési ismeretei, továbbá ennek a programnak a segítségével már előre formázott szövegeket is be tudunk illeszteni a weboldalunkra.

2.1.11. Dompdf

A Dompdf egy HTML-ből PDF-be konvertáló program. Lényegében a dompdf (többnyire) egy CSS 2.1- kompatibilis HTML-elrendezés és renderelő motor, amelyet PHP-ben írtak. Ez egy stílus-vezérelt renderer: letölti és beolvassa a külső stíluslapokat, a soron belüli stíluscímkéket és az egyes HTML-elemek stílusattribútumait. Támogatja a legtöbb prezentációs HTML attribútumot is.

2.1.12. WinSCP

A WinSCP (Windows Secure Copy) egy népszerű, ingyenes SFTP és FTP kliens Windowshoz, egy erőteljes többfunkciós eszköz. WinSCP segítségével helyi és távoli számítógépek között fájlokat másolhatunk több protokoll használatával. Arra használtuk fel, hogy minden egyes változás automatikusan mentésre kerüljön a Nethelyre is, ahová feltöltöttük a weboldalunkat.

2.2. A weboldal programozása

Munkánk során 2 db HTML fájlt hoztunk létre, amelyek az e-mail küldésért felelősek, 47 db PHP fájlt, és 31 stíluslapot készítettünk el. Éppen ezért csak azokat a kódrészleteket mutatjuk be a következőkben, amelyekre büszkék vagyunk, hogy meg tudtuk valósítani.

2.2.1. Felhasználók kitiltása

Mivel a felhasználók írhatnak nekünk üzenetet, arra gondoltunk, hasznos lehet a kitiltás is, arra az esetre, ha valaki sértő tartalmú szöveget küldene. Első lépésként a felhasználóhoz az adatbázisban hozzáadtunk egy engedélyezés oszlopot, amelynek az értéke 0 vagy 1 lehet. Ha az érték 0, akkor a felhasználó kitiltásra kerül.

Az 1.10.6.-os pontban bemutattuk, hogyan tilthatjuk ki a felhasználót, és onnantól kezdve nem tud belépni. Ezt a szerver oldalon úgy valósítottuk meg, hogy először egy autentikációs vizsgálatot indítottunk el, vagyis ha valaki megpróbál belépni az oldalra, megvizsgáljuk a jogosultsági szintjét, hogy user vagy admin, és ha user, akkor egy adatbázis-lekérdezést hajtunk végre, a „users” táblára vonatkozóan, ahol megvizsgáljuk

az engedélyezés értékét, és ha az egyenlő 0-val, akkor átirányításra kerül a kitiltásról tájékoztató oldalra.

```
if (isset($_SESSION['user_type'])) {  
    if ($_SESSION['user_type'] == 'user') {  
        $kitiltas = "SELECT * FROM users WHERE engedelyezes = 0 AND name =  
'".$_SESSION['name']."'";  
        $eredmeny = mysqli_query($dbconnect, $kitiltas);  
        if (mysqli_num_rows($eredmeny) > 0) {  
            header("location: kitiltottOldal.php");}}}
```

2.2.2. A népszerű termékek megjelenítése a főoldalon

Ebben az esetben is egy adatbázis-lekérdezést hajtottunk végre, ahol két táblából kerülnek lekérdezésre az adatok. Először a „kedvenctermekek” táblából kérdezzük le, hányszor szerepel a termék neve, és ebből kerül kiszámításra a Count() függvény segítségével, mennyi kedvelést kapott. Beállítottunk egy limitet is, hogy csak az első 15 darab legtöbb kedvelést kapott terméket mentjük el az \$sql változóba, ezeket csökkenő sorrendben kérdezzük le. Beolvassuk hozzá a termék adatait is a „termek” táblából, majd ezután jelenítjük meg őket a böngészőben a főoldalon.

```
$sql = "SELECT termek.id, termek.termekNev, termek.foto, termek.darab,  
termek.ar, COUNT(termek.termekNev) AS 'kedveles' FROM `kedvenctermekek`  
INNER JOIN termek ON kedvenctermekek.termekId=termek.id GROUP BY  
termek.termekNev ORDER BY kedveles DESC LIMIT 15;";  
$eredmeny = mysqli_query($dbconnect, $sql);
```

Kezeltük azt az eshetőséget is, ha nem lennének kedvelt termékek. A következő feltétel akkor valósul meg, amikor az \$eredmeny kevesebb, mint 1, vagyis egy terméket sem kedveltek még, akkor a program kiírja, hogy „Nincs kedvelt termék”. Ezt a sztringet a \$kimenet változóban tároltuk el, HTML tag-ek közé összefűzve.

```
if ((mysqli_num_rows($eredmeny)) < 1) {  
    $kimenet = "<article>  
    <h2>Nincs kedvelt termék</h2>  
    </article>"; }  
else {
```

Az else ág akkor fut le, ha vannak kedvelt termékek.

```
$kimenet = "";
```

```

while ($sor = mysqli_fetch_assoc($eredmeny)) {
    $kedvenc = $sor['kedveles'];
    if ($kedvenc >= 1) {
        $kimenet .=
            <<<URLAP
            <div class=" kedvenc">
            <div>
                <a href=adat.php?id={$sor['id']} ">
                <h2 style="word-wrap: normal;">{$sor['termekNev']}</h2>
                </a>
            </div>
            URLAP;}}}

```

Az eredményeken végigiterálva a `mysqli_fetch_assoc()` segítségével le tudjuk kérni, hogy hányan kedvelték összesen a termékeket.

Szelekcióval megnézzük, amennyiben egy terméket egynél többször kedveltek, akkor létrehozunk egy `$kimenet` változót, amibe HTML elemek segítségével felépítjük a fő oldalon látható mozgatható kártyákat. Ahogyan a fenti kódban is látható, a formázást meghatározó HTML tag-ek közé fűzzük össze a megjelenítendő elemeket. A `div` elemhez hozzárendeltünk a „kedvenc” osztályt, amely meghatározza a kártyák megjelenését a weboldalon. Anchor tag-en belül olvassuk ki a termékek adatait az adatbázisból sorról sorra. Így az adott termékre kattintva elérhetővé válnak a termékkel kapcsolatos részletek is.

```

<div class="kedvencScroll">
    <h1 class="kedvencH1">Legkedveltebb termékek</h1>
    <?php print_r($kimenet) ?>
</div>

```

A „kedvencScroll” osztály kiszelektálásával adtuk meg az elem azon tulajdonságait és értékeit, amelyek meghatározzák az elhelyezkedésüket a weboldalon. Ezen belül kerülnek beolvasásra az adatbázisból beolvasott adatok, amelyeket előzetesen a `$kimenet` változóba összefűztünk.

2.2.3. Belépés

A bejelentkezés oldalon először is egy kapcsolót kellett létrehoznunk, amelynek segítségével válthatunk a regisztrációs és belépés űrlap között. Ehhez konstans változókat

deklaráltunk, amelyekbe eltároltuk az űrlapokat osztály név, illetve a kapcsoló gombokat id alapján.

```
const loginsec=document.querySelector('.login-section')
const loginlink=document.querySelector('.login-link')
const registerlink=document.querySelector('.register-link')
const reggomb=document.getElementById('re')
const logingomb=document.getElementById('be')
const errorMessageDivInnerHTML = document.getElementById("error-message");
```

A következőkben egy esemény-figyelőt adtunk hozzá a register-link változóba eltárolt elemre, amelyre ha kattintás történik, akkor lefut egy ún. nyíl függvény (arrow function), melynek scope-jában meghatároztuk, mi történjen a kattintás esemény hatására. A hiba üzenetet tartalmazó elem üres értéket kap, az űrlapokat befoglaló elemhez hozzárendelésre kerül az „active” osztály, amelynek hatására a regisztrációs űrlap úszik be jobbról.

```
registerlink.addEventListener('click',()=>{
  errorMessageDivInnerHTML.innerHTML = "";
  loginsec.classList.add('active')
})
```

Ha a belépésre történik a kattintás, az esemény-figyelő elindítja az „arrow function-t”, amelyben a befoglaló div-ről eltávolításra kerül az active osztály, melynek hatására a belépés űrlap jelenik meg a regisztrációs űrlap helyett.

```
loginlink.addEventListener('click',()=>{
  errorMessageDivInnerHTML.innerHTML = "";
  loginsec.classList.remove('active')
})
```

Létrehozunk egy függvényt, hogy refaktoriálni tudjuk a kódunkat, ami azt jelenti, hogy többször is fel tudjuk használni. A functionnek beszédes nevet adtunk fetchError néven.

```
function fetchError(c, formId) {
  const formResult = [];
  if(formId == 'reg-form' ) {
    formResult.push(document.getElementById('name').value);
    formResult.push(document.getElementById('reg-email').value);
    formResult.push(document.getElementById('reg-password').value);
```

```

        formResult.push(document.getElementById('passwordAgain').value);
    } else if (formId == 'login-form') {
        formResult.push(document.getElementById('login-email').value);
        formResult.push(document.getElementById('login-password').value);    }
    fetch('../Webshop/api.php', {
        method: 'post',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
            'c': c,
            'formData': formResult, })
    })
    .then((response) => response.json())
    .then((data) => {
        if(data.message == "") {
            document.getElementById(formId).submit();
        } else {
            document.getElementById("error-message").innerHTML = data.message;
        }
    })
    .catch((error) => console.log(error));}

```

A függvény paraméterének megadtuk a „c” változót, amely az api.php-ben kerül eldöntésre, hogy melyik űrlap validálását indítsa el. A második paraméterként a formId-t adtuk meg, amely a két űrlapot különbözteti meg egymástól id-k kiolvasásával.

Ezt követően létrehoztunk egy formResult[] tömböt, melyet a szögletes zárójel jelez. Ebbe a tömbbe tudjuk eltárolni a beírt adatokat, amit a felhasználó tölt ki. Azt, hogy a felhasználó a regisztrációs, vagy a belépési űrlapot tölti-e ki, a formId segítségével különböztetjük meg. A tömb feltöltését a push() functionnel oldjuk meg, egy-egy input mező azonosítójának kiolvasásával. Az azonosítók sikeres kiolvasása után a bevitt adatokat fetch-eljük, azaz kinyerjük. A fetch-nek paraméterként meg kell adnunk, hogy hol tartózkodik az általunk használni kívánt fájl, ami jelen esetben php kiterjesztést kapott. A fetch-en belül megadjuk, hogy milyen metódust szeretnénk használni, amelynek segítségével kiolvassuk az adatokat. A következő a headers, amelyben átállítjuk a szöveges fájl típusát JSON-re.

A body a fetch legfontosabb eleme, hiszen itt tudjuk átadni a bevitt adatokat, amelyeket `JSON.stringify()` segítségével a szöveget JSON formátumra alakítja át. A `.then(response)` segítségével a fájlt JSON fájlá konvertáljuk.

