Szabó Máté – Kordics Balázs

A weboldal egy meglévő offline bolt új üzleti lehetőségeit mutatja be, online felületen.

Webshop

Halas üzlet az internetre költözik

Tartalom

[1. Felhasználói dokumentáció 3](#_Toc129857711)

[1.1. Főoldal 3](#_Toc129857712)

[1.2. Webshop oldal 3](#_Toc129857713)

[1.3. Megrendelés megerősítése oldal 4](#_Toc129857714)

[1.4. Adminisztrációs oldalak 4](#_Toc129857715)

[1.4.1. Belépés 5](#_Toc129857716)

[1.4.2. Lista nézet 5](#_Toc129857717)

[1.4.3. Felvitel 5](#_Toc129857718)

[2. Fejlesztői dokumentáció 6](#_Toc129857719)

[2.1. Adatbázis tervezés 6](#_Toc129857720)

[2.1.1. Adatbázis célja 6](#_Toc129857721)

[2.1.2. Táblák kapcsolata 6](#_Toc129857722)

[2.1.3. Egyedek meghatározása 6](#_Toc129857723)

[2.1.4. Termék tábla 6](#_Toc129857724)

[2.1.5. Alkategóriák tábla 7](#_Toc129857725)

[2.1.6. Kategória tábla 7](#_Toc129857726)

[2.1.7. Megrendelés tábla 7](#_Toc129857727)

[2.2. Weboldal 8](#_Toc129857728)

[2.2.1. Kezdő oldal 8](#_Toc129857729)

[2.2.2. Beléptető és Regisztrációs oldal 8](#_Toc129857730)

[2.2.3. Webshop működése 8](#_Toc129857731)

[2.2.4. Beléptetés (titkosítás, sha256) 9](#_Toc129857732)

[2.2.5. Listázás 9](#_Toc129857733)

[2.2.6. Felvitel 9](#_Toc129857734)

[2.2.7. Módosítás 9](#_Toc129857735)

[2.2.8. Törlés 9](#_Toc129857736)

[3. Felhasznált technológiák 9](#_Toc129857737)

[4.1 Frontend eszközök 9](#_Toc129857738)

[4.1 HTML5 9](#_Toc129857739)

[4.2 CSS 9](#_Toc129857740)

[4.3 JavaScript 10](#_Toc129857741)

[4. Összegzés 11](#_Toc129857742)

[5. Források: 12](#_Toc129857743)

**Bevezetés**

Záró project munkánk 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 forgalom növekedést szeretnének elérni és ismerté válni szélesebb földrajzi környezetben is.

A kis vállalkozások 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 érkezhetnek majd a vásárlók, hanem akár az ország egész területéről rendelhetnek majd a leendő ügyfelek, még akár a határon túlról is. Az elmúlt évek eseményei is, a pandémia, a bezárás időszaka is megmutatta, hogy azok az üzletek tudták könnyedén átvészelni ezt az időszakot, akik lehetőséget nyújtottak ügyfeleiknek az online vásárlásra.

Ezen az ötleten tovább indulva , nem csak egy közönséges weboldalt terveztünk létrehozni, ahol bemutatjuk a termék választékot, 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, amellyel 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.

# Felhasználói dokumentáció

Az általunk elkészített weblap 3 oldalból áll, egy az üzletet bemutató lapból, külön oldalra került a webshop, és a megrendelés.

## Főoldal

A látogató az index nevű fő oldalunkkal találkozik először. Éppen ezért itt nagy gondot fordítottunk a dizájnra, de ugyanakkor figyelembe vettük azt is, hogy a legfontosabb információkat is tartalmazza. Mint magának a boltnak a bemutatása, mi minden kapható, néhány cikk az új termékekről, és hogyan juthat el hozzánk a vásárló, ide egy Google térképet is beágyaztunk.

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. Alatta a főbb termékcsoportok bemutatása képekkel és rövid leírással, majd végül az üzlet elérhetőségei és a térkép látható.

