

RUANDER Oktatási Kft.

ZÁRÓDOLGOZAT
PIZZARENDELŐ FELÜLET

Készítette:

Farkas Bence László

Web-fejlesztő

Budapest

2022

Tartalomjegyzék

1. Bevezetés.....	4
1.1. Projekt célja.....	4
2. Az oldal	6
2.1. Rendszerkövetelmények.....	6
2.2. Az oldal elérése	6
2.3. A weboldal lehetőségei	6
2.4. Az oldal kinézete	6
2.5. Az oldal navigációja.....	7
2.6. Termék rendelés	8
2.7. Use Case diagramm.....	9
3. Admin felület.....	10
3.1. Login.php	10
3.2. Dashboard.php.....	11
3.3. Orderlist.php.....	11
3.4. Order-details.php	11
3.5. Users.php.....	12
3.6. Userdata.php.....	12
4. Az adatbázis	13
4.1. users tábla.....	13
4.2. Pizza tábla.....	15
4.2.1. Pizzas_toppings tábla	15
4.2.2. Toppings tábla tábla	15
4.3. Sizes tábla.....	16
4.3.1. pizzas_sizes tábla.....	16
4.4. orders tábla	16
4.4.1. order_pizzas tábla.....	17
4.5. Új termék bevitele	17
5. Fejlesztői dokumentáció.....	19
5.1. A HTML és a CSS.....	19
5.2. A képek feldolgozása	20
5.3. Fájl szerkezet, a weboldal részei	20
5.3.1. Registration.php	23
5.3.2. Login.php	24
5.3.3. Logout.php	25

5.3.4. Account.php	25
5.3.5. Profile.php	26
5.3.6. Product_list.php.....	26
5.3.7. Product.php.....	27
5.3.8. Checkout.php.....	27
5.3.9. Checkout.js	28
5.3.10. init.js	30
5.3.11. shopping-cart.php	32
5.3.12. shopping-cart.js	32
5.3.13. Orders.php	33
5.3.14. Order-completed.php.....	34
5.3.15. Ajax.php	34
6. Az oldal elérése	37
7. Felhasználói dokumentáció:	38
7.1. Rendelés	38
7.2. Regisztráció	42
7.3. Belépés	43
7.4. Admin bejelentkezés	44
7.5. Userek.....	46
7.6. Megrendelések.....	47
8. Tesztelési dokumentáció	49
9. Továbbfejlesztési lehetőségek	54
9.1. Elfelejtett jelszó	54
9.2. Rendelés esetén e-mailes tájékoztató kiküldése a felhasználó felé	54
9.3. Új pizza felvitele adatbázis használat nélkül.....	54
10. Összegzés	55
11. Forrásjegyzék	56

1. Bevezetés

A tanfolyam ideje alatt az eddigi érdeklődésem a webfejlesztés iránt nagyot nőtt, köszönhetően az Oktatók által átadott tudásnak. A mai világban, ahol minden az információs technikára épül, nagyon fontosnak tartom, hogy egy adott weboldal megfelelően készüljön el, jó alapokon álljon, hasznos legyen egy felhasználó számára. Weboldalak bonyolultságától függetlenül elmondható, hogy a felhasználó oldalról nézve a legfontosabb dolog, hogy egy adott weblap érthető legyen, könnyű legyen azon eligazodni és használni azt. Mindezen indokok alapján választottam a pizzarendelés felületet, hiszen a mai világ szerves részét képezi az étel rendelés.

1.1. Projekt célja

A célom a záródolgozat témájának megválasztásakor az volt, hogy egy jól használható, felhasználóbarát weboldalt készítek. Ebben nagy szerepet játszott az is, hogy családon belül van olyan személy, aki rendelkezik pizzériával, ezért a választásom kézenfekvő volt. Szerettem volna egy olyan weboldalt készíteni, ami egyszerűvé teszi a megrendelés leadását és céljaim közé tartozik az is, hogy az étterem honlapja stabilan működjön a felhasználók számára. Sokat gondolkodtam a megvalósításon, hogy milyen is lehet egy ideális webshop, ami után sokat kutattam és használatban lévő webshopokat tanulmányoztam. Végül eljutottam oda, amit záródolgozatombként készítettem.

