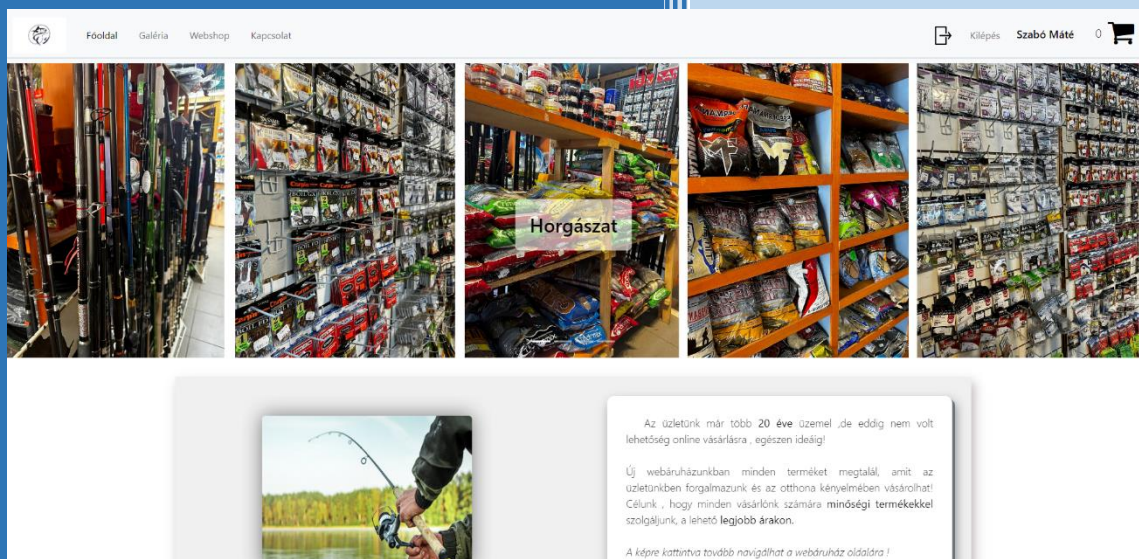


2023

Horgászboltunk az internetre költözik



Szabó Máté – Kordics Balázs
Nógrád megyei Szakképzési Centrum
Szent-Györgyi Albert Technikum
szakma megnevezése:
szoftverfejlesztő és -tesztelő technikus
szakma azonosító: 5-0613-12-03
2023.04.24.

Tartalom

1.	Felhasználói dokumentáció	4
1.1.	Főoldal.....	4
1.2.	Galéria	5
1.3.	Díszállatok.....	6
1.4.	Webshop oldal.....	7
1.5.	Megrendelés megerősítése oldal	9
1.6.	Belépés	10
1.7.	Regisztráció.....	10
1.8.	Adminisztrációs oldalak.....	11
1.8.1.	Admin belépés	12
1.8.2.	Lista nézet	12
1.8.3.	Megrendelések	12
1.8.4.	Felvitel	12
1.8.5.	Módosítás.....	13
1.8.6.	Törlés	13
2.	Fejlesztői dokumentáció	14
2.1.	Adatbázis tervezés.....	14
2.1.1.	Adatbázis célja	14
2.1.2.	Egyedek meghatározása	14
2.1.3.	Kapcsolatok	15
2.1.4.	Táblák meghatározása	16
2.1.5.	Adatbázis diagram	22
2.2.	Kódmagyarázat.....	22
2.2.1.	Kezdő oldal.....	22
2.2.2.	Regisztráció	22
2.2.3.	Belépés	24

2.2.4. Felhasználói adatok módosítása	25
2.2.5. Kilépés.....	25
2.2.6. Kosár	25
2.2.7. API.....	31
2.2.8. Kereső funkció.....	35
2.2.9. Képek nagyítása.....	35
2.2.10. Adminisztrációs lista	36
2.2.11. Termékek felvitele	37
2.2.12. Termékek módosítása	39
2.2.13 Termék törlése	40
2.2.14. Tesztelés	40
3. Felhasznált technológiák	42
3.1. Fejlesztői környezet	42
3.1.1. Visual Studio Code.....	42
3.1.2. GitHub	42
3.1.3. Photopea	42
3.2. Frontend eszközök	43
3.2.1. HTML.....	43
3.2.2. CSS	43
3.3. Backend eszközök	44
3.3.1. JavaScript	44
3.3.2. PHP.....	44
3.3.3. MySQL és XAMPP.....	44
Összefoglalás.....	46
Források.....	48
Ábrajegyzék.....	50

Bevezetés

Záró projektmunkánk alap ötletét egy már meglévő és működő bolt adta. Kordics Balázsék egy Horgász - Díszállat - Háztartási boltot működtetnek évek óta. A boltnak eddig még nem készült weboldala. Manapság viszont egy weboldal elengedhetetlen egy üzlet számára, ahol jelentősebb forgalomnövekedést szeretnének elérni és ismertté válni szélesebb földrajzi környezetben is.

A kisvállalkozások esetében sokszor hallottuk, hogy a weboldal a szükséges rossz, és csak plusz költséget jelent az egyébként is jól működő vállalkozásnak. Ezért úgy gondoltuk bebizonyítjuk, hogy számukra is fontos lehet, egy a mai felhasználói elvárásoknak megfelelő üzleti weboldal, online vásárlási lehetőséggel.

Véleményünk szerint azzal, ha egy üzlet az „internetre költözik”, végtelen üzleti lehetőségek nyílnak meg előtte, hiszen nem csak a szűk földrajzi környezetből érkehetnek majd a vásárlók, hanem akár az ország egész területéről rendelhetnek majd a leendő ügyfelek, illetve még a határon túlról is. Az elmúlt évek eseményei, a pandémia, a bezárás időszaka is megmutatta, hogy azok az üzletek tudták könnyedén átvészelni ezt a válságot, akik lehetőséget nyújtottak ügyfeleiknek az online vásárlásra.

Ezen az ötleten tovább indulva, nem csak egy átlagos weboldalt terveztünk létrehozni, ahol bemutatjuk a termékválasztékot, és a különböző információkat a boltról, hanem egy funkcionális működő webshopot, ahol a vásárlók kiválaszthatják a kívánt terméket, kosárba rakhatják és meg is rendelhetik. Ennek az előnye, hogy a vásárló, az otthona kényelméből böngészhet a bolt kínálatából és rendelheti meg a számára tetsző termékeket.

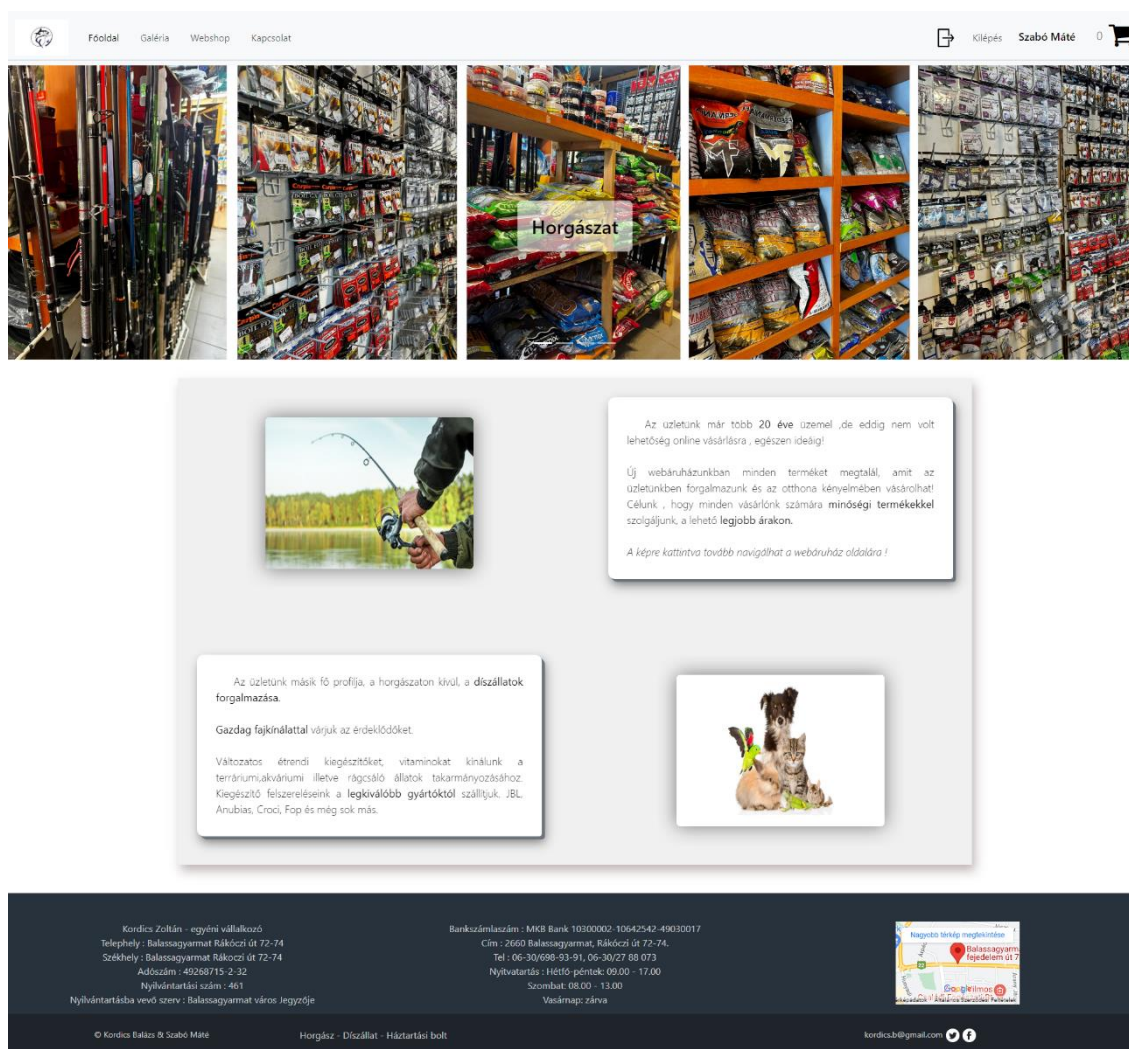
Dolgozatunk első fejezetében részletesen bemutatjuk a weboldalunk működését, és a különböző oldalakat, amelyekkel a felhasználó találkozik az interneten. A második fejezetben az adatbázis-tervezés lépéseit, a weboldalakkal kapcsolatosan pedig azokat a programozási részleteket emeltük ki, amelyekre büszkék vagyunk, hogy meg tudtuk valósítani. A fejezet végén összefoglaljuk a munkánk során felhasznált technológiákat. Végezetül az összefoglalásban értékeljük a közös munkánkat, és felvetjük a továbbfejlesztési lehetőségeket, amelyeket majd akkor fogunk tudni megvalósítani, ha tovább mélyítjük az ismereteinket a webprogramozás területén.

1. Felhasználói dokumentáció

Ebben a fejezetben részletesen bemutatjuk, hogyan tudja használni a látogató a weboldalunkat, milyen oldalakat látogathat meg, hogyan tud navigálni az oldalak között a menü, valamint a képek segítségével.

1.1. Főoldal

A látogató a „main.php” nevű főoldalunkkal találkozik először. Éppen ezért itt nagy gondot fordítottunk a dizájnrá, de ugyanakkor figyelembe vettük azt is, hogy a legfontosabb információkat is tartalmazza. Itt kap helyet magának a boltnak a bemutatása, mi minden kapható, néhány cikk az új termékekről, és hogyan juthat el hozzánk a vásárló. A könnyű keresés érdekében ide egy Google térképet is beágyaztunk.



1) ábra Kezdőoldal

Az oldal tetejére került a menü, ahonnan a további oldalakra navigálhat tovább a látogató. A navigáció alá egy ún. „carousel” került, ahol több kép váltakozik az aktuális

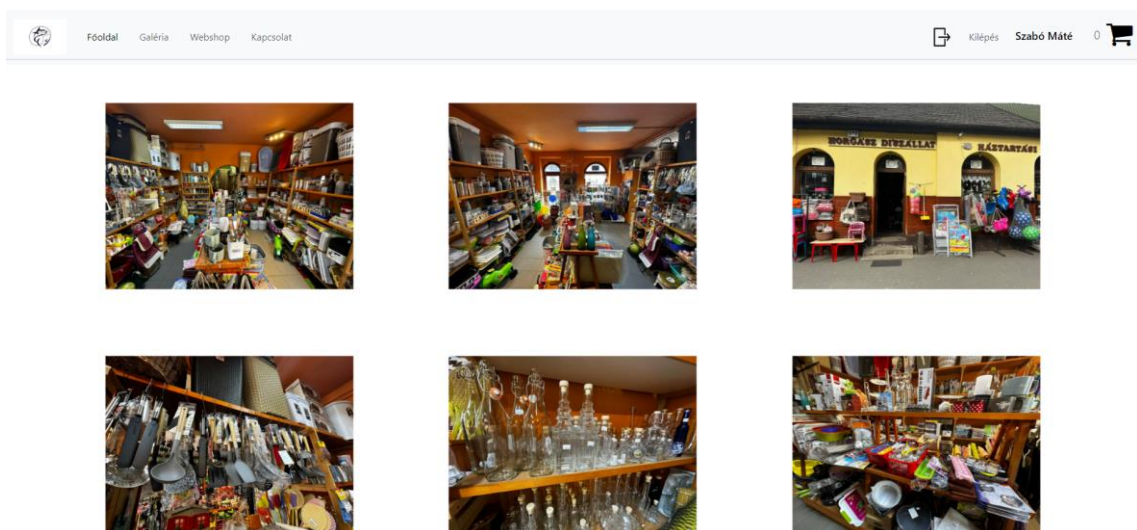
tudnivalókról, majd ez alatt a főbb termékcsoporthoz jelennek meg képekkel, majd végül a láblécen belül az üzlet elérhetőségei és a térkép látható.