Először azt terveztük a termékek bemutatása alá kerül a webshop, de utána olvastunk a weboldalak ún. hőtérkép vizsgálatának, és annak 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.

+ képernyő kép

## Webshop oldal

A webshop odalán is elhelyeztünk egy navigációt, hogy a látogató vissza tudjon térni a fő oldalra. A menü alatt találhatóak a rendelhető termék csoportok kártyákon, amelyekre kattintva a webshopra navigálja a látogatót, ahol azok a termékek lesznek csak láthatóak, amelyek abban a termék csoportban 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 horgácikkeken belül csak a horgászbotokra és így tovább.

Szerettünk volna megvalósítani egy olyan kereső funkciót is, amivel a termék egy olyan adatára kereshet a látogató , amit épp tud a termékről, mint például a neve, cikkszáma ,gyártója stb. . Ez a keresőmező sokkal gyorsabb keresést eredményez , ami a látogató kedvére tehet.

+kép a keresésről

A termékek kártyákon kerülnek megjelenítésre, felül a termék fotója, alatta a neve, a raktáron lévő mennyiség, a termék ára, egy rövid leírás és egy input mező, ahol beállítható, hány darabot szeretne rendelni az adott termékből a látogató és egy küldés gomb „kosárba” felirattal. Ha a felhasználó egy terméket helyez a kosárba, akkor a jobb felső sarokban megjelenik a bevásárló kocsi ikonja mellett a rendelt termékek darabszáma. Ha a bevásárlókocsira kattint a látogató, akkor felugró ablakban megjelenik egy táblázat, amely a rendelése adatait tartalmazza.

+kép a bevásárló kocsiról

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

## 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.

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

## Adminisztrációs oldalak

A bolt tulajdonosának készítettünk egy olyan adminisztrációs felületet, ahol belépve egyszerű tartalomkezelő felületekkel tud felvinni új termékeket, tud árat módosítani, illetve tud termékeket törölni. Ahhoz, hogy az adminisztráció felületre be tudjon lépni, úgy döntöttünk biztonsági okokból nem helyezünk el figyelemfelkeltő linket, vagy gombot, hogy elkerüljük az illetéktelen felhasználók próbálkozásait. Így akkor tud belépni a tulajdonos, hogyha a domain név után beírja /admin, illetve admin joga van. Az adminisztrációs oldalon van lehetőség új adminokat létrehozni, ha az idő folyamán több adminisztrátorra lenne szükség.

### Belépés

Itt egy belépési felületre jut, ahol meg kell adnia a felhasználónevét és jelszavát. E nélkül nem tud a további oldalakra lépni, még akkor sem ha tudja a megfelelő oda vezető linket, mivel lapvédelmet adtunk az adminisztrációs oldalakhoz.

### Lista nézet

Ha az adatokat megfelelően adta meg, akkor jut a termékek listájának oldalára, ahol a termékek táblázatosan felsorolásra kerültek. A táblázat tetejére került egy link, amely az új termék felvitele oldalra vezeti.

### Felvitel

Van lehetőség a lista oldalról új terméket felvinni az adatbázisba, és a webshopba egyaránt. Az” új termék felvitele” gomb egy másik oldalra navigál , ahol a termék adatainak megadása történik. Miután minden szükséges adat megadásra került, megerősítés után a termék felkerül a webshopba , illetve az adatbázisba beszúrásra kerül.

* + 1. **Módosítás**

A módosítás gombra kattintva, a már meglévő termékek módosítására van lehetőség. Ez a funkció azért nagyon fontos, hiszen ha valamit elgépelünk felvitel közben ,nem kell a termék összes adatát törölni, elég a hibás adatot módosítani.