Ezután a megkapott adatokat vizsgáljuk meg kétágú elágazás használatával. A szelekció első ága azt vizsgálja, hogy ha nincs hiba, akkor elküldi a szerver felé az adatokat, hogy azok beszúrásra kerüljenek az adatbázisunk megfelelő tábláiba.

Ha a vezérlés az else ágra fut, akkor az űrlap fölött kiírja a hibákat, milyen adatokat mulasztott el megadni, vagy adott meg nem megfelelő formátumban a felhasználó a bejelentkezés vagy regisztráció során.

```
if($data['c'] == 'regisztracioValidalas') {  
    $name = mysqli_real_escape_string($dbconnect, $data['formData']['0']);  
    $email = mysqli_real_escape_string($dbconnect, $data['formData']['1']);  
    $pass = sha1($data['formData']['2']);  
    $cpass = sha1($data['formData']['3']);  
    $select = " SELECT * FROM users WHERE email = '$email' && password =  
'$pass' ";  
    $result = mysqli_query($dbconnect, $select);  
    if (mysqli_num_rows($result) > 0) {  
        $message = 'Létezik már egy olyan felhasználó!';  
    } else if ($pass != $cpass) {  
        $message = "Nem egyezik meg a két jelszó!";  
    }  
    $result = array(  
        'message' => $message,  
    );  
    echo json_encode($result);  
}
```

A fetch átvisz az `api.php`-re ahol a felhasználó által beírt adatokat ellenőrizzük, tároljuk és megjelenítjük. A json fájlt visszafejtjük a `json_decode()` segítségével, hogy el tudjuk olvasni a tartalmát. A szelekcióval megvizsgáljuk, hogy melyik gombra kattintottuk a refaktorált függvény paramétereinek segítségével. Mindkét elágazásban el lehet olvasni a függvényben eltárolt tömbből az adatokat és azokat átadjuk egy új változónak, és így össze tudjuk hasonlítani a kapott adatokkal. Az adatok közül, amelyiknél hiba merül fel, azokat eltároljuk egy tömbbe. Ezt a tömbből adjuk át az

űrlapnak, ahol kiíródik a felmerült hiba vagy hibák. Ezt az eljárást használjuk mindkét űrlap esetében.

2.2.4. Termékek felvitele az adatbázisba

Azt szerettük volna elérni, ha már akár egy Excelben megformáztunk egy táblázatot, akkor azt egy az egyben, a formátumot megőrizve (azaz Ctrl+C és Ctrl+V segítségével) fel tudjunk vinni adatokat. A redactor erre nem volt alkalmas, amellyel az órán megismerkedtünk, mert azzal csak szöveget tudtunk formázni. Az interneten keresgeltünk, és így találtunk rá a CKeditor-ra, melynek kapcsán azt olvastuk, hogy ennél a szerkesztőnél van erre lehetőség, amit mi szeretnénk elérni. Ki is próbáltuk a hivatalos oldalán a demó verzióban, hogy hogyan működik a táblázat beillesztése. Letöltöttük a hivatalos oldaláról a standard verziót, amelyben benne volt az az ikon, amellyel az egyszerű formázott szöveges nézetet át tudjuk váltani nyerskódra, amely tartalmazza a HTML tag-eket is. Nem sikerült azonnal hozzáilleszteni a forráskód ikont, az oldalunk működéséhez egy YouTube videó segítségével sikerült megvalósítani a megfelelő működést. A CKeditor hasonlóan működik, mint a MS Word, lehet benne szöveget szerkeszteni, és ami fontos volt a project munkához, hogy tudunk benne táblázatot beilleszteni, ami a formázásokat is megőrzi.

2.2.5. Termékek hozzáadása a kosárhoz, és mentése localStorage-be

A webshopok esetében, ha meglátogatunk egy weboldalt, akkor előfordul, hogy nem hozunk döntést azonnal, hanem később visszatérünk az oldalra. Ilyen esetre véleményünk szerint nagyon fontos az a funkció, hogy a böngésző a lokális tárolóba mentse el azokat a termékeket, amelyek a kosárba kerültek. Ha nem kerülnek oda, és újra egy üres kosárral találkozik a felhasználó, akkor lehet, megfedezik az adott termékről, ha nem volt létszükséglet számára. Ez az üzlet tulajdonosának nem jó, mert elesik a megrendeléstől és a bevételtől. Viszont, ha a felhasználó, az oldalra visszatérve, a kosárban újra találkozik az előzetesen oda elhelyezett termékekkel, akkor nagyobb valószínűséggel folytatja a vásárlást.

Így a programunkba beleépítettük ezt a lehetőséget is, hogy a kosár gomb megnyomására, a termékek a local Storage-be is mentésre kerüljenek.

Tehát amikor a felhasználó egy terméket a kosárba helyez, az a local Storage-be is mentésre kerül. A local Storage olyan tulajdonság, amely lehetővé teszi, hogy a JavaScript-webhelyek és alkalmazások kulcs-érték párokat menthessenek el a böngésző

lokális tárolójába lejáratí dátum nélkül. Ez azt jelenti, hogy a böngészőben tárolt adatok a böngészőablak bezárása után is megmaradnak a böngésző memóriájában. A mentési műveletsort egy olyan függvényben helyeztük el a JavaScript fájlban, amely akkor kerül meghívásra, amikor az oldal betöltődik.

```
window.onload = function() {
```

A következő lépés a különböző elemek összegyűjtése és változókba való eltárolása volt. Erre a `const` azaz konstans változót használtuk a modern JavaScript szabályainak megfelelően. Így, ahogyan a kódunkból is látható a régi `var` azaz variable kulcsszót már egyáltalán nem használtuk, helyette csak a `const` és `let` kulcsszavakat, amelyek a változók deklarálására szolgálnak.

Elsőként eltároltuk egy konstansban a kosár ikonját, amely a weboldal jobb felső sarkában található:

```
const kosarikon = document.querySelector('.kosarikon');
```

Ellenőriztük, hogy valóban sikerült-e azt az elemet eltárolni, amelyet szerettünk volna. Egy összetett hosszabb kód esetében fontosnak tartjuk a lépésenkénti ellenőrzést, hogy ne a programozás végén derüljön ki, hogy esetleg valamit elírtunk, mert akkor sokkal nehezebb lesz kizárni a hibát. Ezt minden lépésnél megtettük:

```
console.log(kosarikon);
```

Következő a kosár tartalmának bezárása ikon tárolása a felugró ablakban:

```
const cartCloseBtn = document.querySelector('.fa-close');
```

Kosár tartalom, felugró ablak:

```
const cartBox = document.querySelector('.cartBox');
```

A felugró ablak megjelenítése az `active` osztály hozzáadásával történt meg, amelyet a `css`-ben előzőleg meghatároztunk. Tehát, ha a felhasználó rákattint az elemre, akkor megkapja az `active` osztályt az elem és az abban meghatározott tulajdonságokat és értékeket, ez által megjelenik a weboldalon felugró ablakban.

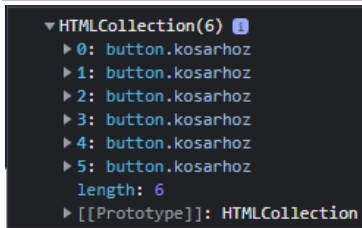
```
kosarikon.addEventListener("click", function() {  
    cartBox.classList.add('active');  
});
```

A felugró ablak bezárására egy `X`-et helyeztünk el a felső sarokban, így a felugró ablak bezárását az `active` osztály eltávolításával oldottuk meg.

```
cartCloseBtn.addEventListener("click", function() {  
    cartBox.classList.remove('active');  
});
```


2.2.6. Adatok hozzáadása a kosárhoz

```
const kosarhozgomb = document.getElementsByClassName('kosarhoz');
```



26. ábra Javascript

ellenőrzés, sikeresen eltároltuk-e az adatokat

Magukat a gombokat eltároltuk egy konstans változóba, majd ellenőriztük, valóban kezünkbe kerül-e az adat:

```
console.log(kosarhozgomb);
```

Amint a konzolon látható, a gombokat egy tömbben tárolja el, amin egy for ciklussal iterálunk végig, hogy megtudjuk, melyik elemre történt a kattintás. A kosárba kerülő termékeknek létrehoztunk egy üres tömböt.

```
let termekek = [];
```

Majd ezt követően for ciklussal megvizsgáltuk, hogy melyik elemre került a kattintás.

```
for (let i = 0; i < kosarhozgomb.length; i++) {
```

A gombokhoz hozzáadtunk egy esemény-figyelőt, amely két paramétert vár, az első a figyelendő esemény, amely jelen esetben egy klikk, vagyis kattintás a gombra, a második pedig egy függvényhívás.

```
kosarhozgomb[i].addEventListener("click", function(e) {
```

A függvény scope-jában létrehoztunk egy elágazást, hogy amennyiben a Storage nem üres, akkor egy termék nevű változóba olvassa be az adatokat. A termék nevét úgy kapjuk meg, hogy mivel a gomboknak van egy div itemInfo osztályú szülő eleme, azon belül pedig a termékek adatai, így azokra hivatkozunk, mint gyermek elemek, melyeket index alapon válogatunk le az alábbi módon:

```
if (typeof(Storage) !== 'undefined') {  
    let termek = {  
        id: i + 1,  
        name: e.target.parentElement.children[0].innerHTML,  
        price: parseInt(e.target.parentElement.children[1].innerHTML),  
        no: parseInt(e.target.parentElement.children[2].value)  
    };  
}
```

A későbbiek során a tervezés miatt úgy döntöttünk, kiegészítjük a div konténert egy újabb elemmel egy <hr> horizontal rule vonallal. Viszont utána a kosárba nem olvasta be a program az elemeket, így elkezdtünk vizsgálni, hol lehet a hiba. Rájöttünk, hogy az

újonnan hozzáadott vonalat is természetesen gyermek elemnek számolja a böngésző, így a darabszám már nem 2-es indexű lett, hanem 3-as. Ezt kijavítottuk, és így már újra megfelelően működött a kosárba helyezés gomb.