Először azt terveztük, hogy a termékek bemutatása alá kerül a webshop, de utána olvastunk a weboldalak ún. hőterkép vizsgálatának, és a különböző vizsgálatok eredményeinek. Megállapították, hogy a látogatók többsége nem szívesen görget sokáig egy oldalon lefelé, a legtöbben azt a területet nézik meg, amelyik a böngésző ablakban eléjük tárul. Így ezt a gondolatot elvetettük, és külön oldalra terveztük meg a webshopot.

A projekt tervezésénél nem vettük számításba először, hogy hogyan működik a Google keresőmotorja, és hogy a kereséseket hogyan tudnánk optimalizálni, de minden tőlünk telhetőt megtettünk, hogy az új technikákat, amelyeket megtanultunk, helyesen alkalmazzuk a keresőoptimalizálás során.

1.2. Galéria

Ezen az oldalon a felhasználó megtekintheti a boltunkat kívülről és belülről. Azért gondoltuk fontosnak egy ilyen oldalt, hiszen itt aktuális képeket láthat az üzletünkről, illetve a termékek széles választékáról.



2) ábra Galéria

1.3. Díszállatok

[Főoldal](#)[Galéria](#)[Webshop](#)[Kapcsolat](#)

[Bejelentkezés](#)[Regisztráció](#)



A díszmadarak káprázatosan színes és barátságos fajából, a papagályok és díszpityék között minden madártartó megtalálhatja leendő kedvencét. Színes apróságok és kézzel nevelt szelíd nagy papagályok megvásárolhatók üzletünkben. A madarak nagy része ellenőrzött, tenyésztőinktől származik. Üzletünkben megtalálható illetve megrendelhető minden díszmadár számára az ideális táplálék, karika és egyéb szükséges felszerelés.

Kiunozódo méretben és színben a vevő és a díszmadár igényeire igazodva. A papagályok kedvelt időtöltése a járók, ezért mefeledkezünk el gondoskodni arról sem, ha és műanyag játékok, hinták, létrák, túrók mind a szárnyas kedvencünk boldog mindennapjainak kedvelői.

A táplálékok között minden fajnak megtalálható a legideálisabb összeállítás. Etetők és itatók széles választékban kaphatók minden méretű díszmadárhoz.

A díszhalak tartása egyre népszerűbb. Nem csoda, hiszen a nappalinkban csodálni a víz élővilágát nem csak lenyűgöző, hanem nyugtatóan is hat helikitus hétévezőnagjainkban.

Az állatkereskedésben fajták százaival találkozhat: Nem mindegyik ugyanolyan alkalmas azonban egy házi akváriumba. A táppal, növényekkel, vízminőséggel és vízhőmérséklettel kapcsolatos igények minden halnál különbözőek, így választásuknál ezeket is mindegyikre figyelembe kell venni. Szintén előre érdemes meggyőződni arról, hogy a kiválasztott díszhalak jól kijöjenek egymással.

A következőkben bemutatjuk Önnek a tíz legnépszerűbb édesvízi akváriumba való díszhalat. Ezenkívül részletesen elmagyarázzuk, mire érdemes ezek kiválasztásánál mindenképp odafigyelnie.





Ha azt gondolnánk, hogy kisállatot tartani gyerejénél, akkor nagyon is tévedünk.

Minden kisállat odafigyelést és sok törődést igényel. A kismamák népszerűségüket annak köszönhetik, hogy még kifejlett állapotukban is egy kólyok méretével rendelkeznek. A megfelelő tartáshoz rengeteg rácsálmiváló és sok – sok törődés kell. Ezek az állatok ugyanis szeretik a társaságot, és nagyon ragaszkodnak a gazdájukhoz.

Hogy milyen fajtát választunk, az már csak rajtunk múlik, mivel egyre több lehetőség áll rendelkezésünkre. Üzletünkben számos hörcsög, tengerimalac és nyúl fajtából válogathat.

Kordics Zoltán - egyéni vállalkozó

Telephely : Balassagyarmat Rákóczi út 72-74

Székhely : Balassagyarmat Rákóczi út 72-74

Adószám : 49268715-2-32

Nyilvántartási szám : 461

Nyilvántartásba vevő szerv : Balassagyarmat város Jegyzője

Bankkiváltás: MKB Bank 10300002-10647542-49030017

Cím : 2660 Balassagyarmat, Rákóczi út 72-74.

Tel : 06-30/608-93-91, 06-30/27-88 073

Nyitvatartás : Hétfő-péntek: 09.00 - 17.00

Szombat: 09.00 - 13.00

Vasárnap: zárva

Nyitvatartás: Hétfő-péntek: 09.00 - 17.00

Szombat: 09.00 - 13.00

Vasárnap: zárva

© Kordics Balázs & Szabó Mária


Horgász - Díszállat - Háztartási bolt

kordics4@gmail.com

3) ábra Díszállatok

Figyelmet kapott weboldalunkon a díszállat-tartás is, hiszen a boltunkban már évek óta forgalmazunk díszállatokat és egyéb felszereléseket. Fontosnak tartottuk, hogy népszerűsítsük a díszállattartást, olyan módon, hogy egy külön oldalon ismertetjük őket. Három csoportra különítettük el ezeket az állatokat: madarak, díszhalak, kisállatok.

A képekre kattintva tovább léphet a felhasználó egy olyan oldalra, ahol részletes leírást olvashat az állatokról.



Guppi


Diszhalak

A Karib-térségből származó édesvízi hal kétségtelenül az egyik legnépszerűbb akváriumi díszhal. A guppi igénytelen halnak számít, így az akváriumok világában kezdők számára is alkalmas.

[Több információ](#)

A színes, maximum 5 cm-es nagyságúra növő díszhal eredeti élőhelyén rajokban él. Az akváriumban ezért szintén kisebb csapatba érdemes költöztetni. A társaságkedvelő guppi kiválóan megérti magát más halfajtákkal – különösen a páncélosharcsákkal. A guppi egyébként nagyon gyorsan szaporodnak, és a szárazzeledelt, illetve az élő eledelt is szívesen fogyasztják.

- Eredet: Dél-Amerika északi részei, a Karib-térség egyes részei
- Hossz: 3-5 cm
- Tartás: rajokban
- Tartás: rajokban
- Nehézségi fok: egyszerű
- Akvárium: édesvízi akvárium, legalább 54 l-es űrtartalom (különböző hossz 60 cm)
- Vízértékek: optimális hőmérséklet 24-27°C, optimális pH-érték 6-7,5
- Szocializáció: különösen kedveli a páncélosharcsákat, de a nőstény harcoshalakat, a császárlazacokat, a daniókat, a törpe gurámikat, a vörbőstorkú diszcsukákat, a garnélákat és a közönséges vértessharcsákat is
- Eledet: száraz pehely, heti 1-3 alkalommal fagyasztott vagy élő növényekkel és állatokkal kiegészítve
- Különlegességek: nagyon könnyű a gondozása, szereti a sűrű növényzetet, különösen szeret úszni, gyorsan szaporodik (Aki tehát utóbbit szeretné elkerülni saját akváriumában, csak az egyik nemből vásárljon guppiakat, például csak hím guppiakat.)



Neonhal

Diszhalak

Világító piros és kék csíkjaival a neonhal igazi látványosságnak számít az akváriumban. Ezek a kis halak az Amazonas-medence régiójából származnak és csak rajokban érzik jól magukat. Az akváriumban tehát legalább 10 neonhalat tartson együtt – de igazából minél többet, annál jobb. Ez a mottó pedig az akváriumra is érvényes, mely legalább 60 l-es űrtartalmú kell, hogy legyen.

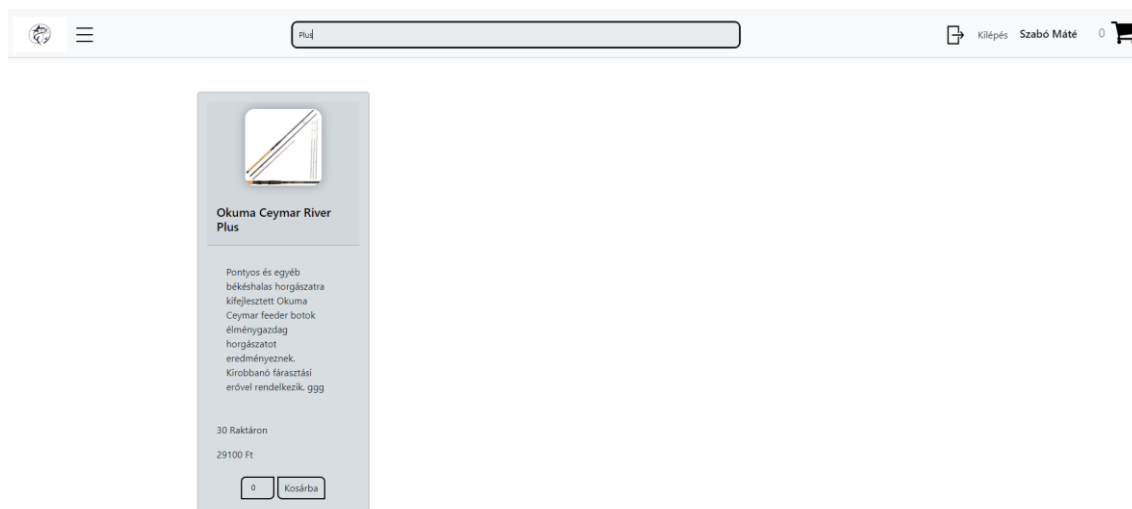
[Több információ](#)

4) ábra Halak

1.4. Webshop oldal

A webshop oldalán elhelyeztünk egy navigációt, hogy a látogató vissza tudjon térni a fő oldalra. A menü alatt találhatóak a rendelhető termékcsoportok kis kártyákon, amelyekre kattintva a webshopra navigálja a látogatót, ahol azok a termékek jelennek csak meg, amelyek abban a termékcsoportban találhatóak. A látogató tovább tudja szűrni a keresését alkategóriák szerint, amely jóval megkönnyíti a termékek közötti specifikus keresést. Például a látogató kereshet a horgászcikkek között, de leszűkítheti a keresést a horgászcikkekben belül csak a horgászbotokra is.

Megvalósítottunk egy olyan kereső funkciót is, melynek segítségével a termék olyan adataival is kereshet az árucikkek között a látogató, mint például a neve, cikkszama, vagy gyártója.

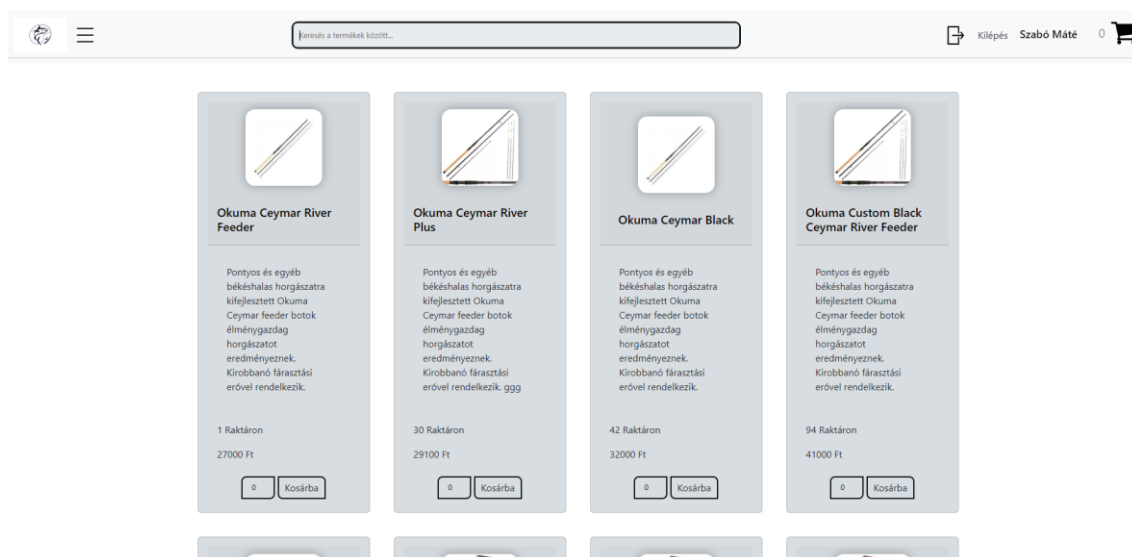


5) ábra Keresőmező működése

A keresőmezőbe beírva egy nevet, vagy akár egy név részletét, azonnal megjelennek a keresésnek megfelelő találatok. Ez a keresőmező sokkal gyorsabb keresést eredményez.

Ha a kereső mezőbe olyan termék nevét írja be a felhasználó, ami nincs, felugró ablakban megjelenik egy szöveg „Nincs ilyen termék az oldalon.”

A termékek kártyákon kerülnek megjelenítésre, felül a termék fotója, alatta a neve, a raktáron lévő mennyiség, a termék ára, egy rövid leírás és egy beviteli mező, ahol beállítható, hány darabot szeretne rendelni az adott termékből a látogató és egy küldés gomb „kosárba” felirattal.



6) ábra Termék kártyák megjelenése

Ha a felhasználó egy terméket helyez a kosárba, akkor a jobb felső sarokban megjelenik a bevásárlókocsi ikonja mellett a rendelt termékek darabszáma. Ha a bevásárlókocsira kattint a látogató, akkor felugró ablakban megjelenik egy táblázat, amely a rendelése adatait tartalmazza.



7) ábra Kosár tartalma

A táblázatban lehetősége van törölni termék kategóriánként, darabonként és az összes termék törlésére is lehetőséget biztosítottunk.

1.5. Megrendelés megerősítése oldal

A megrendelés elküldése gomb egy új oldalra navigálja a látogatót, ahol a megrendelt termékei összesítésre kerülnek, és ott véglegesítheti a megrendelését. Mivel ezt az oldalt

csak akkor célszerű megjeleníteni, amikor valóban rendel a felhasználó az oldalról, ezért nem kapcsoltuk a menühöz.



8) ábra Megrendelés megerősítése

A megrendelés megerősítése során a látogatónak nincs több dolga, hiszen a regisztráció során minden olyan adatot meg kellett már adnia, amely a megrendeléshez szükséges. A megrendelés megerősítése után az adatbázis frissül és az a mennyiség, amit a látogató megrendelt, törlődik a raktárkészletből.