# Fejlesztői dokumentáció

## Adatbázis tervezés

A megfelelő tervezés létfontosságú feladat az adatbázissal végzett munka céljának elérésében. A gondos megtervezés 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.(kieg: lekérdezések korrekt eredmenyt adjanak)

Gondosan figyeltünk arra, hogy redundáns adat ne szerepeljen az adatbázisunkban, hiszen helyet pazarol és növeli a hibák számát és az ellentmondások előfordulásának esélyét.(!kieg: konzisztens legyen az adatbázis, hivatkozási intergritás szabályainak megfeleljen) Az adatbázis megtervezése közben figyeltünk arra is, hogy lehetőleg konzisztens legyen.

### Adatbázis célja

Az adatbázis célja egy adatbázis, egy olyan webáruházhoz, ami képes olyan adatok tárolására, mint például regisztrált felhasználók adatai, felvitt termékek adatai, illetve a megrendelésekhez fűződő egyéb adatok eltárolása

## 2.2Egyedek meghatározása

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

### 2.2.1 Termék

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

### 2.2.2 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 adminiok 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, hiszen ez tovább bonyolítaná az adatbázist, 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.

Egy másik akadályba is ütköztünk tervezés során, mégpedig a személyek rendelési címének tárolása. A rendelési címet ajánlott szétválasztani több táblára…

### 2.2.3 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.

### 2.2.4 Alkategória

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

## 2.3 Kapcsolatok

Az adatbázisunk 2 féle kapcsolatot kezel, 1:N illetve 1:M. 1:N kapcsolat esetén az egyik egyedhez több másik egyedet tudunk társítani, de a másik egyed 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 .

### 2.3.1 Alkategóriák - Termék

A két egyed között **1:N** kapcsolat áll fenn, hiszen egy termékhez csak egyetlen alkategória társul és egy alkategóriába több termék is megtalálható

### 2.3.2 Kategóriák - Alkategóriák

Itt is szintén **1:N** kapcsolat figyelhető meg. Egy kategóriába több alkategória tartozik továbbá egy alkategória csak egy kategóriába sorolható.

### 2.3.3 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 relációban , hiszen egy termékből több személy is vásárolhat. A termék és a személyek N:M-es kapcsolatát kettő 1:N kapcsolatra kellet szétbontanunk, ezt a **megrendelés** kapcsolótáblával tettük lehetővé.

### 2.3.4.1 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.

### 2.3.4.2 Személyek - Megrendelés

A személyek és a megrendelés között 1:N kapcsolat található. Egy személyhez tartozhat több megrendelés, viszont egy adott megrendelés csak egy személyhez tartozik.

### 2.3.5 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

## 2.4 Táblák meghatározása

Ezen a ponton már ismerjük az egyedeket, illetve a kapcsolatok egymás között. Így már meg tudjuk határozni ténylegesen a táblákat.

### 2.5.1 Termek 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 a 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 alkategoriak elsődleges kulcsára hivatkozik | int (11) |

****

### 2.4.2Kategoria tábla

Leírás: A webáruház termékei 4 fő kategóriára sorolható. 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) |