Először is egy elágazás feltételeként megvizsgáljuk, van-e már a böngésző lokális tárolójába mentett adat, és ha nincs, vagyis az értéke egyenlő nullával, akkor a push metódussal hozzáadjuk a „termékek” nevű tömbhöz.

```
if (JSON.parse(localStorage.getItem('termékek')) === null) {  
    termékek.push(termek);
```

Ezt követően elmentjük a termékeket a setItem-el, mivel ez lehetővé teszi az értékek tárolását a localStorage objektumban. A mentéshez két paraméterre van szükségünk, egy kulcsra és egy értékre. A kulcsra később lehet hivatkozni a hozzá csatolt érték beolvasásához.

```
    localStorage.setItem("termékek", JSON.stringify(termékek));  
    window.location.reload();  
} else {
```

Az else ágban azt az esetet programoztuk le, amikor a localStorage nem üres. A getItem metódus visszaadja a localStorage-be eltárolt adatokat. Ezt a map függvény segítségével nyertük ki, majd újra rendereltük az oldalt, amely a böngésző lokális tárolójába mentett adatokat kirajzolja a kosár tartalmaként.

```
    const localtermékek =  
JSON.parse(localStorage.getItem("termékek"));  
    localtermékek.map(data => {  
        if (termek.id == data.id) {  
            termék.no += data.no;  
            termék.price += data.price;  
        } else {  
            termékek.push(data);  
        }  
    });  
    termékek.push(termek);  
    localStorage.setItem('termékek', JSON.stringify(termékek));  
    window.location.reload();}
```

Gondoltunk arra az esetre is, ha valami miatt, pl. a localStorage megtelik, és nem működne a felhasználó gépén, ezt a hibalehetőséget az else ágban kezeltük le.

```
} else {alert('A local storage nem működik a böngészőjében!'); }
```

2.2.7. A rendelés hozzáadása a kosárhoz

Először eltároltuk a kosár ikonját egy változóba, majd deklaráltunk egy „no” nevű változót, amelyet 0 értékkel inicializáltunk. Ennek a változónak az értéke kerül növelésre a rendelt áruk darabszámával.

```
const kosarikonP = document.querySelector('.kosarikon p');
let no = 0;
JSON.parse(localStorage.getItem('termek')).map(data => {
  no = no + data.no;
});
kosarikonP.innerHTML = no;
```

Adatok hozzáadása a táblázathoz:

```
const cardBoxTable = cartBox.querySelector('table');
```

Az üres tábla létrehozása majd a sorok hozzáfűzése:

```
let tableData = '';
```

A táblázat fejlécének összefűzése:

```
tableData += '<tr><th>Cikk szám</th><th>Termék  
neve</th><th>Darab</th><th>Ára</th><th></th></tr>';
if (JSON.parse(localStorage.getItem('termek'))[0] === null) {
```

Ha nincs hozzáadva termék, akkor is adjon hozzá egy üres sort:

```
tableData += '<tr><td colspan="6"></td></tr>'
} else {
  JSON.parse(localStorage.getItem('termek')).map(data => {
    tableData += '<tr><th>' + data.id + '</th><th>' + data.name + '</th><th>'
+ data.no + '</th><th id="prices">' + (data.price * data.no) + '</th><th><a href="#"  
onclick=Delete(this);>Törlés</a></th></tr>';
  });
}
```

A rendelések összeadása, mennyit kell fizetnie összesen a megrendelőnek:

```
let sum = 0;
JSON.parse(localStorage.getItem('termek')).map(data => {
  sum += data.no * data.price;
});
```

A kapott összeget ellenőriztük a konzolon, majd hozzáfűztük a megjelenő táblázathoz.

```
console.info(sum)

tableData += '<tr><th colspan="3"><a href="#"
onclick="megrendeles();">Megrendelés elküldése</a></th><th>'+sum+'</th><a
href="#" onclick=allDelete(this);>Összes törlés</a></th></tr>';

cardBoxTable.innerHTML = tableData;
}
```

2.2.8. Törlés a kosárból

Lehetőséget adtunk a felhasználónak, ha szeretné törölni a termékeket, akkor terméktípusonként is tud törölni, illetve az összes terméket is tudja törölni. A törlés egy termék esetében úgy működik, hogy tulajdonképpen a nem törölt termékek kerülnek visszatöltésre a böngésző lokális tárolójába. Tehát megvizsgáljuk, hogy ahol nem egyenlő az id a törlésre kijelölt termék id-jával azok kerülnek a termékek nevű tömbbe, amelyek ezután kiolvasásra kerülnek. Majd a Delete() függvény újra rendereli a kosár tartalmát amely a böngészőben megjelenik.

```
function Delete(elem) {
  let termek = [];
  JSON.parse(localStorage.getItem('termek')).map(data => {
    if (data.id !== elem.parentElement.parentElement.children[0].textContent) {
      termek.push(data);
    }
  });
  localStorage.setItem('termek', JSON.stringify(termek));
  window.location.reload();
};
```

Összes elem törlése esetében az összes elem „kiszűrésre” kerül a lokális tároló tömbjéből.

```
function allDelete() {
  let termek = [];
  JSON.parse(localStorage.getItem('termek')).map(data => {
    termek.splice(data);
  });
};
```

```
localStorage.setItem('termek', JSON.stringify(termek));  
window.location.reload();  
};
```

2.2.9. Számla kiállítása

A személyes adatok kitöltése után, amikor a felhasználó megrendelte a terméket, szeretnénk volna egy olyan számlát is kiállítani a részére, hogy értesüljön arról, mi volt a rendelése. Ennek a megoldására rengeteg YouTube videót néztünk meg, hogy hogyan lehet ezt megoldani. Hosszas kutatás után rátaláltunk egy egészen egyszerű megoldásra, amely segített bennünket az elindulásban. Először le kellett töltenünk a Github linken lévő dompdf működéséhez a composert, amelynek segítségével tudtunk számlát generálni, azaz a HTML kódot PDF formátumúvá nyomtatni. A composer letöltése után be kellett linkelni a Dompdf-et a működtetéshez.

```
$szamlaSorszamaSql = "SELECT sorszam FROM megrendeles ORDER BY  
sorszam DESC LIMIT 1";  
$szamlaSorszamQuery = mysqli_query($dbconnect, $szamlaSorszamaSql);  
$beolvasas = mysqli_fetch_assoc($szamlaSorszamQuery);  
$sorszam = $beolvasas['sorszam'];  
switch(strlen((string)$sorszam)) {  
    case 1: $szamHossz = "0000"; break;  
    case 2: $szamHossz = "000"; break;  
    case 3: $szamHossz = "00"; break;  
    case 4: $szamHossz = "0"; break;  
    default: $szamHossz = "";}
```

A számlán található sorszámot egy adatbázis-lekérdezés segítségével olvassuk ki. Létrehoztuk a \$szamlaSorszamaSql változót, amelybe lekérdezzük, hogy melyik a legnagyobb sorszám az adatbázisban. A lekérdezés végrehajtása után eltároljuk a sorszámot egy \$sorszam változóban, amelynek értékét a számlán megjelenítjük, mint a számla sorszáma.

A \$szamHossz változóban eltároljuk switch case esetekkel megvizsgált értéket. Ezt azért kell megvizsgálnunk, mert ha a szám karaktereinek száma növekszik, akkor csökkentenünk kell a 0-k számát a bizonylat sorszámanál.

Ezáltal biztosítjuk, hogy nem nekünk kell átírnunk a számla nulláit, hanem a rendszer automatikusan ezt megteszi helyettünk.

```
require_once 'vendor/autoload.php';
use Dompdf\Dompdf;

$dbconnect = new PDO('mysql:host=mysql.omega;dbname=webshopv3',
'webshopv3', 'Szuperbat01');
```

A megrendelő adatainak kikeresése:

```
$sql = "SELECT megrendeles.id AS 'megrendeles', gyarto.gyartoNev,
termek.termekNev, megrendeles.raktaron, termék.id AS 'termekid', termék.ar,
users.name, users.vezetekNev, users.keresztNev, users.kartyaszam,
users.kiszallitasiCim, users.varos, users.megye, users.email FROM termék
INNER JOIN megrendeles ON termék.id = megrendeles.termekId
INNER JOIN gyartokategoria ON
termék.gyartoKategoriaId=gyartokategoria.gyartoKategoriaId
INNER JOIN gyarto ON gyartokategoria.gyartoId=gyarto.gyartoId
INNER JOIN users ON users.id= megrendeles.usersId WHERE users.name =
'$felhasznalo' AND megrendeles.szamlazva = 0;";
```

A \$dbconnect változóban megadtunk utasításokat az adatbázis kapcsolásához a szolgáltató nevét és az adatbázis nevét, valamint a hozzátartozó jelszót is. Következően létrehoztunk egy SQL utasítást, amiben megadtuk a megrendelés azonosítóját, a termék gyártóját, a termék nevét, a felhasználó keresz- és vezetéknévét, személyes adatait. Az SQL utasítás végén a „where” feltételben kritériumként megadtuk, hogy csak azokat az adatokat gyűjtse össze a rendszer, amelyek az éppen belépett felhasználóhoz tartoznak és még nincsenek kiszámlázva.

```
$stmt = $dbconnect->prepare($sql);
$id = $dbconnect->lastInsertId();
$stmt->execute();
$sor = $stmt->fetch(PDO::FETCH_ASSOC);
```

Az sql utasítás végrehajtását objektum-orientáltan valósítottuk meg, amelynek hatására a program átláthatóbbá válik, ellentétben a strukturált megoldással. Előkészítjük a lekérdezést a \$stmt változóba, ezzel is biztosítjuk azt, hogy nem fordul elő lekérdezési hiba, ami megakasztja az oldalt. Az execute() functionnel pedig tudjuk futtatni a lekérdezést. A lekérdezés eredményét a \$sorok változóba beolvassuk a fetchAll() function segítségével, aminek a paraméterébe megadjuk, hogy a beolvasott adatokat hogyan készítse elő PDO formátumúvá. A \$gt változó, ami a végösszeg lesz, 0 értékkel inicializáltuk.

```

foreach ($sorok as $sor) {
    if(in_array($sor["id"],$_SESSION["deal"]));
    $akcio = number_format($sor['ar'] * 0.8, '0', '.', '');
    $ArFormazas = intval($sor['ar']);
    $akciosAr = (in_array($sor['termekid'], $_SESSION['deal'])) ? " { $akcio} " :
"$ArFormazas";
    $termekAkcio = (in_array($sor['termekid'], $_SESSION['deal'])) ? " 20 % " :
"0 %";
    $html .= '
        <tr>
            <td class="rendelesId">' . $sor['megrendeles'] . '</td>
            <td>' . $sor['gyartoNev'] . ' ' . $sor['termekNev'] . '</td>
            <td>' . $sor['raktaron'] . '</td>

            <td class="ar">' . number_format($akciosAr * $sor['raktaron'], 0, '.', '
') . ' Ft</td>

            <td class="ar">' . $termekAkcio . '</td>
        </tr>';
    $gt += number_format($akciosAr * $sor['raktaron'], 0, '.', '');
    $i++;
}
$html .= '
    </tbody>
    <tr>
        <th colspan="4" class="vegossz">Végösszeg</th>
        <td>' . number_format($gt, 0, '.', ' ') . ' Ft</td>
    </tr>
</table>
</body>

```

Elindítunk egy foreach ciklust, amiben végigiterálunk az előbb eltárolt lekérdezésen, az adatokat táblázat formában állítunk össze, amelyet a HTML-ben tudjuk meghívni a \$html változó segítségével. A táblázatnak megadjuk a sorait, ami a <tr> tag névre hallgat. A <tr> tagek között a termék fotóját, a gyártó nevét, a termék nevét, a felhasználó által rendelt mennyiséget, és az ahhoz tartozó árat rendeljük hozzá a <td>

tagekhez. A táblázat alsó részében a végösszeg jelenik meg, amely a felhasználó által megrendelt termék teljes árát tartalmazza.

```
$dompdf = new Dompdf;  
$dompdf->loadHtml($html);  
$dompdf->setPaper('A4', 'portrait');  
$dompdf->render();  
$dompdf->stream('PCszamla.pdf');
```

A \$html változóban lévő táblázatot fogjuk a felhasználónak kinyomtatni. Ehhez először létre kell hoznunk egy \$dompdf változót, amit példányosítunk. A változóba betöltjük az eddig elkészített táblázatot. A setPaper() functionben megadjuk, hogy milyen méretű a számla és milyen formában nyomtassa ki. Ezután a rendszer kirendereli, majd pedig kinyomtatja a megadott néven a felhasználó számára.