Azt vettem alapul, hogy a lehető legegyszerűbb módon lehessen a kiválasztott terméket a kosárba helyezni és azt megrendelni. Manapság a felgyorsult világunknak köszönhetően rengeteg inger éri az embereket, mindemellett sokaknak kevés idejük van arra, hogy lassú weboldalakon keresztül bonyolítsanak le egy rendelést. Arra törekedtem, hogy a weboldal kinézete feltűnő és egyben igényes legyen, emellett szerettem volna, ha a felhasználó az első alkalomkor tájékoztatást kap az adott webshopról - jelen esetben az étteremről - és már egyből megjelennek a legfelkapottabb termékek is. Magamból kiindulva szeretem, ha egy oldal átlátható, egyszerű és letisztult. Napjainkban a felhasználók számára egy webshop felépítése úgy néz ki, hogy baloldalt egy nagyobb kép látható a megvásárolni kívánt termékről, jobboldalt pedig az ár, amely alatt a mennyiségválasztás, egy rövid termék jellemző és végül a kosárba gomb található. Sok kutatás és érdeklődés után rájöttem, hogy az ettől eltérő webshopokat a felhasználók nehezebben értelmezik és nem szívesen térnek vissza azokra az oldalakra, amik a fentebb leírtaktól jóval eltérőbbek. A webshop színvilágnak kiválasztásakor törekedtem arra,

hogy a lehető legjobban illeszkedjen a pizzák képeihez, ezért döntöttem a sárgás-pirosas téma mellett. Úgy gondolom, hogy egy letisztult és egyben jól kinéző oldal vonzza a vásárlókat. Az oldal tetejét, ahol a fő gombok találhatóak, igyekeztem minél figyelem felhívóbbá tenni, hogy ne kelljen keresgélni azt, hogy hol is van adott esetben a bejelentkezés, illetve a regisztráció funkció.

Amellett a tény mellett sem szabad elmenni, hogy a webshopon feltüntetett képek legyenek jó minőségűek, illetve figyelemfelkeltőek. Mivel itt ételeket szeretnénk eladni, fontos az is, hogy meghozzák az emberek kedvét az ételhez és egyfajta vágyat ébresszünk bennük, ezért is törekedtem a legjobb képek felhasználására az oldalon. Viszont, amire érdemes figyelni, hogy ne legyen nagyon pixeles egy kép, de ne is vigyük túlzásba, mert a nagy felbontású képeket nehezebben tölti be az internet, ami azt fogja eredményezni, hogy veszítünk a felhasználói élményből. A legjobb megoldás az az, ha egy kép 200-300 kilobájt között van. Mindemellett a felhasználókat ne untassuk fölösleges tartalmakkal, mert nem szívesen olvasnak 3 bekezdésnél többet, emiatt alkalmaztam a rövid tömör megfogalmazásokat, ami ellátja a felhasználót elegendő információval.

Végezetül figyelni kell arra, hogy megfelelő erőforrással rendelkezünk. Amennyiben egy sikeres webshopot készítünk, amit rengetegen fognak használni és számtalan rendelés érkezik be az oldalról, törekedni kell ezeknek az igényeknek a kielégítésére, hogy éjjel-nappal a maximálisan elégedettek legyenek a felhasználók és ők továbbra is használják az általunk készített oldalt.

Miért jó egy egyszerűen kezelhető weboldal, ahol az emberek rendeléseket tudnak leadni? Azért, mert ezzel a felhasználó időt spórol, nem szükséges sorban állnia az ételért, egyszerűen bárholnan intézheti a megrendelését, mivel az oldal reszponzív, ezért mobil eszközről is könnyen elérhető mindenki számára.

2. Az oldal

2.1. Rendszerkövetelmények

A program Windows 10-en lett fejlesztve, szóval ezen működik megfelelően, illetve szükséges, hogy a számítógép rendelkezzen internetkapcsolattal is. A futtatáshoz szükség van a XAMPP nevű programra, amin futtatjuk az Apache-ot és a MySQL-t is. A fejlesztés során Google Chrome-ot használtam, illetve a Mozilla Firefoxban is végrehajtottam néhány tesztet az oldalon. Azt javaslom, hogy azokban a böngészőkben tekintsük meg az oldalt.

2.2. Az oldal elérése

Jelenleg az oldal nem publikus és csak lokálisan érhető el, amiben a Xampp nyújt segítséget.

Abban az esetben, hogy ha a Xampp-on belül futtatjuk az Apache és a MySQL szerveret, akkor fogjuk tudni elérni a weboldalt. A Xampp-on belül a „htdocs” nevű mappába illesszük be a dokumentumot, mivel abban tudjuk megnyitni a programot.