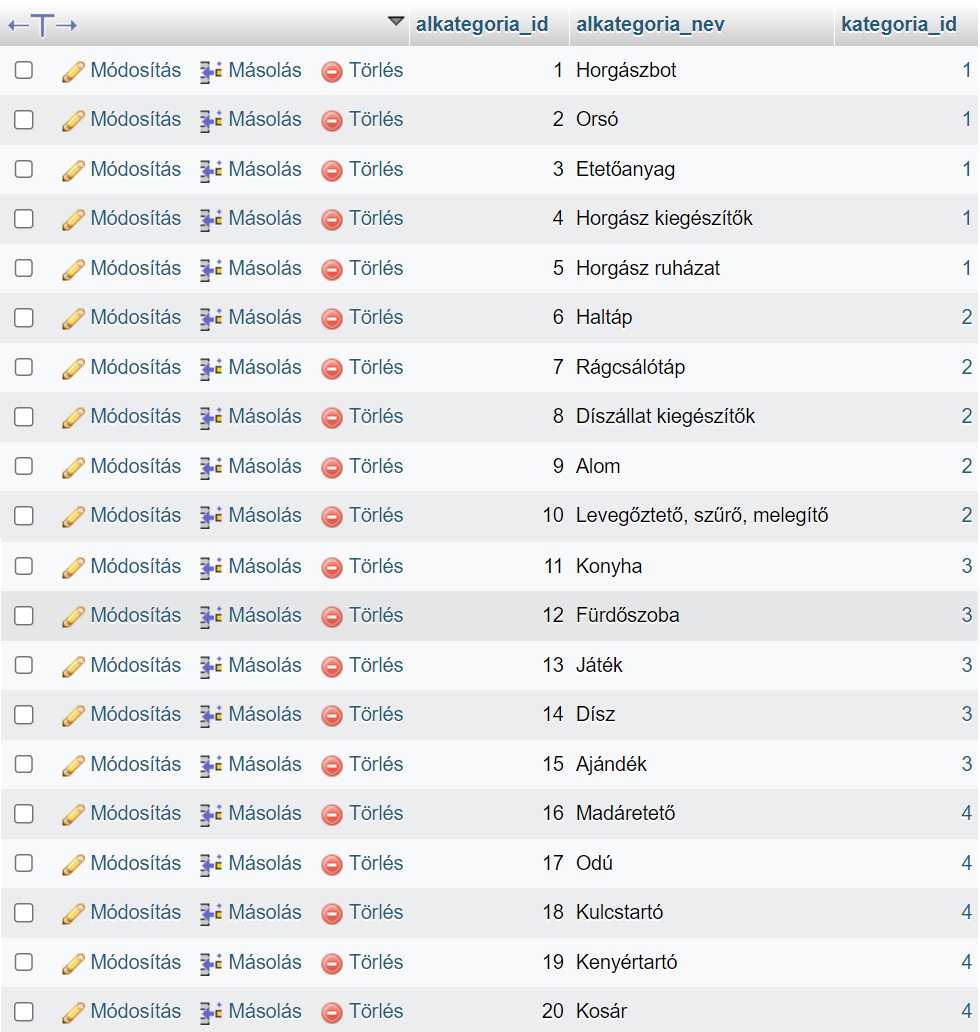


### 2.4.3 Alkategoriak tábla

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 táblát 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) |



### 2.4.4 Megrendeles tábla ! kép

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 tárolódig el az adatbázisban.

Tábla tulajdonságai:

|  |  |  |
| --- | --- | --- |
| Név | Leírás | Típus |
| **id** | azonosító tulajdonság, elsődleges kulcs | int (11) |
| **megrendeles\_datuma** | a megrendelés leadásakor elmentett dátum | date |
| **kezbesites\_datuma** | a termék kézbesítése után mentett dátum | date |
| **rendelt\_darab** | a rendelt termékek darabszáma | int (5) |
| **szemelyek\_id** | idegen kulcs, a szemelyek 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) |

Példa adatok: {kép beszúrása}

### Személyek tábla

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

cím: egy táblában tároljuk , nem szedtük szét többre , továbbfejlesztés , leírni a mostaniba mi a hiba, de most a feladatunknak megfelel ez a módszer

Tábla tulajdonságai:

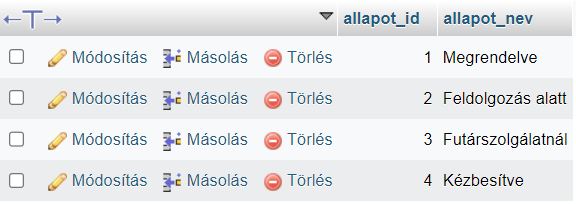
|  |  |  |
| --- | --- | --- |
| Név | Leírás | Típus |
| **id** | azonosító tulajdonság, elsődleges kulcs | int (11) |
| **nev** | a személy neve | varchar (255) |
| **felh\_nev** | a személy egyedi felhasználóneve | varchar (255) |
| **email** | a személy e-mail címe | varchar (255) |
| **jelszo** | 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) |
| **iranyitoszam** | a személy által megadott irányítószám | int (4) |
| **szallitasi\_cim** | a személy címe, utca ,házszám | varchar (255) |
| **telepules** | a személy által megadott település | varchar (120) |