1.6. Belépés

A látogatónak a belépéshez két mezőt kell kitöltenie, egy e-mailt és egy jelszót. Ha a felhasználó hibás adatokat írt be, figyelemfelhívó felugró szöveg jelenik meg.



9) ábra Bejelentkezés

1.7. Regisztráció

A regisztráció során 8 mezőt kell kitölteni: teljes név, felhasználónév, e-mail, jelszó, település, utca, házszám, telefon. Ez későbbiekben egyszerűsíti a rendelést, mivel itt már eltárolásra kerül a kiszállítási cím is. Minden mezőt kötelezően ki kell tölteni, úgy lesz érvényes a regisztráció. Ha üresen hagy a felhasználó egy mezőt, akkor az oldal nem

engedi továbblépni, amíg ki nem tölti megfelelően. Sikeres regisztrálás után átirányításra kerül a felhasználó a belépés oldalra.

The registration form (Regisztráció) contains the following fields and elements:

- Regisztráció** (Title)
- Teljes név** (Full name) input field
- E-mail cím** (Email address) input field, containing "kacsa@gmail.com"
- Jelszó újra** (Repeat password) input field, containing "*****"
- 1011 - Budapest I. kerület** (Address) dropdown menu
- Utca, házszám** (Street, house number) input field
- Telefonszám** (Phone number) input field
- Regisztrálok** (Register) button
- Van már fiókom? [Jelentkezz be itt!](#)

10) ábra Regisztrációs űrlap

1.8. Adminisztrációs oldalak

A bolt tulajdonosának készítettünk egy olyan adminisztrációs felületet, ahová belépve egyszerű tartalomkezelő felületekkel tud felvinni új termékeket, ami annyit jelent, hogy nincs szükség webprogramozói tudásra ahhoz, hogy tudjon árat módosítani, illetve tudjon termékeket törölni.

The product administration interface includes a search bar and a table of products.

Buttons: Megrendelések, Kijelentkezés, Keresés a termékek között, Új ár felvittele

Fotó:	Vonalkód	Név	Kategória	Alkategória	Felvitel dátuma	Darabszám	Ár	Művelet:
	00000000	Okuma Ceymar River Feeder	Horgász	Horgászbot	2023-03-03	1	27000	Törölés Módosítás
	00000000	Okuma Ceymar River Plus	Horgász	Horgászbot	2023-03-03	30	29100	Törölés Módosítás
	00000000	Okuma Ceymar Black	Horgász	Horgászbot	2023-04-14	42	32000	Törölés Módosítás

11) ábra Termékek adminisztrációs oldala

Ahhoz, hogy az adminisztrációs felületre be tudjon lépni az oldal tulajdonosa, úgy döntöttünk biztonsági okokból nem helyezünk el figyelemfelkeltő linket, vagy gombot, hogy elkerüljük az illetéktelen felhasználók próbálkozásait. Így akkor tud belépni a tulajdonos, ha a domain név után beírja a /admin szót, bár tudjuk, a hackerek ezzel is

szoktak próbálkozni egy oldal feltörésénél. Az adminisztrációs oldalon van lehetőség új adminokat létrehozni, ha az idő folyamán több adminisztrátorra lenne szükség.

1.8.1. Admin belépés

A domain név/admin megadása után, egy belépési felületre jut az oldal tulajdonosa, ahol meg kell adnia a felhasználónevét és jelszavát. Enélkül nem tud a további oldalakra lépni, még akkor sem, ha tudja a megfelelő oda vezető linket, mivel lapvédelemmel láttuk el az adminisztrációs oldalakat.

1.8.2. Lista nézet

Ha az adatokat megfelelően adta meg a belépésnél, akkor a termékek listájának oldala jelenik meg, ahol az árucikkek táblázatosan felsorolásra kerültek. A táblázat tetejére került egy link, amely az új termék felvitele, illetve a megrendelések oldalra vezet tovább.

1.8.3. Megrendelések

Ezen az oldalon az adminisztrátor a megrendeléseket kezelheti. Itt is egy táblázatos nézetben kerülnek megjelenítésre az adatok. Itt láthatóak a megrendelő és a megrendelés adatai, mint például a felhasználónév, termék név, és a rendelt termék darabszáma. Van egy rendelés követő funkció is, amelyet az adminisztrátor állíthat be. Négy státusza van egy megrendelésnek: megrendelve, feldolgozás alatt, futárszolgálatnál, illetve kézbesítve. Itt az adminisztrátor gombok segítségével állíthatja be, mely állomásnál tart a csomag, amely a felhasználó felületen is megjelenik, és az adatbázisban is frissül.

Megrendelések			
Termék táblázat		Kijelölés	
Felhasználó neve	Termék neve	Rendelés (DB)	Státusz
Szabó Máté	Okuma Custom Black Ceymar River Feeder	2 db	<div><div>Megrendelve</div><div>Feldolgozás alatt</div><div>Futárszolgálatnál</div><div>Kézbesítve</div></div>
Szabó Máté	Dóme TEAM FEEDER Carp Fighter	2 db	<div><div>Megrendelve</div><div>Feldolgozás alatt</div><div>Futárszolgálatnál</div><div>Kézbesítve</div></div>

12) ábra Megrendelések nyomon követése

1.8.4. Felvitel

Az "új termék felvitele" gomb egy új oldalra navigál, ahol a termék adatainak megadása történik. Miután minden szükséges adat felvitelre került, a megerősítés után a termék beszúrássra kerül az adatbázisba, ahonnan pedig kiolvasásra a webshopba.

Új árú felvitele

A *-al jelölt mezőket kötelező kitölteni!

*Fotó feltöltése	Fájl kiválasztása Nincs fájl kiválasztva
*Vonalkód:	00000000
*Név:	Név
*Kategória:	Kategória választó
*Felvitel dátuma:	éééé. hh. nn. <input type="checkbox"/>
*Ár:	Ár
*Darab:	1
*Leírás:	Adja meg a leírást (200 karakter)

13) ábra Új termék felvitele űrlap

1.8.5. Módosítás

A módosítás gombra kattintva, a már meglévő termékek adatainak módosítására van lehetőség. A már felvitt adatok automatikusan megjelenítésre kerülnek az űrlap mezőiben, így elég csak azokat az adatokat módosítani, amelyeket szeretnénk, és nem kell újra begépelni minden egyes részletet. Ez a funkció nagyon fontos, hiszen előfordulhat, hogy valamit elgépel az adminisztrátor a felvitel közben, vagy árat kell módosítani. Így nem kell a termék összes adatát törölni, elég a hibás adatot javítani.

Módosítás

A *-al jelölt mezőket kötelező kitölteni!

Fotó feltöltése	Fájl kiválasztása Nincs fájl kiválasztva
*Vonalkód:	00000000
*Név:	Okuma Ceymar River Feeder
*Kategória:	Kategória választó
*Felvitel dátuma:	2023. 03. 03. <input type="checkbox"/>
*Ár:	27000
*Darab:	100
*Leírás:	Pontyos és egyéb békéshalas horgászatra kifejlesztett Okuma Ceymar feeder botok élménygazdag horgászatot eredményeznek. Kirobbanó farasztási erővel rendelkeznek.

14) ábra Módosítás űrlapja

1.8.6. Törlés

A törlés gombra kattintva a termékek törlésére van lehetőség. A termékek az adatbázisból, és ezzel egyidejűleg a webshopból is törlésre kerülnek.

2. Fejlesztői dokumentáció

Munkánk során egy 7 táblából álló adatbázist, 1 HTML fájlt, 54 PHP fájlt, 7 JavaScript fájlt, és 7 CSS fájlt készítettünk el. Ezért azokat a részleteket emeltük ki a programozás folyamatából, amelyek megoldására büszkék vagyunk.

2.1. Adatbázis tervezés

A megfelelő tervezés létfontosságú feladat az adatbázissal végzett munka céljának elérésében. A gondos tervezés hosszú időt vett igénybe számunkra, sokszor újra kellett gondolni, mit hogyan szeretnék megvalósítani. Fontos, hogy az adatbázis tervezés alapelveit figyelembe vegyük tervezés során, hiszen ilyen módon jó eséllyel az igényeinknek megfelelő adatbázis fog születni, amelyben a szükséges módosítások könnyedén elvégezhetők, a különböző lekérdezések pedig korrekt eredményekkel térnek vissza.

Figyeltünk arra, hogy redundáns adat ne szerepeljen az adatbázisunkban, hiszen fölöslegesen helyet foglal és növeli a hibalehetőségek számát, illetve az ellentmondások előfordulásának esélyét. Ezért úgy alakítottuk ki, hogy az adatbázis tartalma konzisztens legyen és megfeleljen a hivatkozási integritás szabályainak.

2.1.1. Adatbázis célja

Az adatbázis célja, hogy alkalmas legyen egy webáruház adatainak tárolására, mint például regisztrált felhasználók adatai, felvitt termékek adatai, illetve a megrendelésekhez fűződő egyéb adatok.

2.1.2. Egyedek meghatározása

Az egyedek meghatározása az első dolgunk volt az adatbázis megtervezése során, hiszen a tervezés elején nem rendelkezünk több információval, csak egy kezdetleges képünk volt az adatbázisról. Összesen 4 egyedet különítettünk el: termék, személyek, kategóriák, alkategóriák.

1) Termék

A termék egyednél határoztuk meg azokat a termékeket, amelyeket a webshopban árusítunk. Meghatároztuk egy terméknek, hogy milyen kezdő tulajdonságokkal rendelkezzen, ilyen például, a termék vonalkódja, ára, neve, a raktáron lévő termékek darabszáma, az adott terméket mikor vettük fel az adatbázisba, illetve egy rövid leírás és egy kép is tartozik egy-egy termékhez.

2) Személy

A személy egyednél több elképzelésünk is született a tervezés során. Elsősorban 2 csoportot különítünk el, vannak a felhasználók és az adminok. Az admin szintű felhasználó adminisztrációs feladatokat lát el a weboldalon. Az adminok tudnak termékeket felvinni az adatbázisba, illetve a webáruházba, a rendeléseket intézni és a felhasználó adatait módosítani, ha szükséges. Úgy döntöttünk, hogy nem bontjuk két táblára az admint és a felhasználót, hanem az egyednek megadtunk egy olyan tulajdonságot, ami megmondja, hogy az adott személy milyen jogosultsággal rendelkezik. Ezen túl több más leíró mezője is van: név, felhasználónév, jelszó, lakcím, e-mail cím, telefonszám, jogosultság.

3) Kategória

Létrehoztunk 4 fő kategóriát, hiszen a valós üzletünkben is e szerint a 4 fő termékkategória szerint dolgozunk. Ennek az egyednek csak egyetlen leíró mezője van, hiszen itt csak a kategóriák neveit tároljuk el majd a továbbiakban.

4) Alkategória

A kategóriákat további alkategóriákra bontottuk fel, amely megkönnyíti majd a termékek közötti keresést.

2.1.3. Kapcsolatok

Az adatbázisunk kétféle kapcsolatot kezel, 1:N, illetve N:M. 1:N kapcsolat esetén az egyik egyedhez több másik egyed tudunk társítani, de a másik egyed adott példányához mindössze egyet társítunk. N:M kapcsolatnál egy egyed példánya több másikkal áll relációban, és ez fordítva is igaz. Ezt a kapcsolatot több 1:N kapcsolatra bontottuk.

- 1) **Alkategóriák – Termék:** A két egyed között **1:N** kapcsolat áll fenn, hiszen egy termékhez csak egyetlen al kategória társul és egy al kategóriában több termék is megtalálható.
- 2) **Kategóriák – Alkategóriák:** Itt is szintén **1:N** kapcsolat figyelhető meg. Egy kategóriába több al kategória tartozik, továbbá egy al kategória csak egy kategóriába sorolható.
- 3) **Termék – Személyek:** A termék és a személyek között **N:M** kapcsolat van. Egy személy több termékkel áll kapcsolatban, hiszen egy személy több terméket is

megvásárolhat, illetve egy termék több személlyel áll kapcsolatban, hiszen egy termékből több személy is vásárolhat. A termék és a személyek N:M-es kapcsolatát kettő 1:N kapcsolatra kellett szétbontanunk, ezt a megrendelés kapcsolótáblával tettük lehetővé.

- 4) **Termék – Megrendelés:** A termék és a megrendelés között 1:N kapcsolat van. Egy termék egy adott megrendeléshez tartozik, viszont egy megrendeléshez tartozhat több termék is.
- 5) **Személyek – Megrendelés:** A személyek és a megrendelés között 1:N kapcsolat van. Egy személyhez tartozhat több megrendelés, viszont egy adott megrendelés csak egy személyhez tartozik.
- 6) **Rendelés_állapot – Megrendelés:** A rendelés_állapot és a megrendelés között 1:N kapcsolat van, hiszen egy állapothoz tartozhat több megrendelés, viszont egy megrendelés csak egy állapotba tartozhat.
- 7) **Települések – Személyek:** A települések és a személyek között 1:N kapcsolat van, hiszen egy településhez tartozhat több személy, viszont egy személyhez csak egy települést rendelhetünk.

2.1.4. Táblák meghatározása

Ezen a ponton már ismerjük az egyedeket, illetve a kapcsolatokat közöttük. Így már meg tudjuk határozni ténylegesen a táblákat.












1) Termék tábla

Leírás: Itt tároljuk azokat a termékeket, amelyeket a webáruházunkban árulunk. Ezen a ponton már tudjuk a tulajdonságok pontos szerepét. Egy webáruházban a termékeken van a legfőbb szerep, ezért próbáltuk részletesen kidolgozni annak attribútumait. A vonalkód szerepét az adatbázis továbbfejlesztése menüpontban fejtettük ki.