Elérési út: localhost/pizzarendelo_bead vagy http://127.0.0.1/pizzarendelo_bead/

2.3. A weboldal lehetőségei

Ebben a fejezetben szeretném kifejteni az oldalon elérhető funkciókat. A legfontosabb az, hogy az oldalon pizzák rendelését lehet leadni, melyet az adatbázisban kapunk meg az admin részen. Az oldal lényege tehát, hogy a felhasználó rendelést közvetít a pizzéria felé, amit a pizzéria megkap és el tudja készíteni a beérkezett megrendelést.

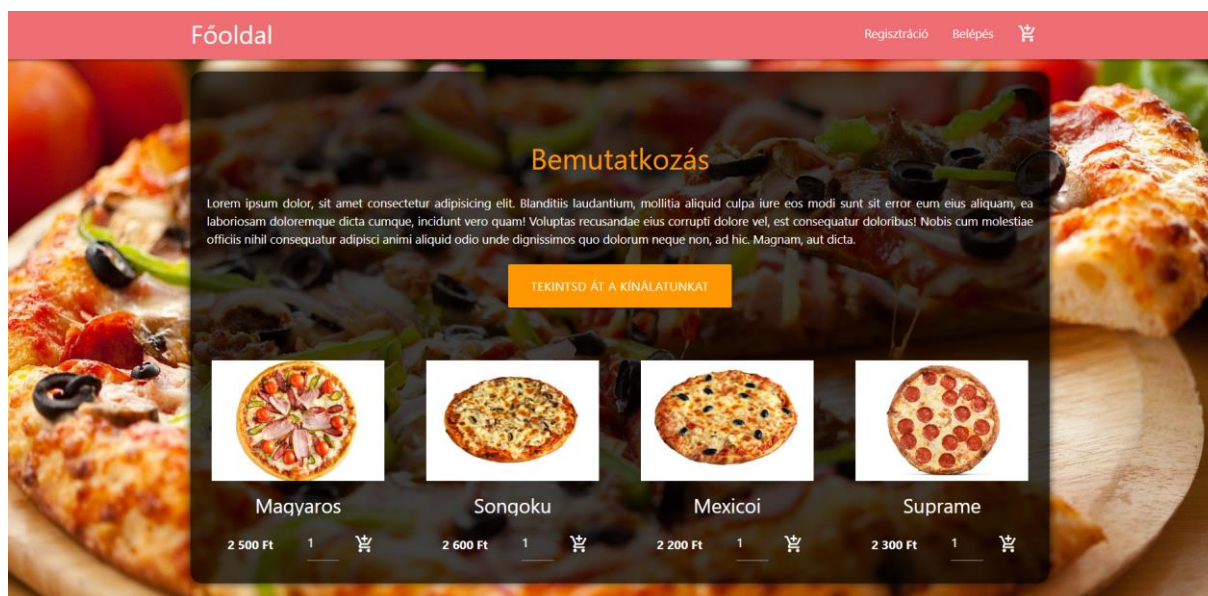
2.4. Az oldal kinézete

Az oldal megjelenítésében nagy szerepe van a Materializenak, mivel úgy döntöttem, hogy ezt a template-t fogom alkalmazni a weboldalon. Ennek több oka is volt: az első indok az, hogy könnyen kezelhető. Második indok, hogy könnyen letölthető a <https://materializecss.com/> oldaláról és egyszerű implementálni a HTML kódba. Mindezek mellett kompatibilis a css-el és

a JavaScriptel is, ezért nem áll távol a működése a Bootstrap-tól sem.¹ Ezt követően kisebb módosításokat hajtottam végre a Materialize témájában, úgy, mint a színek módosítása, kiírások szerkesztésére, illetve a számomra feleslegesnek tűnő elemek törlésére. A Materialize honlapján megtalálhatóak számos ikonok is, amik díszesebbé teszik az oldalt, ezek közül alkalmaztam számos ikont, mint pl.: bevásárlókosár, checked jelzés, dollár jel, kuka ikon. Összefoglalva a meglátásaim szerint egyszerű használhatóság mellett meglehetősen dizájnos megjelenítést ad egy weboldalnak.

2.5. Az oldal navigációja

A felhasználó, aki felkeresi az oldalt, nyitó oldalként ezzel a képpel szembesül:



A bemutatkozás szövegrészhez az adott pizzéria történetét szabadon választva be lehet írni. Alatta pedig a legfelkapottabb pizzák képeit láthatjuk, amennyiben valamelyike megtetszik nekünk, akkor könnyedén, válogatás nélkül tudjuk leadni a megrendelésünket. A „Tekintse át a kínálatunkat” gombra kattintva kapjuk meg az étlapot, ahol jelenleg 4 darab pizza található. Ez természetesen több pizza esetén bővül.