### **Rendeles**\_allapot

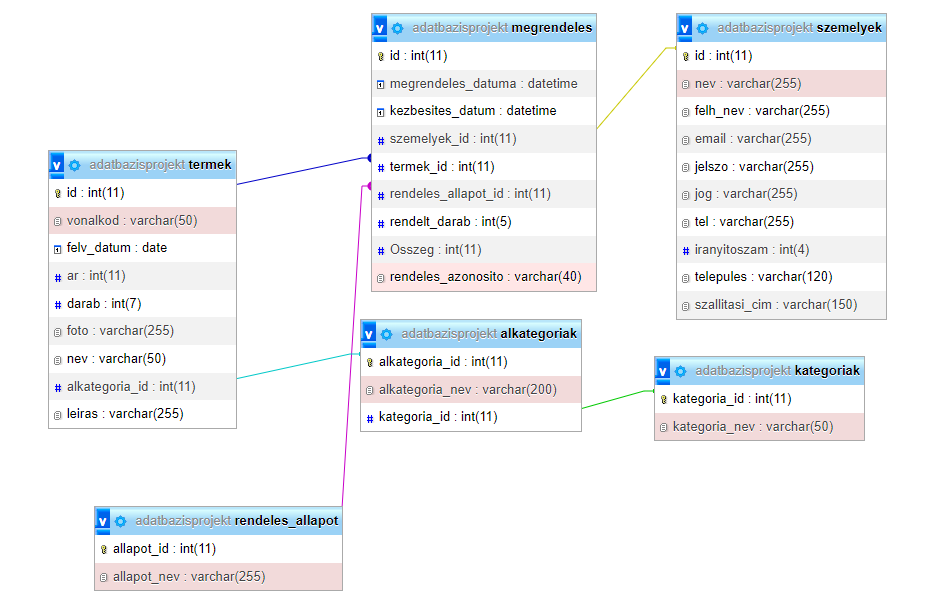
Leírás: A rendeles\_allapot táblába csak egy id és egy leíró mező foglal helyet. A leíró mezőben csak a rendelés státuszának a nevét tároljuk, mint például: megrendelve, feldolgozás alatt, futárszolgálatnál, kézbesítve.

Tábla tulajdonságai:

|  |  |  |
| --- | --- | --- |
| Név | Leírás | Típus |
| **id** | azonosító tulajdonság, elsődleges kulcs | int (11) |
| **allapot\_nev** | a rendelés státusza: megrendelve, feldolgozás alatt, futárszolgálatnál, kézbesítve | varchar (255) |



## Adatbázis diagram



## Weboldal

A kezdő lapunk main.php néven került elmentésre. Azokon az oldalakon, ahol php és HTML kódok is szerepelnek egyaránt, ahol csak lehetséges volt igyekeztünk ketté választani 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.

### Kezdő oldal

Az oldal tetején megjelenő menü elkészítéséhez a Bootstrap keretrendszerét használtuk. Mellette a jobb oldalon, ha a felhasználó nincs belépve, akkor a belépés és regisztráció menüpontok jelennek. 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 ismételten leírni a sorokat, hanem egyszerűen egy include()-al csatoljuk, amely ha nem is éri el a fájlt, attól az oldal még be fog töltődni.