Tábla tulajdonságai:

Név	Leírás	Típus
id	azonosító tulajdonság, elsődleges kulcs	int (11)
vonalkod	a termék vonalkódja	varchar (50)
ar	a termék teljes ára	int (11)
darab	a raktáron lévő termékek darabszáma	int (7)
foto	a termékről egy kép, a képről egy elérési útvonalat tárolunk, ezért lesz varchar a típus	varchar (255)
nev	a termék neve	varchar (50)

leiras	a termékről egy rövid leírás	varchar (255)
felv_datum	a termék felvételi dátuma, amikor az adatbázisba került	date
alkategoria_id	idegen kulcs, az alkategoriák elsődleges kulcsára hivatkozik	int (11)

										id	vonalkod	felvdatum	ar	darab	foto	nev	alkategoria_id	leiras
<input type="checkbox"/>				83	00000000	2023-03-03	27000	18	1677840738.jpeg	Okuma Ceymar River Feeder	1	Pontyos és egyéb békéshalas horgászatra kifejleszt...						
<input type="checkbox"/>				84	00000000	2023-03-03	29100	11	1677844206.jpeg	Okuma Ceymar River Plus	1	Pontyos és egyéb békéshalas horgászatra kifejleszt...						
<input type="checkbox"/>				85	00000000	0000-00-00	32000	75	1677844292.jpeg	Okuma Ceymar Black	1	Pontyos és egyéb békéshalas horgászatra kifejleszt...						
<input type="checkbox"/>				86	00000000	2023-03-03	41000	97	1677844365.jpeg	Okuma Custom Black Ceymar River Feeder	1	Pontyos és egyéb békéshalas horgászatra kifejleszt...						













15) ábra Termék tábla

2) Kategória tábla

Leírás: A webáruház termékei 4 fő kategóriára sorolhatóak. Ebben a táblában ezeket a kategóriákat tároljuk. Ezekbe a fő kategóriákba soroljuk az alkategóriákat.

Tábla tulajdonságai:

Név	Leírás	Típus
kategoria_id	azonosító tulajdonság, elsődleges kulcs	int (11)
kategoria_nev	a kategória neve	varchar (50)

				kategoria_id	kategoria_nev
<input type="checkbox"/>				1	horgász
<input type="checkbox"/>				2	díszállat
<input type="checkbox"/>				3	háztartás
<input type="checkbox"/>				4	fatermék

16) ábra Kategória tábla

3) Alkategóriák tábla

Leírás: A termékeket fő kategóriákra , majd tovább, alkategóriákba soroljuk. Ebben a táblában az alkategória neveket tároljuk el, illetve egy idegen kulcs is szerepel, amely segítségével a kategoriak táblával kötjük össze.

Tábla tulajdonságai:

Név	Leírás	Típus
alkategoriak_id	azonosító tulajdonság, elsődleges kulcs	int (11)
alkategoriak_nev	az alkategória neve	varchar (50)
kategoria_id	idegen kulcs, a kategoriak elsődleges kulcsára hivatkozik.	int (11)

←T→	alkategoria_id	alkategoria_nev	kategoria_id
<input type="checkbox"/> Módosítás Másolás Törlés	1	Horgászbót	1
<input type="checkbox"/> Módosítás Másolás Törlés	2	Orsó	1
<input type="checkbox"/> Módosítás Másolás Törlés	3	Etetőanyag	1
<input type="checkbox"/> Módosítás Másolás Törlés	4	Horgász kiegészítők	1
<input type="checkbox"/> Módosítás Másolás Törlés	5	Horgász ruházat	1
<input type="checkbox"/> Módosítás Másolás Törlés	6	Haltáp	2
<input type="checkbox"/> Módosítás Másolás Törlés	7	Rágcsálótáp	2
<input type="checkbox"/> Módosítás Másolás Törlés	8	Díszállat kiegészítők	2
<input type="checkbox"/> Módosítás Másolás Törlés	9	Alom	2
<input type="checkbox"/> Módosítás Másolás Törlés	10	Levegőztető, szűrő, melegítő	2
<input type="checkbox"/> Módosítás Másolás Törlés	11	Konyha	3
<input type="checkbox"/> Módosítás Másolás Törlés	12	Fürdőszoba	3
<input type="checkbox"/> Módosítás Másolás Törlés	13	Játék	3
<input type="checkbox"/> Módosítás Másolás Törlés	14	Dísz	3
<input type="checkbox"/> Módosítás Másolás Törlés	15	Ajándék	3
<input type="checkbox"/> Módosítás Másolás Törlés	16	Madáretető	4
<input type="checkbox"/> Módosítás Másolás Törlés	17	Odú	4
<input type="checkbox"/> Módosítás Másolás Törlés	18	Kulcstartó	4
<input type="checkbox"/> Módosítás Másolás Törlés	19	Kenyértartó	4
<input type="checkbox"/> Módosítás Másolás Törlés	20	Kosár	4













17) ábra Alkategóriák tábla

4) Megrendeles tábla

Leírás: A megrendelés táblában tároljuk el a megrendeléseket. A termékeket soronként tároljuk el. Ha a felhasználó egyszerre többféle terméket is rendel, akkor az több sorba tárolódik el az adatbázisban.

Tábla tulajdonságai:

Név	Leírás	Típus
id	azonosító tulajdonság, elsődleges kulcs	int (11)
megrendeles_datuma	a megrendelés leadásakor elmentett dátum	date
kezesites_datuma	a termék kézbesítése után mentett dátum	date
rendelt_darab	a rendelt termékek darabszáma	int (5)
szemelyek_id	idegen kulcs, a személyek elsődleges kulcsára hivatkozik	int (11)
rendeles_allapot_id	idegen kulcs, a rendeles_allapot elsődleges kulcsára hivatkozik	int (11)
rendeles_azonosito	egyedi azonosító a rendelt termékeknek	varchar (40)

<div>← T →</div>			id	megrendeles_datuma	kezesites_datum	szemelyek_id	
<input type="checkbox"/>	 Módosítás	 Másolás	 Törés	89	2023-04-17 11:50:30	2023-04-17 11:50:30	5
<input type="checkbox"/>	 Módosítás	 Másolás	 Törés	90	2023-04-17 14:40:19	2023-04-17 14:40:19	4
<input type="checkbox"/>	 Módosítás	 Másolás	 Törés	93	2023-04-24 12:48:25	2023-04-24 12:48:25	2
<input type="checkbox"/>	 Módosítás	 Másolás	 Törés	94	2023-04-24 12:48:46	2023-04-24 12:48:46	4

18) ábra Megrendeles tábla I. része

84	1	3	449670eee7dca9cd166a
110	2	10	1ec941c694fe1e3d3017
97	3	1	1ec941c694fe1e3d301722
96	2	5	1ec9413c694fe1e3d3017

19) ábra Megrendeles tábla II. része










5) Szemelyek tábla

Leírás: Itt tároljuk a személyeket. A személyeket két csoportra bontjuk, vannak a felhasználó szintű személyek, illetve az admin szintű személyek.

Tábla tulajdonságai:

Név	Leírás	Típus
id	azonosító tulajdonság, elsődleges kulcs	int (11)
nev	a személy neve	varchar (255)
felh_nev	a személy egyedi felhasználóneve	varchar (255)
email	a személy e-mail címe	varchar (255)

jelszo	a személy jelszava	varchar (255)
jog	a személy beállított joga, ez adja meg milyen módon használja a weboldalt, a regisztrálásnál automatikusan felhasználói jogot állít be a rendszer, admin jogot csak az ügyintézői felületen lehet beállítani.	varchar (255)
tel	a személy telefonszáma	varchar (255)
szallitasi_cim	a személy címe, utca, házszám	varchar (255)
irsz	idegen kulcs, a település elsődleges kulcsára hivatkozik	varchar (120)

				id	nev	felh_nev	email
<input type="checkbox"/>	 Módosítás	 Másolás	 Törölés	1	teszt	teszt	teszt@teszt.hu
<input type="checkbox"/>	 Módosítás	 Másolás	 Törölés	2	Szabó Máté	Szabo12	szabomate@gmail.com
<input type="checkbox"/>	 Módosítás	 Másolás	 Törölés	3	Kordics Balázs	Balazs	kordics.b@gmail.com

20) ábra Szemelyek tábla I. része

jelszo	jog	tel	szallitasi_cim	irsz
7110eda4d09e062aa5e4a390b0a572ac0d2c0220	user	00000000000	Valami utca 2	502
7110eda4d09e062aa5e4a390b0a572ac0d2c0220	user	06308601620	Petőfi út 2	502
7110eda4d09e062aa5e4a390b0a572ac0d2c0220	user	06308590860	Jószív utca 29	502

21) ábra Szemelyek tábla II. része

6) Rendeles_allapot

Leírás: A rendeles_allapot táblában csak egy id és egy leíró mező foglal helyet. A leíró mezőben csak a rendelés státuszának a nevét tároljuk, mint például: megrendelve, feldolgozás alatt, futárszolgálatnál, kézbesítve.

Tábla tulajdonságai:

Név	Leírás	Típus
id	azonosító tulajdonság, elsődleges kulcs	int (11)
allapot_nev	a rendelés státusza: megrendelve, feldolgozás alatt, futárszolgálatnál, kézbesítve	varchar (255)

				▼ állapot_id	állapot_nev
<input type="checkbox"/>	 Módosítás	 Másolás	 Törlés	1	Megrendelve
<input type="checkbox"/>	 Módosítás	 Másolás	 Törlés	2	Feldolgozás alatt
<input type="checkbox"/>	 Módosítás	 Másolás	 Törlés	3	Futárszolgálatnál
<input type="checkbox"/>	 Módosítás	 Másolás	 Törlés	4	Kézbesítve

22) ábra Rendeles_allapot tábla

23) Telepules tábla

Leírás: A telepules táblában tároljuk az összes magyar települést irányítószámmal együtt, illetve a megyéket kód alapján. Az telepules táblához valós adatokat használtunk, ehhez http://webdraft.eu/orszagok_varosok/ oldalon található adatbázist hasznosítottuk.

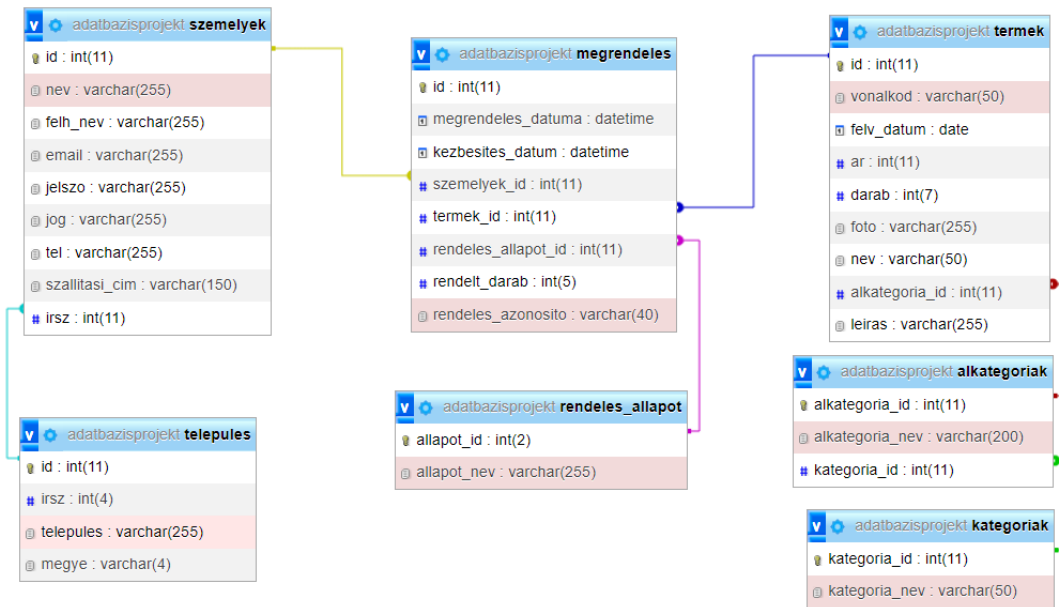
Tábla tulajdonságai:

Név	Leírás	Típus
id	azonosító tulajdonság, elsődleges kulcs	int (11)
irsz	a települések irányítószáma	int(4)
telepules	település neve	varchar(255)
megye	megye kód, amiről a megye azonosítható	varchar(4)

				▼ id	irsz	telepules	megye
<input type="checkbox"/>	 Módosítás	 Másolás	 Törlés	1	1011	Budapest I. kerület	BU
<input type="checkbox"/>	 Módosítás	 Másolás	 Törlés	2	1012	Budapest I. kerület	BU
<input type="checkbox"/>	 Módosítás	 Másolás	 Törlés	3	1013	Budapest I. kerület	BU
<input type="checkbox"/>	 Módosítás	 Másolás	 Törlés	4	1014	Budapest I. kerület	BU
<input type="checkbox"/>	 Módosítás	 Másolás	 Törlés	5	1015	Budapest I. kerület	BU

23) ábra Telepulesek tábla

2.1.5 Adatbázis diagram



24) ábra Adatbázis diagram

2.2. Kódmagyarázat

A kezdő lapunk main.php néven került elmentésre. Azokon az oldalakon, ahol PHP és HTML kódok is szerepelnek egyaránt, ahol csak lehetséges volt igyekeztünk szétválasztani a két technológiát a könnyebb átláthatóság és a későbbi karbantartás végett. Így PHP kódokkal kezdődik az oldal, majd ezt követően került beillesztésre a HTML. A HTML kódoláson belül csak a PHP kimenetet jelenítettük meg.

2.2.1. Kezdő oldal

Az oldal tetején megjelenő menü elkészítéséhez a Bootstrap keretrendszerét használtuk. Mellette a jobb oldalon, ha a felhasználó nem lépett be, akkor a belépés és regisztráció menüpontok jelennek meg. Ha a felhasználó már belépett, akkor a kilépés felirat jelenik meg. A menüt „mainnav.php” oldalra szerveztük ki, azért, ha valahol még szükségünk lenne a navigációra, ne kelljen ismételtten leírni a sorokat, hanem egyszerűen egy „include()”-al csatoljuk, amely ha nem is éri el a fájlt, attól az oldal még be fog tölteni.