2.2.10. Kapcsolat

Először készítettünk egy új e-mail címet, amiről majd a FormSubmit oldalán tudunk regisztrálni.

Következő lépésként az oldalon található kódot beillesztjük az oldal kódjába.

```
<form action="https://formssubmit.co/your@email.com" method="POST">  
  <input type="text" name="name" required>  
  <input type="email" name="email" required>  
  <button type="submit">Send</button>  
</form>
```

Ezután a [your@email.com](https://formssubmit.co/your@email.com) helyére a most regisztrált e-mail címet írjuk be, ami később titkosított formában lesz eltárolva.

```
<form      action="https://formssubmit.co/c811a6521133788e51dc0c445890268d"  
method="POST">
```

A „c811a6521133788e51dc0c445890268d” titkosított kódot úgy kaptuk meg, hogy először be kellett regisztrálni a FormSubmit oldalra és ott a regisztrált e-mail címünkre küldenek egy e-mailt, ami tartalmazza az általunk megadott e-mail címet és annak titkosítását. Ha nem titkosítjuk az e-mail címünket, akkor egy tapasztalt felhasználó ki tudja olvasni a kód sorai közül.

```
<label for="name">Teljes Név*:</label>  
<input type="text" name="name" required placeholder="Kérem írja be a teljes  
nevét">
```


Elsőként kötelezően meg kell adni a felhasználónak a teljes nevét.

```
<label for="email">E-mail*:</label>
<input type="email" name="email" required placeholder="Kérem adja meg
Email címét">
```

Ezután a felhasználónak meg kell adni kötelezően az e-mail címét.

```
<input type="hidden" name="_next"
value="http://www.projectmunka.nhely.hu/Webshop/thanks.html">
```

Ezen input beviteli mezőt a felhasználó elől elrejtjük a hidden segítségével. Ebben a kódban meg kell adnunk az oldal domain címét, amelyről át fogja irányítani a felhasználót a „thanks.html” oldalra.

```
<label for="message">Ön által jelentett probléma*:</label>
<textarea name="message" placeholder="Probléma vagy kérdés írása..."
rows="10" minlength="15" style="resize: none;"></textarea>
```

Itt a felhasználónak meg kell adnia a tapasztalatát, visszajelzését az oldallal vagy egy rendeléssel kapcsolatban. A válasznak minimum 15 karakter hosszúnak kell lennie.

```
<input type="hidden" name="_captcha" value="true">
```

A kódrész segítségével engedélyezzük a captcha-t az oldalon.

```
<input type="hidden" name="_template" value="table">
<input type="hidden" name="_subject" value="PcWebshop visszajelzés">
<p style="padding-top: 14px; color:white;">A *-al jelölt mezők kitöltése
kötelező!</p>
<input type="text" name="_honey" style="display:none">
<div class="buttons" style="display: flex;">
<button type="submit" class="btnForm">Elküldés</button>
<a href="index.php"><button type="button"
class="btnForm2">Vissza</button></a>
</div>
</form>
```

A következő input az egy template-t, azaz egy előre megformázott táblázatot fog küldeni a titkosított e-mail címre, amelynek segítségével az adminok jobban el tudják különíteni ki, miért küldött e-mailt. A name=„subject” input mezőt arra használtuk, hogy az e-mail küldésnél a tárgy szövegének értékét a saját szövegünkre formázzuk át. A _honey input mező segítségével tudjuk kiszűrni a spam üzeneteket.

Ha rákattintunk az „Elküldés” gombra, akkor megjelenik egy Captcha által generált ellenőrző kód. Ha a felhasználó megfejti a képes kódot, akkor az általa jelentett hiba vagy észrevétel elküldésre kerül, amelyet az adminok megkapnak az e-mail címünkre.

2.2.11. Egyéb felhasznált eszközök

Ahhoz, hogy tudjunk egységesen és otthonról is együtt dolgozni, a kommunikáció létrehozására a Discordot használtuk. Először a Google Drive-ot használtuk a programunk egymás közötti megosztására, de későbbiekben áttértünk a GitHub használatára.

2.3. Admin oldal

Az admin oldal csak azoknak érhető el, akiknek van rendszergazda joguk ezáltal a felhasználók nem tudnak ide belépni.

2.3.1. Főoldal

A fő oldalon a keresés dinamikus, vagyis ahogy írjuk be az adott gyártót vagy termék nevét, rögtön már keres is anélkül, hogy lenne egy „nagyító” ikon vagy leütnénk az enter a keresésért.

```
<form method="post">
  <input type="text" name="search_text" id="search_text"
placeholder="Keresés">
</form>
```

A termékek keresését input mező segítségével oldottuk meg, amelynek adtunk egy azonosítót.

Ahhoz, hogy ez a keresés működjön, létre kellett hoznunk két fájlt, mégpedig a search.js-t és fetch.php-t. A fetch.php fájlban olvassuk ki az adatbázisból a szükséges adatokat és állítjuk össze a táblázatot, a másik fájlban pedig a dinamikus keresést oldjuk meg.

fetch.php

```
$output = "";
if (isset($_POST["query"])) {
    $search = mysqli_real_escape_string($dbconnect, $_POST["query"]);
```

Szelekciót indítunk, amellyel ellenőrizzük, hogy létezik-e a \$_POST['query'] lekérdezés.

A `mysqli_real_escape_string()` függvény kihagyja a speciális karaktereket egy karakterláncban az SQL-lekérdezésben való használatához, figyelembe véve a kapcsolat aktuális karakterkészletét.

```
$query = "SELECT * FROM termek INNER JOIN gyartokategoria ON
termek.gyartoKategoriaId=gyartokategoria.gyartoKategoriaId INNER JOIN gyarto ON
gyartokategoria.gyartoId=gyarto.gyartoId INNER JOIN kategoria ON
gyartokategoria.kategoriaID=kategoria.kategoriaID
WHERE termek.termekNev LIKE '% {$search} %'
OR gyarto.gyartoNev LIKE '% {$search} %'
OR kategoria.kategoriaNev LIKE '% {$search} %'";
```

\$query változóba ezt a lekérdezést eltároljuk. Ez a lekérdezés akkor valósul meg, ha bármit is beütünk a kereső mezőbe.

```
} else {
    $query = "SELECT * FROM termek INNER JOIN gyartokategoria ON
termek.gyartoKategoriaId=gyartokategoria.gyartoKategoriaId INNER JOIN gyarto ON
gyartokategoria.gyartoId=gyarto.gyartoId INNER JOIN kategoria ON
gyartokategoria.kategoriaID=kategoria.kategoriaID ORDER BY termek.id";
}
$result = mysqli_query($dbconnect, $query);
```

Ha viszont nem írtunk be semmit, akkor kiírja az összes terméket egy táblázatba.

```
if (mysqli_num_rows($result) > 0) {
    $skimenet = "<div class='tableContainer'>
<table class='kategoriaTable'>
<tr>
    <th>Kép</th>
    <th>Kategória</th>
    <th>Gyártó</th>
    <th>Termék</th>
    <th class='tableAr'>Ár</th>
    <th>Műveletek</th>
</tr>
";
```

Először megnézzük, hogy az előbb megírt lekérdezés ad-e vissza sort. A táblázatot itt állítjuk össze, ha igaz értékre fut a lekérdezés.

```

while ($sor = mysqli_fetch_array($result)) {
    $kimenet .= "<tr>
        <td><img src=\"../img/termekekuj/{ $sor['foto']}\" alt=\"pro\"></td>
        <td>{ $sor['kategoriaNev']}</td>
        <td>{ $sor['gyartoNev']}</td>
        <td>{ $sor['termekNev']}</td>
        <td>\" . number_format($sor['ar'], 0, ',', ' ') .\" Ft</td>

```

While ciklus használatával és a mysqli_fetch_array() segítségével beolvastatjuk a lekérdezés eredményét. A táblázat első oszlopa tartalmazza a termék fotóját, kategória nevét, a gyártó nevét, a termék nevét és végezetül a termék árát. Az árat number_format() segítségével formázzuk.

```

        <td class='muveletek'><a href=\"torles.php?id={ $sor['id']}\"
class=\"gomboks\">Törlés</a> <a href=\"modositas.php?id={ $sor['id']}\"
class=\"gomboks\" >Módosítás</a> <a href=\"reszletek.php?id={ $sor['id']}\"
class=\"gomboks\">Részletek</a></td>
    ";

```

Az utolsó oszlopban 3 gombot jelenítünk meg az admin számára. Az adminnál nem kell soronként törölnie az adatokat, elég csak az általa választott terméket. Az admin tud a törlésen kívül adatot módosítani és részleteket megtekinteni egy adott termékről.

```

$kimenet = "<table>
<tr>
    <th style='padding:10px'>Nem található az Ön által keresett termék</th>
</tr>";
echo $kimenet;

```

Az admin olyan terméket keres, ami nincs az adatbázisunkban akkor lekezeljük ezt amit láthatunk a fent megjelenített kódban is.

search.js

Ez a .js fájl működteti a dinamikus keresést.

```

$(document).ready(function () {
    load_data();
    function load_data(query) {
        $.ajax({
            url: "fetch.php",
            method: "POST",

```

```

        data: {
            query: query
        },
        success: function (data) {
            $('#result').html(data);});});

```

Ez egy jQuery kód, amely az oldal betöltődésekor meghívja a load_data függvényt. Ez a függvény egy AJAX kérést küld a “fetch.php” fájlra, ami az eredményt megjeleníti a #result elemen belül.

```

$('#search_text').keyup(function () {
    var search = $(this).val();
    if (search != "") {
        load_data(search);
    } else {
        load_data();});});
});

```

Ez a kód a #search_text elemen belüli billentyűzet eseményeket figyeli. Ha a felhasználó beír egy karaktert, vagy karakter láncot az input mezőbe, akkor a load_data függvény meghívásra kerül. Ha a felhasználó törli a szöveget, akkor a load_data függvény meghívódik anélkül, hogy bármilyen paramétert átadna neki. Ez azt jelenti, hogy az eredeti adatok újra betöltődnek.