### Beléptető és Regisztrációs oldal

A belépés és a regisztrációs oldalt úgy készítettük el, hogy könnyedén kezelhető legyen. Kettő input mező található a belépésnél, első az e-mailért felelős, amit az input type email-el ellenőrzünk, ez egy beépített HTML funkció, ami képes arra ,hogy megvizsgálja azt a szöveget, amit bele írtak, és hibát ad válaszul, hogyha nincsen benne „@”, és ha nincsen előtte vagy utána karakter. A második bemenet mező a jelszóért felelős, ami ún. password típusú, ami eltakarja a jelszót csillagokkal („\*\*\*\*\*”). Ha kitöltötte a mezőket akkor a belépés gombbal tovább tud menni a webshop.php oldalra, de ha esetleg nincsen még fiókja, akkor a „Regisztráljon itt!” linkre kattintva átkerül, a regisztrációs oldalra ahol 9 input mezőt kell kitöltenie ami, tartalmazza a rendeléshez szükséges adatokat is „regisztrálok” gombra kattintva létrehozza a fiókot, ha esetleg van fiókja és a regisztráció oldalon a „Vissza” feliratra kattintva vissza tud jutni a belépés oldalra. A felhasználóink használhatják a webshopot, kereshetnek, és nézelődhetnek, de ha rendelni szeretnének terméket, akkor be kell regisztrálnia vagy belépni már meglévő fiókba. A regisztrációnál meg kell adnia a szállítási címet, ezt később a profil fülnél módosíthatjuk, ha esetleg megváltozna a regisztráció óta. Az a megoldás, amit alkalmaztunk ,hogy a vásárlónak csak akkor kell regisztrálnia, amikor már rendelni szeretne sokkal felhasználó barátabb és üzleti szempontból is jóval praktikusabb.

### Webshop működése

A webshop használatát úgy találtuk ki, hogy felhasználóbarát legyen, ne kelljen regisztrálni ameddig, böngészik a felhasználó. Amikor valamit a kosárba tesz, azt tároljuk pillanatnyilag a local Storagéba, és amikor kattint a megrendelés gomba, átvezeti, a felhasználót egy olyan oldalra ahol látja azt a terméket, amit rendelt és a darabszámát, ha ott rákattint a „Küldés” gombra, akkor történik egy ajax hívás, ami visszatér egy bool-al igaz lesz vagy sem. Ha sikeres a rendelés akkor az adatbázisból kivonja azt a darabszámot, amit rendelt, és frissíti a raktáron lévő termék számát, és törli a local Storagéból. Lehet választani kategóriák, alkategóriák között is ,erre készítettünk egy SQL lekérdezést az adatbázisból, ami PHP-ban jeleníti meg a termékeket az oldalon.

A kosár működését javascriptel oldottuk meg. A script már akkor kezd lefutni mikor az oldal betölt (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 kiírtva, hogy sikeresen eltároltuk-e a változóba.

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

//console.log(kosarikon);

A kosár ikon egy eseményfigyelőt adtunk hozzá, amely két paramétert vár. Az első paramétere az, hogy mit figyeljen a második pedig, hogy mi történjen amikor az 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 meghatároztunk, hogy a kosár tartalma egy felugró ablakban megjelenjen. Hasonló képpen 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 értékként 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 végig iterál 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 kattintás 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, hogyha 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,

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),

termek\_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 értelem szerűen 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 2 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, ha a tároló nem üres, hiszen az az életszerű ha a vásárló több terméket tehessen a kosárba, akkor is ha nem egy időbe 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 hiba üzenet 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;

});

kosarikonP.innerHTML = no;

//Kosár feltöltése.

const cardBoxTable = document.querySelector('table');

console.log(cardBoxTable);

let tableData = '';

//fejléc

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>';

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>';

}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>';

})

}

let sum = 0;

JSON.parse(localStorage.getItem('cuccok')).map(data =>{

sum += data.darab \* data.price;

});

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>';

cardBoxTable.innerHTML =tableData;

}

/\*\*Törlés a kosárból\*/

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();

}

function deleteall(){

let cuccok = [];

JSON.parse(localStorage.getItem('cuccok')).map(data =>{

cuccok.splice(data);

});

localStorage.setItem("cuccok", JSON.stringify(cuccok));

window.location.reload();