2.2.2. Regisztráció

```
session_start();
require("kapcsolat/kapcs.php");
```

Először elindítjuk a munkamenetet a „session_start()” függvénnyel, majd csatlakozunk az adatbázishoz a „kapcs.php” fájl segítségével, amelyben szerepel az adatbázis kapcsolat létrehozása.

```

$nev = mysqli_real_escape_string($dbconn, $_POST['nev']);
$felhnev = mysqli_real_escape_string($dbconn, $_POST['felhnev']);
$email = mysqli_real_escape_string($dbconn, $_POST['email']);
$jelszo = sha1($_POST['jelszo']);
$jelszoujra = sha1($_POST['jelszoujra']);
$iranyitoszam = $_POST['iranyitoszam'];
$szallitasicim = $_POST['szallitasi_cim'];
$tel$_POST['tel'];
$select = "SELECT * FROM `szemelyek`
          WHERE email = '$email' && jelszo = '$jelszo'";

```

Ha a felhasználó elküldi a regisztrációs űrlapot az "ok" gomb megnyomásával, a kód először megszűri az összes bemeneti adatot, amelyet a „mysqli_real_escape_string()” függvénnyel végez, hogy megakadályozza a „SQL injection” támadásokat. A jelszót „SHA1” algoritmus segítségével kódoljuk biztonsági okokból.

```

$result = mysqli_query($dbconn, $select);
if(mysqli_num_rows($result) > 0)
{
    $error[] = 'Ez a felhasználó már létezik!';
}
else
{
    if($jelszo != $jelszoujra)
    {
        $error[] = 'A jelszavak nem egyeznek!';
    }
}

```

A kód ellenőrzi, hogy van-e már ilyen felhasználói fiók az adatbázisban az email cím és a jelszó ellenőrzésével. Ha a felhasználó már létezik, akkor hibaüzenet jelenik meg. Ellenkező esetben a kód ellenőrzi, hogy a két jelszó megegyezik-e.

```

else {
    $insert = "INSERT INTO `szemelyek`(`nev`, `felhnev`, `email`, `jelszo`,
    `iranyitoszam`, `szallitasi_cim`, `tel`) VALUES
    ('$nev','$felhnev','$email','$jelszo','$iranyitoszam' , '$szallitasicim','$tel')";
    mysqli_query($dbconn,$insert);
    header('Location:belep.php'); } }

```

Ha a két jelszó megegyezik, akkor az összes adatot az adatbázisba helyezi az „INSERT INTO” paranccsal, majd átirányítja a felhasználót a bejelentkezési oldalra a „header()” függvénnyel.

```

$telepulesListaQuery = "
SELECT id, telepules, irsz
FROM telepules
";
$telepulesLista = mysqli_query($dbconn, $telepulesListaQuery);
$telepulesBeolvasas = mysqli_fetch_all($telepulesLista, MYSQLI_ASSOC)
$options = "";

```

```
foreach ($telepulesBeolvasas as $lista) {
    $options .= "<option value='{ $lista["id"] }'>{ $lista["irsz"] } -
{ $lista["telepules"] }</option>";}
```

A település adatait külön kellett kezelnünk, ezért létrehoztunk egy külön táblát is, amelyben a település nevét, irányítószámát tároljuk. Először lekérjük az adatokat a „telepules” táblából. Ezután az eredményeket egy ciklusban olvastuk be és a „\$options” változóhoz hozzáadjuk az „HTML <option>” elemeket a „\$telepulesBeolvasas” tömb alapján. Minden opció értéke az „id” mező, míg a szöveges tartalom a „irsz” és „telepules” mezők együttese. Az adatokkal egy legördülő listát hozunk létre. Az „\$options” változó a legördülő lista összes elemét tartalmazza, majd ezeket jelenítjük meg a felhasználó számára.

2.2.3. Belépés

```
if(isset($_POST['ok']))
{
    $email = mysqli_real_escape_string($dbconn, $_POST['email']);
    $pass = sha1($_POST['jelszo']);
    $select = "SELECT * FROM `szemelyek`
        WHERE email = '$email' && jelszo = '$pass'";
    $result = mysqli_query($dbconn, $select);
    if(mysqli_num_rows($result) > 0)
    {
        $row = mysqli_fetch_assoc($result);
        if($row['jog'] == 'admin')
        {
            $_SESSION['felhnev'] = $row['nev'];
            $_SESSION['id'] = $row['id'];
            $_SESSION['belepett'] = true;
            header('Location:admin/adminlist.php');
        }
        elseif($row['jog'] == 'user')
        {
            $_SESSION['felhnev'] = $row['felhnev'];
            $_SESSION['id'] = $row['id'];
            $_SESSION['belepett'] = true;

            $id = $row['id'];
            header('Location:user/webshop.php');
        }
    }
}
```

Ha a felhasználó rákattint a bejelentkezés gombra, akkor a kód ellenőrzi, hogy van-e olyan felhasználó az adatbázisban, akinek az e-mail címe és jelszava megegyezik a felhasználó által megadott adatokkal. Ha igen, akkor beállítja a felhasználó nevét, azonosítóját és jogosultságát a munkamenet változóiban („\$_SESSION”), majd

átirányítja a felhasználót az adminisztrátori vagy a felhasználói oldalra a „header()” függvény segítségével, attól függően, hogy az adatbázisban milyen jog van elmentve.

```
else{  
    $error[] = "Helytelen e-mail címet, vagy jelszót adott meg!";}  
Ha a bejelentkezés sikertelen, akkor egy hibaüzenetet jelenít meg.
```

2.2.4. Felhasználói adatok módosítása

A módosítás oldal nagyon hasonló a regisztrációs oldalhoz, mind kinézetben, mind működésben, azzal a kivétellel, hogy itt a felhasználó már a meglévő adatait tudja változtatni. A fő különbség, hogy a felhasználó adatait megjelenítjük a módosítás „form”-ban, ahol a felhasználó könnyedén módosíthatja adatait.

2.2.5. Kilépés

```
session_start();  
session_destroy();  
header("Location: ../belep.php");
```

A "session_start()" függvény indítja el a munkamenetet, amit a "session_destroy()" függvény zár le. Ezután a "header()" függvénnyel átirányítja a felhasználót egy másik oldalra, amely a "../belep.php" elérési útvonalon található. Ez a kódrészlet biztonságosan lezárja a felhasználó munkamenetét és átirányítja őt a bejelentkezési oldalra.

2.2.6. Kosár

A kosár működését JavaScript-tel oldottuk meg. A „script” már akkor kezd lefutni, amikor az oldal betöltődik („window.onload”).

Első lépésként konstans változóba eltároltuk a különböző elemeket, mint például a kosár, vagy a kosár bezárása ikon. Minden elemet ellenőriztünk a böngésző konzoljába kiírva, hogy sikeresen eltároltuk-e a változóba.

```
const kosarikon = document.querySelector('.kosarikon');  
//console.log(kosarikon);
```

A kosár ikonhoz egy eseményfigyelőt adtunk hozzá, amely két paramétert vár. Az első paramétere az, hogy mit figyeljen, a második pedig, hogy mi történjen, amikor az adott esemény bekövetkezik:

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

Ekkor a „cartBox” elemünkhöz hozzáadásra kerül egy „active” nevű osztály, amelynek a megjelenését a CSS-ben határoztunk meg. A kosár tartalma egy felugró ablakban jelenik meg, ha a felhasználó a kosár ikonra kattint. Hasonlóképpen oldottuk meg ennek a felugró ablaknak a bezárását is, ha az „X”-re kattint a felhasználó, akkor a „cartBox”-ról eltávolításra kerül az „active” osztály.

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

A következőkben eltároltuk a kosárgombokat, amelyek a webshop weboldalán a termékek alatt megjelentek.

```
const kosarhozgomb = document.getElementsByClassName('kosarhoz')  
console.log("Ez a kosár gomb",kosarhozgomb);
```

Deklaráltunk egy új változót „cuccok” néven, amely értéként egy üres tömböt kapott, abból a célból, hogy a későbbiek folyamán ebbe tároljuk el a megrendelt termékeket.

```
let cuccok = [];
```

A következő feladatunk az volt, hogy meghatározzuk, mely kosárgombra történt a kattintás, a következő „for” ciklus segítségével, amely végigiterál a kosárgombokon.

```
for (let i = 0; i < kosarhozgomb.length; i++) {  
    kosarhozgomb[i].addEventListener("click",function(e){
```

Egy eseményfigyelőt adtunk hozzá ehhez a gombhoz, amelynek két paramétere van. Egy kattintást vár, illetve a kattintás után meghívja a függvényt.

Ezen a ponton akadályoztuk meg, hogy több terméket is a kosárba tudjon helyezni a felhasználó, mint amennyi raktáron van. Ehhez eltároltuk egy változóba a raktáron lévő termékek számát. Majd ezt követően megvizsgáltuk, ha a kosárba helyezett termékek száma több, mint a raktáron lévő termékek száma, akkor egy felugró ablak figyelmezteti erre a felhasználót, de abban az esetben is, ha 0 darabot próbálna a kosárba helyezni.

```
if(typeof(Storage) !== 'undefined'){  
    let cucc = {  
        id: i + 1,  
        name:  
e.target.parentElement.parentElement.parentElement.children[0].children[1].innerHTML,  
L,
```

```

        no:
parseInt(e.target.parentElement.parentElement.children[1].innerHTML),
        price:
parseInt(e.target.parentElement.parentElement.children[2].innerHTML),
        darab: parseInt(e.target.parentElement.children[0].value),
        termék_id: parseInt(e.target.dataset.product_id),
    };

```

Egy elágazásban megvizsgáljuk, hogy a tároló üres-e, vagy sem. Ha a tároló nem üres, akkor beolvassa a „cucc” változóba. A termék adatait úgy olvassuk ki, hogy a gombnak van egy „div” szülőeleme, azon belül pedig a termék adatai. A szülőelemen belül vannak a gyermek elemek, amelyek index alapon leválogatásra kerülnek. Tervezés során az indexekre kiemelt figyelmet kellett szánnunk, hiszen ha több gyermek elemet adunk hozzá, akkor az indexek értelemszerűen változnak.

```

if (JSON.parse(localStorage.getItem('cuccok')) === null){
    cuccok.push(cucc);
}

```

Egy elágazásban megvizsgáljuk, hogy a lokális tárolóban van-e adat, ha az értéke 0, tehát üres, akkor hozzáadjuk a cuccok tömbhöz a „push” metódus használatával.

```

localStorage.setItem("cuccok",JSON.stringify(cuccok));
window.location.reload();

```

A terméket elmentjük a „setItem”-el a lokális tárolóban. A mentéshez két paramétert kell megadnunk. Az egyik paraméter egy kulcs, amelyre később lehet hivatkozni. A másik paraméter egy érték.

```

} else {
    const localcuccok = JSON.parse(localStorage.getItem("cuccok"));
    console.log("localcuccok", localcuccok);
    localcuccok.map(data => {
        if (cucc.id === data.id) {
            cucc.darab += data.darab;
        } else {
            cuccok.push(data);
        }
    });
    cuccok.push(cucc);
    localStorage.setItem('cuccok', JSON.stringify(cuccok));
    window.location.reload();
}

```

Az „else” ágban meg kellett oldanunk az is, amikor a tároló nem üres, hiszen az az életszerű, hogy a vásárló több terméket tegyen a kosárba, akkor is, ha nem egy időben

történik a vásárlás. A „map” függvény segítségével nyertük ki az adatokat, ami a lokális tárolóban mentett adatokat kirajzolta, majd az oldalt újratöltöttük, hogy lefrissüljön a kirajzolt adat.

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

Ha a lokális tároló megtelik, meghibásodik valamilyen okból kifolyólag, vagy felhasználó böngészője nem ismeri fel, akkor egy hibaüzenet jelenik meg.

```
const kosarikonP = document.querySelector('.kosarikon p');  
    console.log("kosarikonP",kosarikonP);  
    let no = 0;  
    JSON.parse(localStorage.getItem('cuccok')).map(data =>{  
        no = no + data.darab;  
    });
```

Az első sorban a „querySelector()” metódussal kiválasztottuk a „.kosarikon p” osztályú HTML elemet, amely a kosárban található tételek számát mutatja. Ezután létrehoztunk egy „no” nevű változót, amely az összes, a kosárban található tétel számát fogja tárolni. A lokális tárolóban tárolt termékek listáját a „JSON.parse()” metódus segítségével nyitottuk meg, majd az „Array.map()” függvény segítségével végigiteráltunk a terméklistán. Az iteráció során hozzáadjuk a termékek darabszámát a „no” változóhoz.

```
    kosarikonP.innerHTML = no;  
    //Kosár feltöltése.  
    const cardBoxTable = document.querySelector('table');  
    console.log(cardBoxTable);  
    let tableData = "";  
    tableData += '<tr><th>Cikk szám</th><th>Termék  
neve</th><th>Darab</th><th>Termék ár</th><th>Darab törlés</th><th>Kategoria  
törlése</th></tr>';
```

Kiválasztottuk a „kosarikonP” osztályú elemet, majd beállítottuk az „innerHTML” értékét a „no” változó értékére. Ezután létrehozunk egy „tableData” változót, majd hozzáadtunk egy táblázat fejléc sort.

```
if(JSON.parse(localStorage.getItem('cuccok')) [0] === null){  
    //Ha nincs hozzá adva termék akkor legyen egy üres sor.  
    tableData += '<tr><td colspan="5"></td></tr>';
```


Ezután megvizsgáljuk, hogy a lokális tárolóban tárolt cuccok nevű tömb első eleme „null”-e, ha igen akkor egy üres sort ad hozzá a táblázathoz.

```
    }else{JSON.parse(localStorage.getItem('cuccok')).map(data => {  
        tableData += '<tr  
id="sor_'+data.id+'"><td>'+data.id+'</td><td>'+data.name+'</td><td  
class="darab">'+data.darab+'</td><td>'+data.price* data.darab+'</td><td><input  
name="dbTorles" id="productNumber_'+data.termek_id+'"  
onchange=deleteProductAmount('+data.termek_id+')></td><td><a href="#"  
onclick=Delete(this);> Törlés</a></td></tr>';})
```