2.3.2. Új gyártó felvitele

Az új gyártó felvitele menüpontban nem fogjuk részletezni az egész kódot, hanem csak azt fogjuk kiemelni, amire büszkék vagyunk, hogy meg tudtuk oldani. A megoldandó probléma az volt, hogy amikor fel akartunk vinni egy új gyártót és a hozzá tartozó termékkategóriát, mint például: AMD ami a processzorhoz tartozik, akkor két táblába is létre kellett hozni őket.

```

$sql = "INSERT INTO gyarto
        (gyartoId, gyartoNev)
        VALUES
        (',{ $gyartoNev}');"

$querys = mysqli_query($dbconnect, $sql);
$utolsoId = mysqli_insert_id($dbconnect);

```

```
$sqls = " INSERT INTO gyartokategoria (gyartoKategoriaId, kategoriaID, gyartoId)
VALUES (', '{$kategoriaID}', '{$utolsoId}')";
```

```
$eredmenyek = mysqli_query($dbconnect, $sqls);
```

Megoldásként létrehoztunk először egy lekérdezést, amivel felvisszük az általunk kiválasztott gyártót a „gyarto” táblába. A \$utolsoId változóba a mysqli_inserted_id() segítségével a már felvitt utolsó gyártónak az azonosítóját kiolvassuk az adatbázisból, ami azért fontos, mert a következő táblába, ami egy kapcsolótáblaként funkcionál (gyartokategoria) fel kellett vinnünk az utolsó azonosítót és a hozzá tartozó kategoriaID-t is, amellyel tudjuk, hogy a gyártó melyik termékkategóriába tartozik.

2.3.3. Új termék felvitele

A kódrészben rögtön található egy lekérdezés, amit \$sql változóban tárolunk. Kigyűjtjük az összes adatot a „gyartokategoria”-ból, utána két táblával is összekötjük vele, pontosabban a „gyarto”-t és a „kategoria” táblával INNER JOIN segítségével. Az eredményét \$eredmeny változóban tároljuk el.

```
$sql = "SELECT * FROM `gyartokategoria` INNER JOIN kategoria ON
gyartokategoria.kategoriaID=kategoria.kategoriaID INNER JOIN gyarto ON
gyartokategoria.gyartoId=gyarto.gyartoId";
```

```
$kiir = "";
while($sor = mysqli_fetch_assoc($eredmeny)){
    $kiir.= "
    <option value=\"{$sor['gyartoKategoriaId']}\">    {$sor['kategoriaNev']}
    ({$sor['gyartoNev']})</option>";
}
```

Amikor betöltjük az oldalt, egy legördülő lista jelenik meg, ahol kiválaszthatjuk, hogy milyen típusú terméket szeretnénk felvinni. A mysqli_fetch_assoc() segítségével beolvassuk a lekérdezés eredményeit, majd a \$kiir változóba eltároljuk a termékkategóriához tartozó gyártó nevét, ami zárójelek között jelenik meg a felhasználó számára. Mint például: processzor (AMD), processzor (INTEL), és így tovább. A kategóriát, a termék nevét, a termék leírását, a termék árát és a termék készletének számát eltároljuk egy-egy változóba. A következő változó a \$mine, amiben három fotó kiterjesztését tároljuk, pontosabban jpeg-t, gif-et és png-t. Ellenőrizzük, hogy a mezők

üres-e. Ha valamelyik mező üres, akkor hozzáad egy hibaüzenetet a \$hibak tömbhöz. A következő kettő szelekció ellenőrzi a feltöltött kép méretét és formátumát.

```
if ($_FILES['foto']['error'] == 0 && $_FILES['foto']['size'] > 2000000) {  
    $hibak[] = "A kép mérete nagyobb, mint 2 MB!";  
}  
if ($_FILES['foto']['error'] == 0 && !in_array($_FILES['foto']['type'], $mine)) {  
    $hibak[] = "A kép formátuma nem megfelelő!";  
}
```

Ha a kép mérete nagyobb, mint 2 MB, akkor hozzáad egy hibaüzenetet a \$hibak tömbhöz. Ha a kép formátuma nem szerepel a \$mine tömbben (azaz nem megfelelő), akkor szintén hozzáad egy hibaüzenetet. A következő kód a feltöltött képfájl típusát vizsgálja és a típusnak megfelelően állítja be a \$kit változó értékét.

```
switch ($_FILES['foto']['type']) {  
    case "image/png":  
        $kit = ".png";  
        break;  
    case "image/gif":  
        $kit = ".gif";  
        break;  
    default:  
        $kit = ".jpg";  
}
```

Ha a kép típusa image/png, akkor a \$kit értéke .png lesz. Ha a kép típusa image/gif, akkor a \$kit értéke .gif lesz. Minden más esetben a \$kit értéke .jpg lesz.

```
$sql = "INSERT INTO `termek`(`id`, `gyartoKategoriaId`, `termekNev`,  
`foto`, `leiras`, `darab`, `ar`)VALUES('{'$gyartoKategoriaId}','{'$termekNev}','{'$foto}','  
{ $leiras}','{'$darab}','{'$ar}')";  
mysqli_query($dbconnect, $sql);  
kép mozgatása a végleges helyére  
move_uploaded_file($_FILES['foto']['tmp_name'], "../img/termekekuj/{$foto}");  
header("location: kategoria.php");
```

A \$sql változóba tároljuk el az INSERT INTO lekérdezést az adatok felviteléhez. A move_upload_file() első paraméterébe kerül, az adatbázisból beolvasott fotók nevei, a második paraméterbe kerül az adott kép útvonala.

2.3.4. Népszerű termékek

Ezen az oldalon az eddig legnépszerűbb termékek találhatóak táblázat formában elrendezve. Létrehoztunk egy lekérdezést, amit egy \$sql változóban tároltunk.

```
SELECT      termék.termekNev,      termék.foto,      termék.darab,      termék.ar,
COUNT(termék.termekNev) AS 'kedveles' FROM `kedvenctermekek` INNER JOIN
termék ON kedvenctermekek.termekId=termék.id GROUP BY termék.termekNev;
```

Lekérjük a termékek nevét, a termékek fotóját, hogy a termékből mennyi van készleten, a termék árát, az egyes termékeket hányan kedvelték. A kedvelések számát COUNT()-tal megszámoaltatjuk, melynek alias nevet adunk. A lekérdezés végén termék neve szerint csoportosítunk növekvő sorrendben. A következőkben megvizsgáltuk a lekérdezést, hogy az előbbi utasítás ad-e vissza sort. Ha ad vissza sort, akkor kiírja táblázat formájában a kedvelt termékeket, amit egy \$kimenet változóban tárolunk el. Az adatok kiírását while ciklussal valósítjuk meg. Ellenkező esetben nem ír ki semmit, tehát a lekérdezés a hamis ágra fut.

2.3.5. Megrendelések megtekintése

A megrendelések megjelenítésének első lépéseként meg kellett néznünk, hogy az adminisztrátor a legördülő listából melyik elemet választja ki és annak az adatbázisban hol található az azonosítója, ez alapján különböztetjük meg a felhasználókat egymástól. A megoldást JavaScript segítségével oldottuk meg, amelyet függvénybe szerveztünk.

```
function showUser(str) {
    if (str == "") {
        document.getElementById("txtHint").innerHTML = "";
        return;
    }
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("GET", "userFetch.php?q=" + str, true);
    xmlhttp.send();}
```

A függvényben először ellenőrizzük, hogy a legördülő listában kiválasztottunk-e egy személyt, amely ha nincsen kiválasztva, akkor visszaad egy üres szöveget. A következő lépésben létrehozunk egy XMLHttpRequest() a new kulcsszóval. A XMLHttpRequest használatával kéréseket tudunk küldeni a szerver felé HTTP vagy HTTPS protokollon keresztül, majd annak válaszát vissza tudjuk küldeni az oldalra.

GET() metódust használtunk, ami a xhttp.open() függvényben található, melynek segítségével kiolvassuk az oldal linkjéből a személy azonosítóját, amelyet átadunk a userFetch.php kiterjesztésű fájlnek.

```
$sql="SELECT * FROM users INNER JOIN megrendeles ON users.id =  
megrendeles.userId INNER JOIN termék ON termék.id = megrendeles.termekId  
INNER JOIN telepulesek ON telepulesek.id = users.telepules WHERE users.id = '{$q}'";  
$result = mysqli_query($dbconnect, $sql);
```

A userFetch.php fájlban először lekérdezést hajtunk végre, amely megkeresi azon felhasználókat, amelynek az adatbázisban lévő azonosítója megegyezik az XMLHttpRequest által küldött azonosítóval, amelyet eltárolunk a \$result változóban.

```
$i = 0;  
while ($sor = mysqli_fetch_array($result)) {  
    $i++;  
    $status = ($sor['status'] == 1) ? "Kész" : "Nincs kész";  
    $szamlazva = ($sor['szamlazva'] == 1) ? "Számlázva" : "";  
    $sar = number_format($sor['ar'] * $sor['raktaron'], 0, ",", " ");  
    echo "<tr>";  
    echo "<td>" . $i . "</td>";  
    // echo "<td>" <img src='../Webshop/img/termekekuj/{$sor['foto']}'`> "</td>";  
    echo "<td>" . $sor['termekNev'] . "</td>";  
    echo "<td>" . $sor['raktaron'] . " darab </td>";  
    echo "<td>" . $status . "</td>";  
    echo "<td>" . $sor['irsz'] . " " . $sor['nev'] . " " . $sor['kiszallitasiCim'] . "</td>";  
    echo "<td style='font-size:18px'>" . $sar . " Ft" . "</td>";  
    echo "<td>" . $szamlazva . "</td>";  
    echo "</tr>";}
```

Létrehoztunk egy \$i változót, ami a lekérdezés sorait fogja számolni. A lekérdezés eredményén a mysqli_fetch_array() segítségével végigiterálunk. Létrehoztunk egy \$status változót, amibe logikai operátor segítségével egyenlőséget vizsgálunk, ha az értéke 1-t ad vissza, akkor a „Kész” szöveg jelenik meg az adminisztrátor számára a táblázatban, más szám esetén pedig a „Nincs kész” szöveg. A \$szamlazva változó ugyanezen logika alapján működik. A következő az \$sar változó, amelyben összeszorozzuk a termék árát és a darabszámot. A következő kódsorokban állítjuk elő a táblázatot, amelyet az echo-val íratunk ki.

```

<form>
    <select name="users" onchange="showUser(this.value)">
    <option value="">Felhasználónév kiválasztása:</option>
        <?php print_r($ki) ?>
    </select>
</form>

```

A HTML oldal legördülő listáját egy űrlapban helyeztük el. A lista a `<select></select>`-en belül érhető el. A `<select>`-hez hozzárendeltünk egy `onchange` eseménykezelőt, vagyis ha kiválasztásra kerül valamilyen érték, meghívásra kerül a `showUser()` függvény. Az érték a `XMLHttpRequest()`-ben átadott azonosító lesz. A kiválasztás után az adatbázisból beolvassuk az értékeket, amelyek a `$ki` változóban kerülnek eltárolásra, így azokat itt azonnal ki tudjuk írni a böngészőben.