}

function megrendeles(){

window.location.href = "megrendeles.php";

}

// Kereső

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";

}

}

}

function search\_itemadmin() {

let input = document.getElementById('searchbar').value

input=input.toLowerCase();

let x = document.getElementsByClassName('nev');

for (i = 0; i < x.length; i++) {

if (!x[i].innerHTML.toLowerCase().includes(input)) {

x[i].parentElement.classList.add("hidden");

}

else {

x[i].parentElement.classList.remove("hidden");

}

}

}

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");

});

}

modal.children[1].addEventListener("click", () => {

modal.classList.add("hidden");

});

function profilinfo(){

if (document.getElementById("profilid").className == "profilinfo")

{

document.getElementById("profilid").className = "profilinfofel";

}

else {

document.getElementById("profilid").className = "profilinfo";

}

}

function megrendelve(){

if (document.getElementById("profilid").className == "profilinfo")

{

document.getElementById("profilid").className = "piros";

}

else {

document.getElementById("profilid").className = "profilinfo";

}

}

function deleteProductAmount(termek\_id) {

let product = localStorage.getItem("cuccok");

$.ajax({

method: "POST",

url: "../api.php",

dataType: "JSON",

data: { c: "deleteProductAmount", termek\_id: termek\_id, darab\_szam: $("#productNumber\_"+termek\_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));

}

})

### Beléptetés (titkosítás, sha256)

Amikor a felhasználunk, regisztrál akkor a jelszavát sha256 módon titkosítjuk, az azt jelenti, hogy adatbázisban nem jeleníti meg a jelszóját hanem, random generál egy 256-bit (32 bytos) sorozatot. Ha sikeres a belépés vagy a regisztráció, akkor az oldal csinál egy automatikus ellenőrzés ,hogy aki be szeretne lépni ,az admin jogosultsággal vagy esetleg userrel rendelkezik, az oldalba beépített kód tovább irányítja a megfelelő felületre, webshop.php vagy az adminlist.php-ra. A felhasználóinkat tároljuk az adatbázisban ,eltároljuk a sorszámát, nevét, felhasználónevét, email-t és a rendeléshez szükséges adatokat.

### Listázás

A listázást admin és user felületen is használjuk, admin felületen a termékek megjelenítésénél több információt jelenítünk meg, hiszen admin-nak több joga van. User csak a termék nevét, darabszámát, árát és a leírását látja. Úgy oldottuk meg a kategóriáknál, a megjelenítést ,hogy a lekérdezés elejét required-be tettük ,hiszen azt mindig egyforma lesz de tettünk minden oldal elejére egy szűrést ami a kategóriákra vonatkozik, és csak azokat jeleníti meg amire megfelelő a szűrés.

### Felvitel

A felvinni terméket csak az admin jogosultsággal rendelkező fiókok képesek. adminlist.php oldalon van egy „Új áru felvitele” gomb ami tovább vezet egy felvitel.php-re ahol megjelenek a mezők amit kikell tölteni egy termékhez ,Fotó feltöltése szükséges amit input type file-ként kérünk be ,ezért az látható ,hogy Fájl kiválasztása erre kattintva megjelenik a fájlkezelő mappa, és lehet választani melyik képet akarjuk a termékhez csatolni. Tettünk több fajta ellenőrzést is , hogy nem lehet egy bizonyos méretnél nagyobb a fájl ,csak png,jpg, és gif lehet beküldeni, A \* jelölt azok amit kikell tölteni ahhoz ,hogy egy termék érvényes legyen. Ha egy mezőt üresen hagyunk, akkor a feltöltés sikertelen lesz és felugrik egy „error-msg” ami figyelmeztet arra, hogy nem lehetnek üresek, mindet ki kell tölteni. Amikor a kategóriát választjuk ki egy selectes legördülő menü jelenik meg és ott kitudjuk választani a megfelelőt ,ez sokkal jobban felhasználó barátabb és praktikusabb, és kevesebb a hiba lehetőség, esetleg arra, hogy elgépeljenek valamit. A felvitel dátumát egy date type típusú input alakítottuk ki ,és ezt lehetőséget adott arra ,hogy a jobb szélén megjelent egy naptár, ami könnyedén kezelhető, és választható a megfelelő nap. Az admin-nak van lehetősége arra, hogy a termékhez adjon egy rövid leírást, amit lekorlátozunk 200 szó-ra. Itt is van lehetősége a felhasználónak a visszajutáshoz, ha esetleg rossz helyre vagy másik terméket szeretné módosítani a „Vissza” felirat segítségével.