Ellenkező esetben a "cuccok" tömbön végigmegy az „Array.map()” segítségével, majd hozzáad egy sort minden elemhez a "tableData" változóhoz.

```
    }let sum = 0;  
  
    JSON.parse(localStorage.getItem('cuccok')).map(data =>{  
  
        sum += data.darab * data.price;})
```

Megszámoljuk a kosárban lévő elemek összértékét azzal, hogy összeszorozzuk azoknak a mennyiségét és árát, majd hozzáadtuk az eredményt a "sum" változóhoz.

```
        tableData += '<tr><td colspan="3" class="jobb"><a  
href="megrendeles.php">Megrendelés</a></td><td>'+ sum +'</td><td colspan="2"><a  
href="#" onclick="deleteall(this)">Összes törlése</a></td></tr>';
```

Végül hozzáadtunk egy utolsó sort a "tableData" változóhoz, amely hivatkozásokat tartalmaz a rendelés teljesítéséhez, és az összes elem törléséhez.

```
cardBoxTable.innerHTML =tableData;
```

Ezután beállítjuk a "cardBoxTable" elem „innerHTML” értékét a "tableData" értékére.

```
function Delete(elem){  
    let cuccok = [];  
    JSON.parse(localStorage.getItem('cuccok')).map(data =>{  
        if(data.id  
elem.parentElement.parentElement.children[0].textContent){  
            cuccok.push(data);  
            let deletedid = data.id;  
            console.log(deletedid);  
        }  
    });  
    localStorage.setItem("cuccok", JSON.stringify(cuccok));  
    !=
```

```
window.location.reload();
```

A „Delete” függvény egy elem törlésére szolgál a kosárból. Az elemet egy paraméterként kapott változóval határoztuk meg. Az eltávolítandó elemet eltávolítjuk a „cuccok” nevű változóból, ami egy tömb. Ezután a módosított tömböt visszatároljuk a lokális tárolóba, így a törlés megvalósul.

```
function deleteall(){
  let cuccok = [];
  JSON.parse(localStorage.getItem('cuccok')).map(data =>{
    cuccok.splice(data);
  });
  localStorage.setItem("cuccok", JSON.stringify(cuccok));
  window.location.reload();}
```

A „deleteall” függvény az összes elem eltávolítására szolgál a kosárból, amely lényegében ugyanúgy működik, mint az egyesével törlés.

```
function megrendeles(){
  window.location.href = "megrendeles.php";}
```

Itt a felhasználót átirányítjuk a „megrendeles.php” oldalra, ahol véglegesítheti a megrendelését.

```
function deleteProductAmount(termek_id) {
  let product = localStorage.getItem("cuccok");
  $.ajax({
    method: "POST",
    url: "../api.php",
    dataType: "JSON",
    data: { c: "deleteProductAmount", termék_id: termék_id, darab_szam:
$("#productNumber_" + termék_id).val(), product: product },
    success: function (result) {
      if (result.deleteline) {
        $("#sor_" + result.lineid).remove();
      } else {
        $("#sor_" + result.lineid + " .darab").text(result.newAmount);
      }
      localStorage.setItem("cuccok", JSON.stringify(result.products));
    }
  })
}
```

A „deleteProductAmount” is elemek törlését valósítja meg, de itt lehetősége van a felhasználónak, hogy bevigye, pontosan hány terméket szeretne törölni. Ha a

felhasználó nagyobb számot ír be, mint a rendelt mennyiség, akkor az egész sor törlődik. A függvény a "termek_id" paramétert veszi át, amely a törlendő termék azonosítója.

Először lekéri a "cuccok" elemet a helyi tárolóból, amely tartalmazza az összes terméket. Ezután egy AJAX hívást indít a "../api.php" URL-re POST módszerrel és JSON adattípussal. Az AJAX egy technológia, amely lehetővé teszi, hogy a weboldalak adatokat küldjenek és fogadjanak anélkül, hogy az oldal újratöltődne. Ez gyorsabb és kényelmesebb felhasználói élményt eredményez a weboldalak használatakor. Az adat paraméter tartalmazza a szükséges információkat a termék mennyiségének törléséhez, ideértve a „termek_id”-t és a törlendő termék darabszámát.

Ha a törlés sikeres, akkor a függvény eltávolítja a megfelelő terméksort a nézetből a "#sor_" + „result.lineid” azonosítójú elem eltávolításával. Ellenkező esetben frissíti a termék új mennyiségét a nézetben a ".darab" osztályú elem szövegének módosításával. Végül frissíti a "cuccok" elemet a helyi tárolóban az új termékek adataival.

2.2.7. API

Az API (Application Programming Interface) egy olyan programozási felület, amely lehetővé tesz két különböző alkalmazás közötti kommunikációt. Az API-n keresztül történő kommunikáció lehetővé teszi a rendszerek közötti adatátvitelt, így az egyik alkalmazás által gyűjtött adatok felhasználhatók lehetnek egy másik alkalmazásban is.

Az általunk fejlesztett API képes az adatok feldolgozására, valamint az adatbázisban történő tárolásra. Az API-n keresztül elvégezhetők az alábbi műveletek:

1) Termék darabszámának lekérdezése

```
function getCurrentProductNumber($order = array(), $dbconn) {
    $products = array();
    foreach($order as $orderItem) {
        $message = "";
        $sql = "
            SELECT darab
            FROM termek
            WHERE id = '{$orderItem["termek_id"]}'
        ";
        $result = mysqli_query($dbconn, $sql);
        $currentProductNumber = mysqli_fetch_array($result);
        if($currentProductNumber["darab"] < $orderItem["darab"]) {
            $message = "Nem áll rendelkezésre megfelelő számú termék";
            return array(
```

```

        "message" => $message,
        "products" => array(),
    ); }
    $products[$orderItem["termek_id"]] = $currentProductNumber["darab"];
}
$result = array(
    "message" => $message,
    "products" => $products,
);
return $result;}

```

A függvénynek két paramétere van. Az első paraméter egy tömb, amely a rendelt termékeket tartalmazza, valamint a darabszámukat. A második paraméter az adatbázis kapcsolatot jelenti, amely segítségével a függvény lekérdezi a termékek darabszámát. Ezután ellenőrzi, hogy minden termék rendelkezésre áll-e a megrendeléshez szükséges mennyiségben. Ha nem, akkor visszatér az üres tömbbel és egy üzenettel. Ha minden termék rendelkezésre áll, akkor a függvény összegyűjti a termékek darabszámát egy tömbben és visszaadja azt az összes termék darabszámával.

2) Termék darabszámának frissítése

```

function updateCurrentProductNumber($order = array(), $data = array(), $dbconn)
{
    foreach($order as $product) {
        $newProductNumber = $data["products"][$product["termek_id"]] -
        $product["darab"];

        $sql = "
            UPDATE termék
            SET darab = {$newProductNumber}
            WHERE id = '{$product["termek_id"]}'
        ";
        mysqli_query($dbconn, $sql);
    }
}

```

Itt az adatbázisban tárolt termékek mennyiségét frissítjük a rendelések feldolgozásakor. A függvény két tömb paramétert kap, az egyik a rendelés adatait, a másik pedig az összes termék és azok mennyiségét tartalmazza. Az adatbázis kapcsolatot is megkapja paraméterként. A függvény ellenőrzi az összes rendelt terméket, majd kivonja a rendelt darab értékét az adatbázisban található értékből. Az új mennyiséget az „UPDATE SQL” utasítással frissíti az adatbázisban. A függvény nem tér vissza semmilyen értékkel, csak végrehajtja a műveletet, amely biztosítja, hogy az adatbázisban tárolt termékek mennyisége mindig naprakész legyen a rendelések feldolgozásakor.

3) Megrendelés felvétele adatbázisba

```
function saveOrder($order = array(), $dbconn) {
    $string = bin2hex(opensslrandompseudobytes(10));
    foreach($order as $orderItem) {
        $sql = "
            INSERT INTO megrendeles
            (szemelyekid, rendeltDarab, termekid,rendelesazonosito,rendelesallapotid)
VALUES (
    '{$SESSION["id"]}','{$orderItem["darab"]}','{$orderItem["termekid"]}','{$string}','1'
    )
        ";
        mysqlquery($dbconn, $sql);
    }

    function updateProductAndSaveOrder($order = array(), $data = array(), $dbconn)
    {
        updateCurrentProductNumber($order, $data, $dbconn);
        saveOrder($order, $dbconn);
        $result = "Sikeres mentés";
        return $result;
    }
}
```

A „saveOrder” függvény egy megrendelést ment az adatbázisba, amely tartalmazza a termékek darabszámát és az azonosítóját. Az „updateProductAndSaveOrder” függvény először frissíti a termékek darabszámát a megrendelés alapján az „updateCurrentProductNumber()” függvénnyel, majd elmenti a megrendelést az adatbázisba a „saveOrder()” függvénnyel. A függvény visszatérési értéke egy "Sikeres mentés" üzenet.

4) Termék darabszámának csökkentése

```
if($_POST["c"] == "deleteProductAmount") {
    $message = "Sikeres mentés"
    $products = json_decode($_POST["product"], true);
    $productToUpdateIndex = array_search($_POST["termek_id"],
array_column($products, 'termek_id'))
    $deleteLine = false;
    $lineId = $products[$productToUpdateIndex]["id"];
    $newAmount = 0;
    if($products[$productToUpdateIndex]["darab"] > $_POST["darab_szam"]) {
        $newAmount = $products[$productToUpdateIndex]["darab"] -
$_POST["darab_szam"];
        $products[$productToUpdateIndex]["darab"] = $newAmount;
    } else {
        $deleteLine = true;
        unset($products[$productToUpdateIndex]);
    }
}
```

```

}
$result = array(
    "products" => $products,
    "deleteline" => $deletLine,
    "lineid" => $lineId,
    "newAmount" => $newAmount,
    "message" => $message,
);
echo json_encode($result);
}

```

Itt egy POST kérést kezelünk. Az adatokat JSON formátumban továbbítjuk a "product" kulcs alatt. A kód először dekódolja a JSON-t, majd megtalálja a termék indexét a "termek_id" alapján. A kód eltávolítja a termék mennyiségét a POST kérésben megadott mennyiséggel, majd ellenőrzi, hogy a termék mennyisége meghaladja-e a POST kérésben megadott mennyiséget. Ha igen, akkor a kód csökkenti a termék mennyiségét, ha nem, akkor a kód törli az egész sort. Az eredmény egy tömb, amely tartalmazza az új termék adatokat, a törölt sort, az új mennyiséget és egy üzenetet a felhasználónak. Végül a kód JSON formátumban elküldi az eredményt az eredeti kérésre válaszul.

5) Termék státuszának frissítése

```

if($_POST["c"] == "updateProductStatus") {

    $message = "Sikeres mentés";

    $sql = "
        UPDATE megrendeles SET rendeles_allapot_id = '{$_POST["status"]}'
        WHERE termek_id = '{$_POST["termek_id"]}' AND személyek_id =
        '{$_POST["szemely_id"]}'
    ";
    echo $sql;
    mysqli_query($dbconn, $sql);

    $result = array(
        "message" => $message,
    );
    echo json_encode($result);
}

```

Ez a kódrészlet frissíti a termék rendelési állapotát az adatbázisban a megadott termék- és személyazonosítók alapján. A kód egy POST kérést kap az „updateProductStatus” paraméterrel, majd frissíti a megrendeles táblában a „rendeles_allapot_id” oszlopot a status paraméter értékével. Az „update” lekérdezés a

„termek_id” és „szemelyek_id” paramétereket használja a táblában frissítendő sor azonosításához.

2.2.8. Kereső funkció

```
function search_item() {  
  let input = document.getElementById('searchbar').value  
  input = input.toLowerCase();  
  let x = document.getElementsByClassName('card');  
  for (i = 0; i < x.length; i++) {  
    if (!x[i].innerHTML.toLowerCase().includes(input)) {  
      x[i].style.display = "none";  
    }  
    else {  
      x[i].style.display = "flex"; }  
  }  
}
```

Ez egy JavaScript függvény, amely egy keresősáv segítségével keres elemeket, jelen esetben termékeket, a webshopunkban. A függvény a keresősáv beviteli mező értékét veszi át, majd kisbetűsre konvertálja a „toLowerCase()” metódus segítségével. Ezután az összes „card” osztálynevű elemet lekéri a „getElementsByClassName()” metódussal, és eltárolja őket az „x” változóban. A függvény végigmegy az „x” összes elemén egy „for loop” segítségével, hogy ellenőrizze, tartalmazza-e az elem „innerHTML”-je a keresési értéket. Ha az elem nem tartalmazza az értéket, akkor az elem „display” stílusa „none” lesz, hogy elrejtse azt. Ha tartalmazza, akkor az elem „display” stílusa „flex” lesz, hogy megjelenítse azt.

Összességében ez a függvény lehetővé teszi a felhasználó számára, hogy konkrét elemeket keressen a weboldalunkon, és csak azokat jelenítse meg, amelyek megfelelnek a keresési értékeknek. Az admin felületünkön is ugyanezt a kereső funkciót alkalmazzuk, viszont ott át kellett írunk a kódot, hiszen ott nem kártyák, hanem egy táblázat elemeit kellett hasonlóképpen szűrni.

2.2.9. Képek nagyítása

```
let modal = document.getElementById('modal');  
let images = document.getElementsByClassName('cikkkep');  
for (const image of images) {  
  image.addEventListener("dblclick", () => {  
    modal.children[0].src = image.src;  
    modal.classList.remove("hidden");  
  });  
}
```

Itt egy „modal” ablakot hozunk létre, amely akkor jelenik meg, amikor a felhasználó duplán kattint egy képre, és egy nagyobb verziót jelenít meg belőle. Erre azért

volt szükség, hiszen a webshopban a termék kártyáin kis hely volt a képeknek, viszont szerettük volna, ha a felhasználó jobban meg tudja vizsgálni a terméket.