2.3.6. Felhasználók kezelése

Ezen az oldalon felhasználókat tudunk tiltani, illetve engedélyezni.

```
SELECT * FROM users;
```

Ez a lekérdezés csak azokat a felhasználókat kérdezi le, akiknek a jogosultsági szintjük a „user”. A kódban található egy szelekció, `if (mysqli_num_rows($eredmeny) > 0)` amelynek segítségével vizsgálatot hajtunk végre, hogy az előbb említett lekérdezés ad-e vissza sort.

```

$kimenet = "<div class='tableContainer'>
    <table>
    <tr>
        <th>Felhasználó neve</th>
        <th>Jogosultsági szint</th>
        <th>Művelet</th>
    </tr>
";
while ($sor = mysqli_fetch_array($eredmeny)) {
    $muvelet = ($sor['engedelyezes'] == 1) ? 'Kitiltás' : 'Engedélyezés';
    $szin = ($muvelet == "Kitiltás") ? 'background-color:red' : 'background-
color:green';
    $admin = ($sor['user_type'] == 'admin' ? "color:red" : "");
    $kimenet .= "<tr>

```

```

        <td style='$admin'>{ $sor['name']}{ $sor['user_type']}

```

A \$muvelet változó értéke a feltételes operátor (?) segítségével kerül meghatározásra. Ha a \$sor['engedelyezes'] értéke 1, akkor a \$muvelet értéke „Kitiltás” lesz, ellenkező esetben „Engedélyezés”. A \$szin változó értéke ismét a feltételes operátor (?) segítségével kerül meghatározásra. Ha a \$muvelet értéke „Kitiltás”, akkor a gomb háttérszíne piros lesz, ellenkező esetben zöld. Ezután folytatódik az adatok kiírása a táblázatban. A táblázat utolsó cellájában (<td>) egy gombot (<button>) jelenít meg. A gomb szövege és értéke a \$muvelet változó értékével megegyezik. A gomb data-id attribútuma a \$sor['id'] értékét kapja. A gomb stílusát a \$szin változó határozza meg. Ha a \$sor['engedelyezes'] értéke 1, akkor a felhasználó engedélyezve van, a gomb pedig piros színű és „kitiltás” feliratú lesz, amire ha rákattintunk, akkor kitiltjuk a felhasználót. Ellenkező esetben, ha a \$sor['engedelyezes'] értéke 0, akkor a felhasználó ki van tiltva és a gomb zöld színű és a szövege az „Engedélyezés” felirat jelenik meg, amivel viszont engedélyezni tudjuk őt.

2.3.7. Törlés

Az admin fő oldalán található táblázat utolsó cellájában található a törlés gomb. Ha a \$_GET['id'] létezik, ami azt olvassa ki az adatbázisból, hogy létezik-e olyan azonosító, ami az adott sorban van feltüntetve, akkor a kód többi része fut le. Először betölti a kapcsolat.php fájlt, majd az \$id változó értékét a \$_GET['id'] értékére állítja be. Ezután több SQL parancsot hajt végre, amelyek során törli az adott termék azonosítójú (\$id) termékeket a „kedvenctermekek”, „megrendeles” és termék táblákból. Három lekérdezést külön-külön változóban tároltunk el.

```

$kedvenc = "DELETE FROM kedvenctermekek WHERE termékId = {$id}";
mysqli_query($dbconnect, $kedvenc);
$megrendeles = "DELETE FROM megrendeles WHERE termékId = {$id}";

```

```
mysqli_query($dbconnect, $megrendeles);  
$sql = "DELETE FROM termek  
WHERE id = {$id}";  
mysqli_query($dbconnect, $sql);
```

Ahhoz, hogy működjön az \$sql változóban lévő törlés, ki kellett törölnünk az adott terméket a kedvencek (\$kedvenc) és a megrendelések (\$megrendeles) közül is. Ezek nélkül az \$sql változóban levő lekérdezés nem működött volna, mert az adatbázisban e két tábla között kapcsolat van. A `header("location: kategoria.php");` kód segítségével visszaléptetjük az adminisztrátort az admin fő oldalára.

2.3.8. Termék módosítása

Az oldal kinézete, működése nagy részben megegyezik a 4.11.3-as fejezetben ismertetetthez hasonlóan, ezért csak a különbségeket részletezzük. A módosítás felülete annyiban tér el, hogy az adatokat a \$_GET['id'] segítségével kiolvassuk az adatbázisból és egyenlőséget vizsgálunk vele. Az adott terméket, amit kiválasztottunk módosítás gyanánt, ennek az azonosítóját megnézzük, hogy az adatbázisban lévő sorok közül szerepel-e benne, és ha igen, akkor az azonosító alapján kiolvassuk a hozzá tartozó adatokat, majd megjelenítjük az adminisztrátoroknak.

2.3.9. Részletek

Az adott termékre, ha rákattintunk, a részleteit láthatjuk, amit azért építettünk bele a programba, mert látni szerettük volna, hogy az, ami az admin oldalon megjelenik termék specifikáció, az hogyan jelenik meg a felhasználói oldalon is. Az \$id változóban eltároljuk az adott termék azonosítóját. Ezután az \$sql változóban eltároljuk a lekérdezést. Lekérdezést hajtunk végre és az eredményt egy asszociatív tömbbe mentjük el. Ezután a tömbből kiolvassuk a termék nevét, gyártóját, leírását, árát és fotóját, majd ezeket elmentjük. A változóba táblázatos formában fűzzük össze a kapott adatokat.

3. Adatbázis

Projekt munkánkból a felhasználó a weboldalt látja, csak ezzel, az interfésszel áll kapcsolatban. A felhasználó elől el van rejtve a weboldal mögötti adatbázis, aminek a megtervezése nagyon fontos feladatunk volt. A weboldal az adatbázisból olvassa ki az adatokat, és felvitelkor, módosításkor, törléskor minden esetben módosul annak tartalma.

3.1. Kiinduló egyedek

Az adatbázisunk 4 egyedből állt az első tervezési folyamatkor, ez viszont bővült a továbbfejlesztések miatt.

3.1.1. Kategória

A kategória egyed azért jött létre, mert nem csak egy adott számítógép-alkatrészt szeretnénk értékesíteni, hanem több különböző terméket. Ebben az egyedben lesznek az alkatrészeknek a termékmegnevezései. Például: processzor, alaplap, videokártya.

3.1.2. Gyártó

A gyártó egyedben tároljuk a különböző gyártók neveit, amelyek különböző kategóriában gyártanak termékeket. Például: Intel, AMD, ASUS.

3.1.3. Felhasználó

Ebben a táblában foglalnak helyet a különböző személyek, akik regisztráltak, megkülönböztetjük őket, jogosultsági szintjük szerint is. Rögzítjük még a felhasználók személyes adatait, amelyeket a rendelés véglegesítés oldalon adnak meg. A felhasználókat megkülönböztetjük engedélyezés szinten is, ami azt jelenti, hogy kitiltási jogosultságot is kaphatnak. A felhasználók a rendelés adatai kitöltésénél megadhatják azt, hogy szeretnék-e, hogy az adataikat a következő vásárlásuk során is változás nélkül használni. Ezért hoztuk létre a mentve mezőt, ami azért felelős. Például: e-mail cím, jelszó, vezetéknév, keresztnév, bankkártya szám, bankkártya kód (CVV/CVC), engedélyezés.

3.1.4. Termék

A termék egyedben foglalnak helyet az általunk felvitt termékek lényeges tulajdonságai. Például: gyártónév, termék neve, raktáron, ár.

3.1.5. Felmerült kérdések

3.1.5.1. Népszerű termékek létrehozása

Az adatbázis megtervezése során először még nem merült fel bennünk az, hogy a felhasználók tudjanak kedvelni egy terméket. Továbbfejlesztésként létrehoztuk ez az opciót is. Azok a termékek találhatóak ebben az egyedben, amelyeket a vásárlók kedvelnek.

3.1.5.2. Az admin és a felhasználó egyed szétválasztásának kérdése

A tervezés során felmerült bennünk az a kérdés, hogy az adminisztrátorokat és a felhasználókat külön táblákban tároljuk-e. Végül arra jutottunk, hogy az adminokat és a felhasználóknak egy külön mező alapján (`user_type`) különböztetjük meg, amelynek két értéke lehet, mégpedig a `user` vagy az `admin`.

3.1.5.3. A regisztrációs kötelezettség

A projektmunkánkban úgy döntöttünk, hogy aki szeretne terméket venni az oldalon, annak létre kell hoznia egy felhasználói fiókot ennek érdekében.

3.1.5.4. A gyártó és kategória kapcsolatának kérdésköre

A tervezésünk során beleütköztünk abba az akadályba, hogy a gyártókat és a gyártóhoz tartozó kategórianéveket milyen módon kössük össze. Végül rájöttünk arra, hogy e két egyed közé egy kapcsolótáblát kell létrehoznunk, ami összeköti a tulajdonságaikat.

3.1.5.5. A lakcím – szállítási cím tárolásának kérdése

A tervezésünk során először az irányítószámot és a hozzá tartozó települést egy egyedben tároltuk el a szállítási címmel együtt. Végül arra jutottunk, hogy a településeket és a hozzájuk tartozó irányítószámokat egy külön táblában tároljuk el. A vásárló kiszállítási címét pedig a megrendelesek táblában vesszük fel, hiszen eddig a `users` táblában volt.