### Módosítás

A módosításnál, egy PHP felületet választottunk ahol adatbázisból már meglevő adatokat ki „printeltettünk” a megfelelő mezőbe, hogy a szerkesztés könnyedén végre hajtható legyen, a stílus szinte megegyezik a felvitellel, azokat az ellenőrzéseket amiket a felvitelnél alkalmazunk, azok itt is megegyeznek, tehát nem tud a felhasználó olyan adatra módosítani ami nem létezik, vagy esetleg ellenkezik a szabályokkal.

### Törlés

A törlést admin joggal rendelkező fiókok tudják csak használni, mivel csak az adminlist.php érhető el ez a funkció, ezt egy sima DELET FROM-al oldottuk meg ,ez azt tette lehetővé ,hogy szimplán töröljük az adatbázisból, és amikor vissza irányít az adminlist-re akkor újra lefut az SQL lekérdezést, csak ekkor már az amit töröltünk az nem fog megjelenni. Az oldalt lapvédelemmel ellátok, hogy fixen legyen felhasználó belépve, ezt egy sima SESSION-al oldottuk meg, egy IF elágazásba, ha nincsen ’belepett’ tehát nem történt meg a belépés, akkor irányítsa tovább a false.html-re, ez azt eredményezi, hogy ne lehessen külső oldalról elérni a PHP oldalt, és idegenek ne tudjanak bele szerkeszteni az adatbázisba.

### 2.7. Tesztelés

A weboldalaink HTML kódrészleteit a<https://validator.w3.org/> oldalán ellenőriztük. Találtunk néhány hibát, amelyet nem vettünk észre, mivel az oldal láthatólag 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 VSC 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> oldalán is ellenőriztük, itt már hibát nem találtunk, mivel a programozás során is folyamatosan ellenőriztük, print\_r-el, vagy var\_dump()- kiíratva folyamatosan a kért és kapott adatokat.

### Kereső optimalizálás

# Felhasznált technológiák

## 4.1 Frontend eszközök

A frontend a programoknak, 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 a tartalomért felelős, amig a JavaScript és a CSS a megjelenítést és a tartalom kinézetét biztosítják.

A weboldalunk fejlesztéséhez a Visual Studio Code állt rendelkezésünkre. Ez egy nyílt forráskódú , ingyenes kódszerkesztő, amelyet a Microsoft fejleszt. A hibakeresőket támogatja , ami a mi munkánkat is megkönnyítette. Képes az intelligens kódkiegészítésre (Intelli Sense). Rendelkezik beépített Git támogatással , ami elengedhetetlen, ha csoportban szeretnénk dolgozni egy projekten.

## 4.1 HTML5

Webfejlesztés folyamán az első nyelv, amivel találkozunk a HTML, vagy Hyper Text Markup Language.

## 4.2 CSS

A Cascading Style Sheet (CSS) a HTML elemek formázására használunk. 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 tovább fejlesztése 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.

## 4.3 JavaScript

# Összegzés

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 rengetek 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. Az 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. De ehhez természetesen szerződnünk kell majd különböző szolgáltató cégekkel, akik a szállítást végzik, illetve amin keresztül a fizetés történhet.

# Források:

minden weboldal linkje, amit megnézel a munka megoldásához

Író, cím, link

excel táblában gyűjtve