Először azonosítjuk az „ID” alapján a „modal” elemet, majd az összes képet a „class” nevük („cikkép”) alapján. Ezután hozzáadtunk egy eseménykezelőt minden képhez, amely figyel a dupla kattintásra történő eseményeket. Amikor egy képre duplán kattintanak, a forrás URL-je frissíti a „modal” elem első gyerek elemének forrását, ami egy kép címke. A „modal” ablak eltávolítja a „hidden” osztályt az osztálylistájából, így megjelenik.

```
modal.children[1].addEventListener("click", () => {  
    modal.classList.add("hidden");});
```

Végül egy eseménykezelőt adtunk hozzá a „modal” elem második gyerek eleméhez, amely egy bezárás gomb. Amikor a bezárás gombra kattintanak, a „hidden” osztály visszakerül a „modal” osztálylistájához, így újra elrejtésre kerül.

2.2.10. Adminisztrációs lista

Itt a termékek listázását valósítottuk meg. Ezen a felületen tudja az adminisztrátor a termékeket felvinni, módosítani, illetve törölni.

```
session_start();  
if(!isset($_SESSION['belepett']))  
{  
    header("Location: ../belep.php");  
    exit();}
```

A kód először ellenőrzi, hogy a felhasználó be van-e jelentkezve, ha nem, akkor átirányítja a felhasználót a belépés oldalra. Az adminisztrációs oldalaknál kifejezetten fontos a lapvédelem, hiszen visszafordíthatatlan károkat okozhatnak az illetéktelen felhasználók.

```
require("../kapcsolat/kapcs.php");  
$sql = "SELECT * from termek  
INNER JOIN alkategoriak  
ON alkategoriak.alkategoria_id = termek.alkategoria_id  
INNER JOIN kategoriak  
ON alkategoriak.kategoria_id = kategoriak.kategoria_id";
```

Ezután kapcsolatot hoz létre az adatbázissal, majd lekérdezi a termékek adatait a "termek", "alkategoriak" és "kategoriak" táblákból.

```
$eredmeny = mysqli_query($dbconn, $sql);  
$kimenet = "<table><thead>
```

```

<tr>
<th>Fotó:</th>
<th>Vonalkód</th>
<th>Név</th>
<th>Kategória</th>
<th>Alkategória</th>
<th>Felvitel dátuma</th>
<th>Darabszám</th>
<th>Ára</th>
<th>Műveletek:</th>
</tr>";
$skimenet .= "</thead><tbody class='tabla'>";
while($sor = mysqli_fetch_assoc($eredmeny))
{
$skimenet .= "
<tr>
<td class='kep'><img src='../keps/{ $sor['foto']}' alt='{ $sor['foto']}'
style='width: 100%;'></td>
<td class='vonalkod'>{ $sor['vonalkod']}</td>
<td class='nev'>{ $sor['nev']}</td>
<td class='kategoria_nev'>{ $sor['kategoria_nev']}</td>
<td class='alkategoria_nev'>{ $sor['alkategoria_nev']}</td>
<td class='felvdatum'>{ $sor['felv_datum']}</td>
<td class='darab'>{ $sor['darab']}</td>
<td class='ar'>{ $sor['ar']}</td>
<td class='padd'><a href='torles.php?id={ $sor['id']}'>Törlés</a> | <a
href='modos.php?id={ $sor['id']}'>Módosítás</a></td>
</tr> ";
}
$skimenet .= "</tbody></table>";

```

Az eredményeket egy HTML táblázat formájában jeleníti meg, amely tartalmazza a termékek fotóját, vonalkódját, nevét, kategóriáját, alkategóriáját, felvitel dátumát, darabszámát és árát. Az utolsó oszlopban találhatók a törlési és módosítási gombok.

2.2.11. Termékek felvitele

Ezen az oldalon az adminisztrátornak lehetősége van új termékeket felvinni.

```

session_start();
if (!isset($_SESSION['belepett'])) {
    header("Location: ../belep.php");
    exit();
}

```

Ez a rész ellenőrzi, hogy az adminisztrátor belépett-e a munkamenetbe, ha nem, akkor a felhasználó átirányítódik a belépés oldalra. Ez biztosítja, hogy csak bejelentkezett felhasználók férjenek hozzá az oldalhoz és tölthessenek fel új termékeket.

```

if (isset($_POST['ok'])) {
    $vonalkod = strip_tags(trim($_POST['vonalkod']));
}

```

```

$nev = strip_tags(trim($_POST['nev']));
$ar = strip_tags(trim($_POST['ar']));
$darab = strip_tags(trim($_POST['darab']));
$felvdatum = strip_tags(trim($_POST['felvdatum']));
$alkategoria_id = strip_tags(trim($_POST['alkategoria_id']));
$leiras = strip_tags(trim($_POST['leiras']));
$mime = array("image/gif", "image/png", "image/jpg", "image/jpeg");

```

Itt olvassuk be az adatokat az űrlapból, majd megtisztítjuk őket. Az „SQL injection” támadások elleni védelem érdekében a script előkészített utasításokat használ a felhasználói bemenet előzetes megtisztítására, mielőtt azokat az adatbázisba illesztené. Ez segít megakadályozni a kártékony kódok végrehajtását.

```

if (empty($vonalkod) ){
    $hibak[] = "Adjon meg vonalkódot!";
}
if (empty($nev) ){
    $hibak[] = "Adjon meg nevet!";
}
if (empty($alkategoria_id) ){
    $hibak[] = "Adjon meg kategóriát!";
}
if (empty($felvdatum) ){
    $hibak[] = "Adjon meg dátumot!";
}
if (empty($ar) ){
    $hibak[] = "Adjon meg árat!";
}
if (empty($alkategoria_id) ){
    $hibak[] = "Adjon meg darabszámot!";
}
if (empty($leiras) ){
    $hibak[] = "Adjon meg leírást!";
}
if ($_FILES['foto']['error'] == 0 && $_FILES['foto']['size'] > 6000000) {
    $hibak[] = "Nagy a kép formátuma";
}
if ($_FILES['foto']['error'] == 0 && !in_array($_FILES['foto']['type'], $mime))
{
    $hibak[] = "Rossz a kép fájl kiterjesztése";
}
switch ($_FILES['foto']['type']) {
    case "image/png":
        $kit = ".png";
        break;
    case "image/gif":
        $kit = ".gif";
        break;
    case "image/jpg":
        $kit = ".jpg";

```

```

        break;
    default:
        $kit = ".jpeg";
    }
    $foto = date("U") . $kit;

```

A kód tartalmazza a fájl feltöltésekre vonatkozó hibakezelést is, biztosítva, hogy az összes hiba észlelésre kerüljön és megjelenjen az adminisztrátornak.

```

if (isset($hibak)) {
    $kimenet = "<ul class=\"error-msg\">\n";
    foreach ($hibak as $i) {
        $kimenet .= "<li>$i</li>";
    }
    $kimenet .= "</ul>";
}

```

A script ellenőrzi az űrlapadatokat hibákra. Ha bármilyen hiba van, akkor megjeleníti azokat az adminisztrátornak, hogy javítsa ki azokat, mielőtt újra véglegesítene az űrlapot. Ez segít biztosítani, hogy az űrlapba bevitt adatok pontosak és teljesek legyenek.

```

    } else {
        require("../kapcsolat/kapcs.php");
        echo "$alkategoria_id";
        $sql = "INSERT INTO termek
            (vonalkod,nev,felvdatum,ar,darab,foto,alkategoria_id,leiras)
        VALUE('{$vonalkod}','{$nev}','{$felvdatum}','{$ar}','{$darab}','{$foto}','{$alka
        tegoria_id}','{$leiras}')
        ";
        mysqli_query($dbconn, $sql);
        move_uploaded_file($_FILES['foto']['tmp_name'], "../keps/{$foto}");
        header("Location: adminlist.php");
    }
}

```

Feltételezve, hogy nincsenek hibák, a script beilleszti az űrlapadatokat az adatbázisba. A script kezeli a fájl feltöltéseket is, áthelyezve a feltöltött képet az adott mappába a szerveren. Ez biztosítja, hogy a feltöltött fájlok biztonságosan tárolódnak, és később könnyen hozzáférhetők, ha szükséges.

2.2.12. Termékek módosítása

Ez a kód szinte megegyezik a „felvitel.php” oldallal, azzal a különbséggel, hogy itt az adminisztrátor a termékek adatait tudja szerkeszteni. A kód elkezd a feldolgozást, ha a felhasználó megnyomja az "ok" gombot. Ettől kezdve ellenőrzi, hogy minden szükséges mező kitöltésre került-e, és ha nem, akkor hibaüzenetet jelenít meg a felhasználónak. Ha minden mező kitöltésre került, a kód továbblép a fájl feltöltésének ellenőrzésére. Az ellenőrzés során azt ellenőrzi, hogy a feltöltött fájl megfelelő formátumú-e, és ha nem, akkor ismét hibaüzenetet jelenít meg a felhasználónak. Ha a feltételeknek megfelelően

minden mező kitöltésre került és a feltöltött fájl megfelelő formátumú, a kód frissíti az adatokat az adatbázisban. Ezután a kód a képet a végleges helyére helyezi, és megjeleníti a felhasználónak a frissítés sikerességéről szóló üzenetet.

A kód tartalmaz hibaüzeneteket, ha valamelyik mezőt nem, vagy hibásan töltötte ki a felhasználó. Ez biztosítja, hogy a felhasználók könnyen érthető hibaüzeneteket kapjanak, ha valami nem megfelelő az űrlap kitöltésében. A kód hatékonyan és biztonságosan dolgozza fel az űrlapadatokat, és lehetővé teszi az adminisztrátor számára a termékek adatainak egyszerű és gyors szerkesztését.

2.2.13 Termék törlése

```
if(isset($_GET['id']))
{
    require("../kapcsolat/kapcs.php");

    $id= (int)$_GET['id'];
    $sql = "DELETE FROM termek
        WHERE id = {$id}";
    mysqli_query($dbconn, $sql);
}
header("Location: adminlist.php");
```

A törlés gombra kattintva kapcsolódunk az adatbázishoz, majd töröljük azokat az adatokat a "termek" táblából, amelyeknek az „id” mezője megegyezik a „id” változó értékével. Ezután átirányítja az oldalt az „adminlist.php” oldalra. Ha az „id” változó nem létezik, akkor nem történik semmi.

2.2.14. Tesztelés

A weboldalaink HTML kódrészleteit a <https://validator.w3.org/> oldalon ellenőriztük. Találtunk néhány hibát, amelyet nem vettünk észre, mivel az oldal látszólag megfelelően működött. Például voltak olyan „div” konténerek, amelyek nem kerültek lezárásra. Így ezeket a hibákat kijavítottuk.

A CSS-t a <https://jigsaw.w3.org/css-validator/> weboldal segítségével ellenőriztük. A CSS-ben nem találtunk hibát, melyben nagy segítségünkre volt a Visual Studio Code fejlesztői környezet is, amely azonnal jelzi a hibát. A megfelelő formázást a böngésző fejlesztői eszközében is ellenőriztük.

A Javascript kódjainkat a böngészőben a konzolon ellenőriztük. A kód írása során a „console.log()” segítségével ellenőriztük folyamatosan, hogy megkapjuk-e azt az elemet, amelyet éppen aktuálisan el akartunk tárolni pl. egy változóba.

A PHP kód ellenőrzését a <https://www.phptools.online/php-checker> oldalon is ellenőriztük, itt már hibát nem találtunk, mivel a programozás során is folyamatosan ellenőriztük, „print_r”-el, vagy „var_dump()”-al kiírva folyamatosan a kért és kapott adatokat.

3. Felhasznált technológiák

Projekt munkánkhoz igyekeztünk minden olyan technológiát felhasználni, amelyet a tanév során tanultunk. Ezeket a következő pontokban mutatjuk be.

3.1. Fejlesztői környezet

3.1.1. Visual Studio Code

A Visual Studio Code egy ingyenes és nyílt forráskódú kódszerkesztő, amit a Microsoft fejlesztett. Windows, macOS és Linux operációs rendszereken is elérhető. Olyan funkciókat nyújt, mint a szintaxis kiemelés, kódbevitel, hibakeresés és Git integráció, ezek a funkciók a munkánkat nagyban segítették. Emellett bővítményeket is támogat, amelyek letölthetők a Visual Studio Code-ból. A bővítményeket előszeretettel alkalmaztuk munkánk során, hiszen olyan plusz funkciókkal látja a felhasználót, ami hozzájárul a kód átláthatóságához, mint például a zárójelek kiemelése. A Visual Studio Code a fejlesztők körében népszerű választás lett egyszerűsége, könnyű használhatósága és testreszabhatósága miatt. Emellett a Visual Studio Code számos programozási nyelvhez használható, ami sokoldalú választássá teszi a fejlesztők számára.

3.1.2. GitHub

GitHub egy webes, illetve asztali platform, amely lehetővé teszi a szoftverfejlesztők számára, hogy közösen dolgozzanak együtt a projektjeiken. A GitHub lehetővé teszi a fejlesztők számára, hogy tárolják a kódot, nyomon kövessék a változtatásokat, ellenőrizzék a verziókat és együttműködjenek más fejlesztőkkel. A platform ingyenes és nyílt forráskódú projektekhez is használható. A GitHub nagyon népszerű a fejlesztők körében, hiszen a használata egyszerű és könnyen tanulható és számos integrációval rendelkezik más fejlesztői eszközökkel, mint például az IDE-k. A GitHub-ot végig alkalmaztuk projekt munkánk során, kifejezetten a verziókövetés funkciót, hiszen ketten dolgoztunk a projekten, és így könnyedén tudtunk akár egy időben is dolgozni rajta. A GitHub-on a projekt munkánk teljes egészében megtalálható.

3.1.3. Photopea

Photopea egy felhőalapú képszerkesztő szoftver, amely bármely webböngészőn keresztül elérhető. A Photopea segítségével úgy szerkeszthettük és javíthattuk fényképeinket anélkül, hogy letöltenék a szoftvert. Azért ezt a szoftvert választottuk, mert bárholnan elérhető és ingyenesen használható. Bár kevesebb funkcióval rendelkezik, mint az Adobe Photoshop, széles körű eszközök és funkciók állnak rendelkezésre,

beleértve az egyéni felületet, a rétegszerkesztést és a különböző fájlformátumokkal való munkavégzés képességét.