3.1 Kapcsolatok meghatározása

katategoria => gyarto (N:M) : Azért N:M-es kapcsolat van a két tábla között, mert egy gyártó tud több termékkategóriát gyártani, és egy termékkategória több gyártóhoz

tartozhat. Ezért a továbbiakban e két tábla közé létrehoztuk egy kapcsolótáblát, amivel szétbontottuk két 1:N-es kapcsolatra.

gyarto => gyartokategoria (1:N) : Egy gyártóhoz a „gyartokategoria” táblában több sor tartozhat. A kapcsolótábla segítségével egy gyártóhoz több kategóriát tudunk rendelni.

kategoria => gyartokategoria (1:N) : Egy kategóriához a „gyartokategoria” táblában több sor tartozhat. A kapcsolótábla segítségével egy kategóriához több gyártót tudunk rendelni.

termek <=> users (N:M) : Egy terméket több felhasználó is meg tud venni, egy felhasználóhoz több termék is tartozhat.

termek => megrendeles (1:N) : Ezen táblák között 1:N-es kapcsolat van, hiszen egy terméket többször is lehet rendelni, de egy konkrét termék, amit megrendeltek, az csak egyszer szerepel a termék táblában.

gyartokategoria => termék (1:N) : A „gyartokategoriában” találhatóak a gyártók és a hozzájuk tartozó kategóriák. Egy „gyartokategoriához” több termék tartozik, de egy konkrét terméknek csak egy „gyartokategoriája” lehet.

felhasznalo => kedvenctermekek (1:N) : E két tábla között 1:N-es kapcsolat van. Azok a felhasználók tudják kedvelni a termékeket, akik beléptek. Egy felhasználó több terméket is tud kedvelni. Egy kedvelt termék egyszerre csak egy felhasználóhoz tartozik.

termek => kedvenctermekek (1:N) : Egy termék csak egyszer szerepelhet kedvenc termékként. Egy kedvenc termék csak egy termékhez tartozhat.

telepulesek => megrendeles (1:N) : Egy településen lehet több rendelés, de egy megrendelést csak egy településre lehet küldeni.

3.3.Táblák

3.3.1. A kategoria tábla

A táblában a különböző termékek kategóriái vannak eltárolva.

Oszlop	Típus	Leírás
kategoriaID	int	az azonosító arra szolgál, hogy a „kategoria” táblában azonosítani tudjuk a kategóriák neveit
kategoriaNev	varchar	a termék kategória neve pl.: processzor

kategoriaID	kategoriaNev
1	processzor
2	alaplap
3	gépház

27. ábra kategoria tábla adatokkal

3.3.2. A gyarto tábla

Az egyes termékek gyártói helyezkednek el ebben a táblában.

Oszlop	Típus	Leírás
gyartoId	int	az azonosító arra szolgál, hogy a „gyarto” táblában azonosítani tudjuk a gyártók neveit
gyartoNev	varchar	a gyártó nevét tartalmazza

gyartoid	gyartoNev
1	AMD
2	INTEL
3	ASUS

28. ábra gyarto tábla adatokkal

3.3.3. A gyartokategoria tábla

A tábla egy kapcsolótábla a „gyarto” és a „kategoria” tábla között.

Oszlop	Típus	Leírás
gyartoKategoriaId	int	az azonosító arra szolgál, hogy a gyartokategoria táblában azonosítani tudjuk a sorokat
kategoriaID	int	az azonosító idegen kulcs szerepet tölt be a táblában, ami kapcsolódik a kategoria tábla azonosító mezőjéhez
gyartoId	int	az azonosító idegen kulcs szerepet tölt be a táblában ami kapcsolódik a gyarto tábla azonosító mezőjéhez

gyartoKategoriaId	kategoriaID	gyartold
1	1	1
2	1	2
3	2	3

29. ábra gyartokategoria tábla adatokkal

3.3.4. A termek tábla

A gyártóknak a lényeges tulajdonságai találhatóak a táblában.

Oszlop	Típus	Leírás
id	int	az azonosító arra szolgál, hogy a „termek” táblában azonosítani tudjuk a termékeket
gyartoKategoriaId	int	az azonosító idegen kulcs szerepet tölt be a táblában, ami kapcsolódik a gyartokategoria tábla azonosító mezőjéhez
termekNev	varchar	az adott termék nevét tartalmazza
foto	varchar	az adott termék fényképét tartalmazza, ami egy .png, .jpg vagy .gif kiterjesztésű (kép formátumú) fájl
leiras	text	az adott termék egyéb tulajdonságait tartalmazza
darab	int	az adott termékből hány darab van készleten
ar	int	az adott termék nettó ára forintban
akcio	int	0 = a termékhez nincs akció 1 = a termék akciós

id	gyartoKategoriaId	termekNev	foto	leiras	darab	ar	akcio
1	1	Ryzen 5 5600X	1_Ryzen 5 5600X_1679839394.jpg	<table> <tbody> <tr> <td>Típus</td> </tr> </tbody> </table>	0	71990	0
2	1	Ryzen 5 5600	1_Ryzen 5 5600X_1679839394.jpg	<table> <tbody> <tr> <td>Típus</td> </tr> </tbody> </table>	26	57990	0

30. ábra termék tábla adatokkal

3.3.5. A megrendeles tábla

Az adott felhasználók által kiválasztott termékek találhatóak a táblában.

Oszlop	Típus	Leírás
id	int	az azonosító arra szolgál, hogy a „megrendeles” táblában azonosítani tudjuk az adott megrendelő megrendeléseit
usersId	int	az azonosító idegen kulcs szerepet tölt be a táblában, ami kapcsolódik a users tábla azonosító mezőjéhez
termekId	int	az azonosító idegen kulcs szerepet tölt be a táblában, ami kapcsolódik a termék tábla azonosító mezőjéhez
raktaron	int	a felhasználó hány darabot rendelt a termékből
status	int	0 = a rendelés folyamatban van 1 = vásárló megrendelte a termékeket
idopont	date	a felhasználó mikor rendelte a terméket
telepules	int	az azonosító idegen kulcs szerepet tölt be a táblában ami kapcsolódik a telepulesek tábla azonosító mezőjéhez
utcaNev	varchar	a felhasználó által megadott utca
hazszam	varchar	a felhasználó által beírt utcához tartozó házszám
torles	int	0 = nem törlendő áru a kosárból 1 = törlendő áru a kosárból
samlazva	int	0 = nincs számlázva 1 = online formában számlázva 2 = postán is kiküldés
sorszam	int	a számla generálása során automatikusan növekszik

id	usersId	termekId	raktaron	status	idopont	telepules	utcaNev	hazszam	torles	samlazva	sorszam
416	18	36	1	1	2023-04-26	693	Munkás	6	0	1	10
415	18	10	1	1	2023-04-26	693	Munkás	6	0	1	10

31. ábra megrendelés tábla adatokkal

3.3.6. A kedvenctermekek tábla

Ebben a táblában az összes olyan termék megtalálható, amiket a vásárló kedvelt.

Oszlop	Típus	Leírás
id	int	az azonosító arra szolgál, hogy a „megrendelo” táblában azonosítani tudjuk a megrendelő adatait
termekId	int	az azonosító idegen kulcs szerepet tölt be a táblában, ami kapcsolódik a termék tábla azonosító mezőjéhez
usersId	int	az azonosító idegen kulcs szerepet tölt be a táblában, ami kapcsolódik a users tábla azonosító mezőjéhez

id	termekId	usersId
2	8	1
3	9	1
4	10	1

32. ábra kedvenctermekek tábla adatokkal

3.3.7. A telepulesek tábla

Itt a településeket tároltuk az irányítószámmal együtt.

Oszlop	Típus	Leírás
id	int	az azonosító arra szolgál, hogy a “telepulesek” táblában azonosítani tudjuk a településeket
irsz	int	a település irányítószáma
nev	varchar	a település neve

id	▼ 1	irsz	nev	megye
4154		9955	Szentgotthárd	VA
4153		9740	Bük	VA

33. ábra A telepulesek tábla adatokkal

3.3.8. A users tábla

Ebben a táblában az adminok és a felhasználók találhatók. Az adminoknak joguk van ahhoz, hogy az adatokat karbantarthassák az adatbázisban weboldalon keresztül. A felhasználók pedig csak vásárolni tudnak. Itt foglalnak helyet a felhasználók lényeges tulajdonságai.

Oszlop	Típus	Leírás
id	int	az azonosító arra szolgál, hogy a “users” táblában azonosítani tudjuk a felhasználókat és az adminokat
name	varchar	az adott személy felhasználónevét tartalmazza
email	varchar	az adott felhasználó e-mail címe
password	varchar	a felhasználó vagy admin titkosított jelszava
user_type	varchar	user = átlagos felhasználó admin= adminisztrátor
keresztNev	varchar	az adott felhasználó keresztnéve
vezetekNev	varchar	az adott felhasználó vezetéknéve
kartyaszam	varchar	a felhasználó bankkártyaszáma
kartyaKod	varchar	a felhasználó kártyaszámához tartozó CVC/CVV
telefonszam	int	a felhasználó telefonszáma
mentve	int	0 = rendeléskor bekért személyes adatok a felhasználó számára nem jelennek meg újra a következő rendelésénél 1 = rendeléskor bekért adatok a felhasználó számára nem kell újra beírnia a következő rendelésénél
engedelyezes	int	0 = kitiltott felhasználó 1 = engedélyezett felhasználó

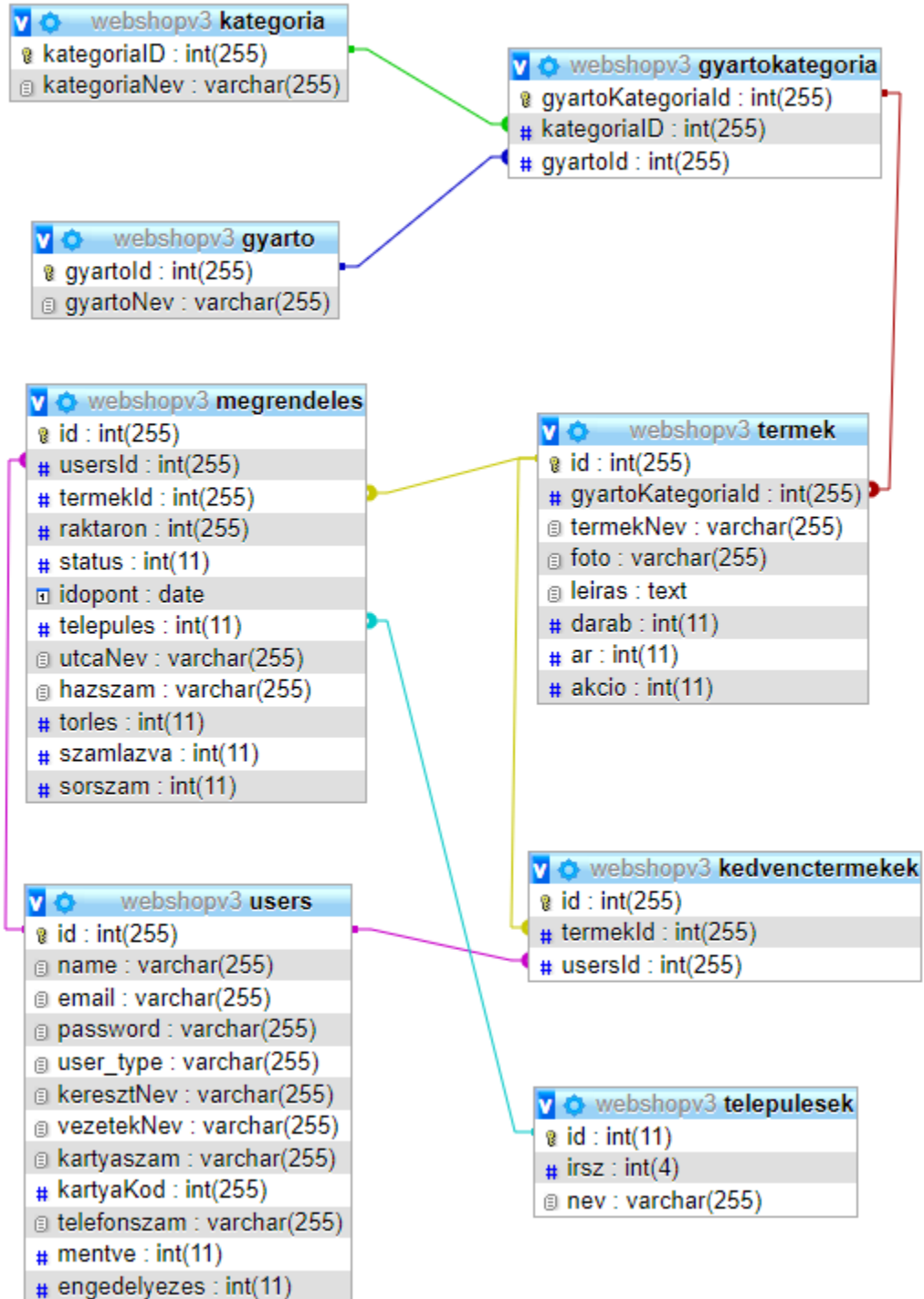
kartyaszam	kartyaKod	telefonszam	mentve	engedelyezes
5344-1234-1234-1234	452	06302125411	1	1
1111-1111-1111-1111	341	06205418422	1	1