3.2. Frontend eszközök

A frontend a programoknak a weboldalaknak azt a részét tekintjük, amely közvetlenül kapcsolatban áll a felhasználóval. Feladata az adatok megjelenítése, befogadása a felhasználó, vagy más esetekben egy másik rendszer felől. A weboldalak fő célja, hogy a felhasználók számára jelentős információt prezentáljon. A legismertebb nyelvek, amelyeket erre a célra használunk a HTML5, a CSS és a JavaScript. A HTML elsősorban az információtartalomért felelős, amíg a JavaScript és a CSS a megjelenítést biztosítják.

3.2.1. HTML

Az HTML (HyperText Markup Language) egy olyan szabványosított kódolási nyelv, amelyet weboldalak létrehozására használnak. Az HTML segítségével lehet strukturálni egy weboldalt, vagyis meghatározni az oldal elemeit, mint például a címsorokat, szövegeket, képeket, linkeket, űrlapokat és táblázatokat. Az HTML szabványosított nyelv, amely minden modern böngészőben támogatott. Az HTML-t könnyű megtanulni, és az egyik alapvető nyelv, amelyet minden webfejlesztőnek ismernie kell. Az HTML és a CSS (Cascading Style Sheets) együtt alkotják az alapját minden weboldalnak, és az egyszerű oldalaktól kezdve a bonyolult alkalmazásokig minden feladatra használhatóak.

3.2.2. CSS

Cascading Style Sheet (CSS) kódokat a HTML elemek formázására használjuk. A CSS kódot egy külön fájlban tároljuk, ezt külső stíluslapnak nevezzük. Összetett weboldalak esetén, mint a mi projektünkben szereplő weboldalak, célszerű a külső stíluslapok használata, mivel ezzel a módszerrel csak egyszer kell beállítanunk az oldalakon található HTML elemek stílusjellemzőit. A CSS stíluslapokkal könnyedén biztosítható az egységes megjelenés, hiszen az elkészült fájl több oldalon is felhasználható. Fontos tényező még, hogy az oldal karbantartása és továbbfejlesztése során jelentősen lerövidíti a vele járó munkát, hiszen ha egy stíluselemet kívánunk később módosítani, nem kell megváltoztatni a weboldal összes oldalán (manapság ez lehet több száz), és a belső stíluslapban egyesével átírni a kódot, elegendő egy helyen elvégezni a változtatást, a CSS fájlban.

3.3. Backend eszközök

A backend technológiák olyan eszközök, amelyek lehetővé teszik az alkalmazások működését a szerver oldalán. A hátoldali fejlesztés során a fejlesztők olyan eszközöket használnak, amelyekkel a webalkalmazások adatbázisokkal kommunikálhatnak, adatokat tárolhatnak, feldolgozhatnak és szolgáltathatnak. Mi elsősorban a JavaScript és a PHP nyelvet használtuk a backend fejlesztés során.

3.3.1. JavaScript

A JavaScript az egyik legfontosabb eszköz a webfejlesztők számára a sokoldalúsága és könnyű alkalmazhatósága miatt, a nyelvet interaktív weboldalak és webalkalmazások létrehozásához használjuk. Az egyik legnépszerűbb programozási nyelv a világon, és az összes nagyobb böngészőt támogatja, mint például a Chrome, a Firefox és a Safari, így külön telepítésre nincs szükség. A JavaScript lehetővé teszi a dinamikus tartalmat a weboldalakon, például a legördülő menüket, a felugró ablakokat és az űrlapok ellenőrzését. Emellett használják animációk, játékok és más interaktív funkciók létrehozásához. A JavaScript az évek során sokat fejlődött, és most már számos keretrendszer és könyvtár elérhető a fejlesztés hatékonyságának növelése érdekében. A legnépszerűbb keretrendszerek közé tartozik az AngularJS, a React és a Vue.js. A munkánk során mi is számos helyen alkalmaztuk ezt a nyelvet, mint például a webshop kosarának működéséhez, vagy a kereső funkciókhoz.

3.3.2. PHP

PHP egy szerveroldali, nyílt forráskódú programozási nyelv, amelyet kifejezetten webes fejlesztésre terveztek. A PHP kódot általában a szerver feldolgozza, és a kimenetet HTML-ként jeleníti meg a böngészőben. A PHP lehetővé teszi a weboldalak dinamikus felépítését, interaktív funkciók hozzáadását, és adatbázisokkal történő kommunikációt. A PHP-t gyakran használják olyan webes alkalmazások fejlesztéséhez, mint például a tartalomkezelő rendszerek, az internetes boltok és az online szolgáltatások. A projektünkben számos helyen alkalmaztuk, mint például az elemek beolvasása az adatbázisból, majd kiírása a webshopba. A PHP nyelv könnyen tanulható, és rengeteg kész funkcióval rendelkezik, amely megkönnyítette a fejlesztési munkánkat.

3.3.3. MySQL és XAMPP

A XAMPP egy ingyenes és nyílt forráskódú szoftvercsomag, amely képes telepíteni és konfigurálni egy teljes webes szerverkörnyezetet a saját számítógépünkön. Ez a

szoftvercsomag tartalmazza az Apache web szerver, a MySQL adatbázis-kezelő rendszert, a PHP programozási nyelvet, valamint az OpenSSL és az phpMyAdmin eszközöket.

A MySQL az egyik legnépszerűbb nyílt forráskódú relációs adatbázis-kezelő rendszer a világon. Ez a rendszer hatékonyan kezeli az adatokat, és számos biztonsági funkcióval rendelkezik, amelyek megvédik az adatokat a jogosulatlan hozzáféréstől. Az MySQL könnyen integrálható más programozási nyelvekkel, mint például a PHP, Python, Ruby, Java és mások, így rugalmas lehetőséget kínál a fejlesztők számára.

A XAMPP és az MySQL kombinációja egy nagyon hatékony környezetet teremtett számunkra, amely lehetővé tette, hogy helyben dolgozzunk a saját számítógépjeinken anélkül, hogy szükségünk lett volna egy valódi szerverre.

Összefoglalás

Vizsgaprojektünk során szinte az összes tanult technikát alkalmaztuk, de ahogy készült a weboldal, egyre inkább születtek az újabbnál újabb ötletek, amelyek megvalósításához már az interneten igyekeztünk különböző megoldásokat találni. Ezeket a megoldásokat, kódokat, kódrészleteket értelmeztük és szükség esetén átszabtuk a saját alkalmazásunkhoz paraméterezve. Ennél fogva rengeteg új ismeretet szereztünk, nemcsak a programozásban, hanem a keresés, és kód-újrahasznosítás területén is.

A vizsgamunkánk elkészülte után, év végén foglalkoztunk keretrendszerekkel, mint CakePhp és a Laravel. A tanév rövidsége miatt már ezeket a technológiákat nem tudtuk olyan mélységig megismerni, hogy az ötleteinket meg tudjuk valósítani benne. Viszont nagyon megtetszett a könnyű kezelése és karbantarthatósága, így majd a jövőben tervezzük, hogy az oldalt Laravel 10 keretrendszerbe ültetjük át. További ötletünk még, hogy szeretnék online fizetési lehetőséget is biztosítani a vásárlóknak, valamint a szállítási mód kiválasztását. Ehhez természetesen szerződnünk kell majd különböző szolgáltató cégekkel, akik a szállítást végzik, illetve amin keresztül a fizetés történhet. Számla generátort egyelőre azért nem terveztünk a projektbe, mert már a bolt egy meglévő és működő számlázási rendszert használ, viszont gondolkodtunk, hogy a jövőben elkészítenénk egy saját számlázó modult a webáruházhoz, ami saját igényünk szerint működne.

Az adatbázis is ki kell bővítenünk a jövőben a további terveinkhez. Több fejlesztést is szeretnénk majd végezni. Az egyik ilyen például, hogy jelenleg az összes regisztrált felhasználót egy táblában tároljuk, legyen az admin vagy vásárló, ezt két külön táblában szeretnénk majd tárolni. A másik, hogy a rendelések alkalmával, a felhasználóhoz társított kiszállítási címre van lehetőségük csak rendelni a vásárlóknak, viszont szeretnénk bővíteni egy olyan funkcióval, amely elősegíti, hogy a felhasználó alapértelmezett címétől, eltérő címre is lehessen rendelést leadni.

A település táblát ki szeretnénk majd egészíteni, olyan módon, hogy a külföldi rendeléseket is el tudjuk tárolni. Erre legfőképpen azért van szükség, mert üzletünk Balassagyarmaton, szlovák határ mellett helyezkedik el, és jelentős tömegű vásárló érkezik a határon túlról is. Ebből az okból kifolyólag lehetőséget szeretnénk adni, hogy webshopunkat ők is használhassák.

Fontos fejlesztés lenne a jövőben, hogy a felhasználók regisztrálása e-mail-ben küldött megerősítőhöz legyen kötve, ezzel kiszűrnénk a nem valós regisztrálásokat. Úgy tervezzük, hogy a vásárlás megerősítése után is kapjon majd egy e-mail-t tőlünk a felhasználó, de ezt még most nem tartottuk fontosnak, hiszen az oldalunkon nyomon tudja követni rendeléseit a felhasználó. A kézbesítés folyamatáról a futárszolgálat értesíti a felhasználót.

A díszállatokról szóló oldalunkat úgy szeretnénk volna megvalósítani, hogy egy cikkszerkesztő felületet készítsünk hozzá, és az adminisztrátornak lehetősége nyíljon új cikkeket, leírásokat létrehozni. Ennek a fejlesztését elkezdtük, de beláttuk, hogy ez lényegesen nagyobb munkával jár, ezért ez az ötlet felfüggesztésre került, de a jövőben mindenképpen szeretnénk ezt megvalósítani.

Összességében nagyon jól tudtunk együtt dolgozni a projekten, hiszen számunkra kedvelt és ismert témát dolgoztunk fel.

Volt néhány olyan ötletünk, amelyekhez szükségünk volt oktatóink segítségére is, mivel hasonló feladatmegoldással még nem találkoztunk. Ilyen volt például, a localStorage-ből való adatkiolvasás és adatbázisba beírás, amelynek megoldásával nagyon sok új technikát tanultunk.

Projektünk elérhetősége a Github csatornánkon:

https://github.com/Starkman56/2023_zaro_dolgozat

A nethely ingyenes tárhelyén:

http://zarovizsgaszak.nhely.hu/20230112/2023_zaro_dolgozat/Projektmunka/user/main.php

Források

- 1) A Complete Guide to CSS Grid:
<https://csstricks.com/snippets/css/complete-guide-grid/>
- 2) A Complete Guide to Flexbox: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- 3) Responsive, customizable, accessible (wai-aria) replacement for javascript's popup boxe: <https://sweetalert2.github.io/>
- 4) Select2 gives you a customizable select box with support for searching, tagging, remote data sets, infinite scrolling, and many other highly used options.: <https://select2.org/>
- 5) Create the perfect palette or get inspired by thousands of beautiful color schemes: <https://coolers.co/palettes/trending>
- 6) HTML tables: https://www.w3schools.com/html/html_tables.asp
- 7) HTML Input Types:
https://www.w3schools.com/html/html_form_input_types.asp
- 8) JavaScript Arrow: Function:
https://www.w3schools.com/js/js_arrow_function.asp
- 9) CSS Layout Overflow: https://www.w3schools.com/css/css_overflow.asp
- 10) CSS Grid Container:
https://www.w3schools.com/css/css_grid_container.asp
- 11) Ajax function: <https://api.jquery.com/jquery.post/>
- 12) Just-add-water CSS animations: <https://animate.style/>
- 13) Find child element of parent:
<https://stackoverflow.com/questions/16302045/finding-child-element-of-parent-with-javascript>
- 14) Add class to elements: <https://stackoverflow.com/questions/507138/how-to-add-a-class-to-a-given-element>
- 15) Double-click on objects:
<https://stackoverflow.com/questions/23926921/how-do-i-double-click-on-objects-using-javascript-do-i-have-to-click-twice>
- 16) LocalStorage and JSON.stringify JSON.parse:
<https://stackoverflow.com/questions/23728626/localstorage-and-json-stringify-json-parse>

17) SQL INSERT INTO SELECT Statement:

https://www.w3schools.com/sql/sql_insert_into_select.asp

18) Send JSON data via POST (ajax) and receive json response from

Controller (MVC): <https://stackoverflow.com/questions/8517071/send-json-data-via-post-ajax-and-receive-json-response-from-controller-mvc>

19) Országlista, országok, megyék és városok listája SQL-ben, TXT-ben

(CSV/TSV), Excelben http://webdraft.eu/orszagok_varosok/

Ábrajegyzék

1)	ábra Kezdőoldal.....	4
2)	ábra Galéria	5
3)	ábra Díszállatok	6
4)	ábra Halak.....	7
5)	ábra Keresőmező működése	8
6)	ábra Termék kártyák megjelenése	9
7)	ábra Kosár tartalma.....	9
8)	ábra Megrendelés megerősítése.....	10
9)	ábra Bejelentkezés	10
10)	ábra Regisztrációs űrlap.....	11
11)	ábra Termékek adminisztrációs oldala	11
12)	ábra Megrendelések nyomon követése	12
13)	ábra Új termék felvitele űrlap	13
14)	ábra Módosítás űrlapja.....	13
15)	ábra Termék tábla	17
16)	ábra Kategória tábla	17
17)	ábra Alkategóriák tábla.....	18
18)	ábra Megrendeles tábla I. része.....	19
19)	ábra Megrendeles tábla II. része	19
20)	ábra Szemelyek tábla I. része.....	20
21)	ábra Szemelyek tábla II. része	20
22)	ábra Rendeles_allapot tábla	21
23)	ábra Telepulesek tábla	21
24)	ábra Adatbázis diagram	22