34. ábra A users tábla adatokkal 1

id	name	email	password	user_type	keresztNev	vezetekNev
1	DrSeres	drseres@gmail.com	7c4a8d09ca3762af61e59520943dc26494f8941b	admin	Seres	Szabolcs
4	user	user@gmail.com	7c4a8d09ca3762af61e59520943dc26494f8941b	user	Kiss	János

35. ábra A users tábla adatokkal 2

3.3.9. Adatbázis diagram



36. ábra Adatbázis diagram

4. Tesztelés

A weboldalaink HTML kódrészleteit a <https://validator.w3.org/> oldalán ellenőriztük. Akadtak apróbb hibák, amelyeket nem vettünk észre, mivel a böngészőben tökéletesen működött minden, de ennek ellenére ezeket kijavítottuk, mert tudjuk, hogy a keresőmotorok hibás weboldalakat nem fognak indexelni és megjeleníteni a találatok között.

A CSS kódrészleteket a <https://jigsaw.w3.org/css-validator/> oldalán ellenőriztük. A validátor nem talált hibát a CSS kódunkban.

A JavaScript kód írása közben `console.log()` segítségével folyamatosan az elejétől kezdve ellenőriztük a programozás során, hogy sikeresen tároltuk-e el minden adatot a változóban, és azt az eredményt kapjuk-e, amelyet várunk. A böngésző konzoljában minden lépés után ellenőriztük, hogy nem észlel-e hibát a programban.

A folyamatos JavaScript kód ellenőrzés azért fontos, mert ha csak a legvégén ellenőrizzük és hibát tartalmaz a program, akkor már sokkal nehezebb megtalálni, mint folyamatában. A JavaScript tesztelését több oldalon is bennehagytuk a kódunkban, mint például a 2.2.5 és a 2.2.6-os fejezetben leírt ellenőrzések során, amelyeket megnézhetjük, ha megnyitjuk a fejlesztői eszköztáron belül a konzolt.

A fenti képen látható, hogy folyamatosan ellenőriztük, azaz kiíratunk a konzolra a változóba eltárolt értékeket, így biztosak lehettünk benne, hogy a megfelelő értékek kerültek a kezünkbe.

A PHP kódok ellenőrzését a <https://www.phptools.online/php-checker> oldalán is ellenőriztük, habár a Visual Studio Code, ha hibát észlel a PHP kódrészletekben, azt piros aláhúzással jelzi. Továbbá több kiegészítő modult is letöltöttünk a VSC-hez, amelyek segítik a debugolást a php programozás során.

Összefoglalás

A webshopunkat eredetileg sokkal egyszerűbbre terveztük, de ahogyan dolgoztunk rajta, szinte napról-napra újabb és újabb ötleteink támadtak, hogy mivel lehetne még bővíteni, illetve látványosabbá tenni az oldalt.

Ez történt a megjelenített termékek esetében is, ugyanis eredetileg az összes kép háttere hófehér volt. Később arra gondoltunk, mennyivel egységesebb képet mutatna, ha képszerkesztő programmal eltávolítanánk a képekről a fehér hátteret, és így a weboldal színéből emelkednének ki a termékek.

Más webshopnál is láttuk, hogy lehet kedvelni termékeket, így kitaláltuk, hogy helyezünk el alattuk egy szív ikont, amellyel tulajdonképpen szavazhat egy termékre a felhasználó, viszont itt már az eredeti adatbázisunkat is tovább kellett bővíteni. Innen jött a további ötletünk, hogy amelyekre a legtöbb szavazat érkezett, jelenítsük meg a főoldalon, mint népszerű termékeket.

Az is eszünkbe jutott, hogy készítsünk kapcsolati űrlapot, ahol közvetlenül tudnak nekünk üzenetet küldeni a regisztrált felhasználók.

A hírlevél-küldés jelenleg még nem működik, de éles weboldal esetén majd érdemes lesz tovább dolgoznunk rajta, hiszen egy-egy jól célzott akciós hírlevéllel vissza tudjuk csábítani a felhasználókat az oldalunkra.

Szintén a továbbfejlesztési lehetőségek között maradt az online fizetési rendszer, valamint a csomagküldő szolgáltatók közötti választási lehetőség.

Egy induló weboldalnak véleményünk szerint kiváló ingyenes reklámozási lehetőséget jelentenek a közösségi oldalak, ezért érdemes létrehozni Facebook vagy Twitter oldalakat is. Jó ötletnek tartanánk a Youtube csatornát is, ahol bemutatnánk rövid videókban a termékeket, melyiket miért tartjuk jónak, vagy egyéb karbantartási tanácsokat.

Terveztünk még olyan oldalt is, ahol online meg lehet tervezni és össze lehet állítani egy komplett számítógépet. Választó menüből ki lehetne választani a szükséges részeit, és ha valamit nem választ a felhasználó, akkor a program figyelmeztetné, hogy pl. alaplap nélkül nem fog működni, vagy ha egy alkatrész nem kompatibilis a már kiválasztottakhoz, akkor eleve nem engedné kiválasztani.

Projekt munkák elérhetősége a Github-on: <https://github.com/DrSeres/2-14Szakdolgozat>

Online megtekinthető a weboldalunk a nethely ingyenes tárhelyén:

<http://www.projectmunka.nhely.hu/Webshop/index.php>

Források

1. Appsloveworld: <https://www.appsloveworld.com/php/837/countdown-timer-reloading-again-on-refreshing>
2. CKEditor: <https://ckeditor.com/>
3. Codepen: <https://codepen.io/medrupaloscil/pen/QWrowxK>
4. Composer: [https://en.wikipedia.org/wiki/Composer_\(software\)](https://en.wikipedia.org/wiki/Composer_(software))
5. Edureka: <https://www.edureka.co/community/94514/how-to-get-input-field-value-using-php>
6. Felhasznált fotók: <https://computeremporium.hu>
7. Formssubmit: <https://formssubmit.co/>
8. GitHub: <https://github.com/dompdf/dompdf>
9. GitHub: <https://github.com/dompdf/dompdf>
10. SourceForge: <https://sourceforge.net/projects/dompdf.mirror/>
11. Stack Overflow: <https://stackoverflow.com/questions/20728839/get-element-by-classname-with-domdocument-method>
12. Stackoverflow: <https://stackoverflow.com/questions/1283327/how-to-get-url-of-current-page-in-php>
13. Stackoverflow: <https://stackoverflow.com/questions/1506527/how-to-replace-if-statement-with-a-ternary-operator>
14. Stackoverflow: <https://stackoverflow.com/questions/1685860/how-do-i-get-the-last-inserted-id-of-a-mysql-table-in-php>
15. Stackoverflow: <https://stackoverflow.com/questions/2108663/in-php-how-to-remove-specific-class-from-html-tag>
16. Stackoverflow: <https://stackoverflow.com/questions/28433798/php-get-length-of-digits-in-a-number>
17. Stackoverflow: <https://stackoverflow.com/questions/44102855/remove-class-using-javascript-inside-of-php>
18. YouTube: <https://www.youtube.com/watch?v=D5nnrGSDUzI>

Ábrajegyzék

1. ábra Menü belépett felhasználó esetében	6
2. ábra Menü és akciós kategóriák carousel	6
3. ábra Termékkategóriák a kezdőoldalon	7
4. ábra Kedvelt termékek a kezdőoldalon	8
5. ábra Lábléc	8
6. ábra Bejelentkezés oldal	9
7. ábra Termékek, ha a felhasználó nem lépett be az oldalra	9
8. ábra Termékek megjelenése a belépett felhasználó esetében	10
9. ábra Bővebb információk egy kiválasztott termékről	11
10. ábra Captcha kód generátor	11
11. ábra Kapcsolati űrlap	11
12. ábra Kapott e-mail, a felhasználó által küldött üzenetről	12
13. ábra Felhasználó által kedvelt termékek	12
14. ábra Kosár tartalmának megtekintése	13
15. ábra Rendelés megerősítése, adatok megadása	14
16. ábra Letölthető PDF számla	15
17. ábra Adminisztrátor számára megjelenő menü	15
18. ábra Termékek az admin oldalon	16
19. ábra Új gyártó felvitele	17
20. ábra Új termék felvitele	18
21. ábra Kedvelt termékek	19
22. ábra Megrendelések megtekintése	19
23. ábra Felhasználók kezelése	19
24. ábra Kitiltott felhasználó	20
25. ábra Jogosulatlan belépési próbálkozás figyelmeztető üzenet	20
26. ábra Javascript ellenőrzés, sikeresen eltároltuk-e az adatokat	32
27. ábra kategoria tábla adatokkal	55
28. ábra gyarto tábla adatokkal	55
29. ábra gyartokategoria tábla adatokkal	56
30. ábra termék tábla adatokkal	56
31. ábra megrendelés tábla adatokkal	57
32. ábra kedvenctermekek tábla adatokkal	58

33. ábra A telepulesek tábla adatokkal	58
34. ábra A users tábla adatokkal 1	59
35. ábra A users tábla adatokkal 2	59
36. ábra Adatbázis diagram	60