A fejlécen található 4 darab gomb: „Főoldal”, amely visszavigyel minket a fenti képen látható oldalra. A „Regisztráció”, ahol be tudunk regisztrálni egy felhasználót. „Belépés” gomb már meglévő felhasználó belépését szolgálja az oldalra. Utolsóként a „Kosár ikon”, ami a jelenlegi kosarunk tartalmát mutatja meg.

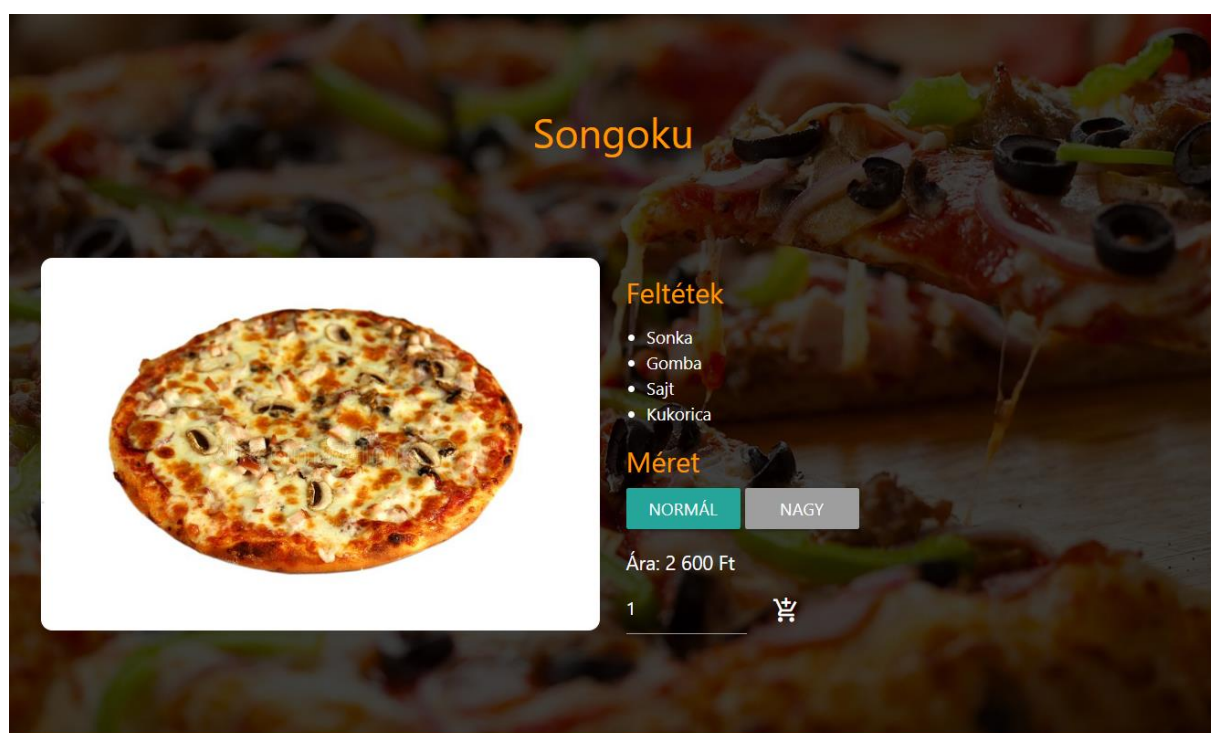
¹ <https://materializecss.com/templates/starter-template/preview.html>

Miután az oldalra bejelentkeztünk, akkor a fejlécben látható „Regisztráció” és „Belépés” át fog váltani, arra, hogy „Profilom” és „Kilépés”.



Az oldal használatának nem feltétele a regisztráció, azonban hasznos, mivel ennek segítségével láthatjuk a korábbi rendeléseket, illetve a belépési adatokat is lehet módosítani az oldalon.

Abban az esetben, ha rákattintunk egy pizza nevére, akkor egy részletesebb leírást kapunk a pizzáról, mégpedig azt, hogy milyen feltéteket tartalmaz az adott pizza, illetve még azt is, hogy mekkora méretben szeretnénk a pizzát megvásárolni.



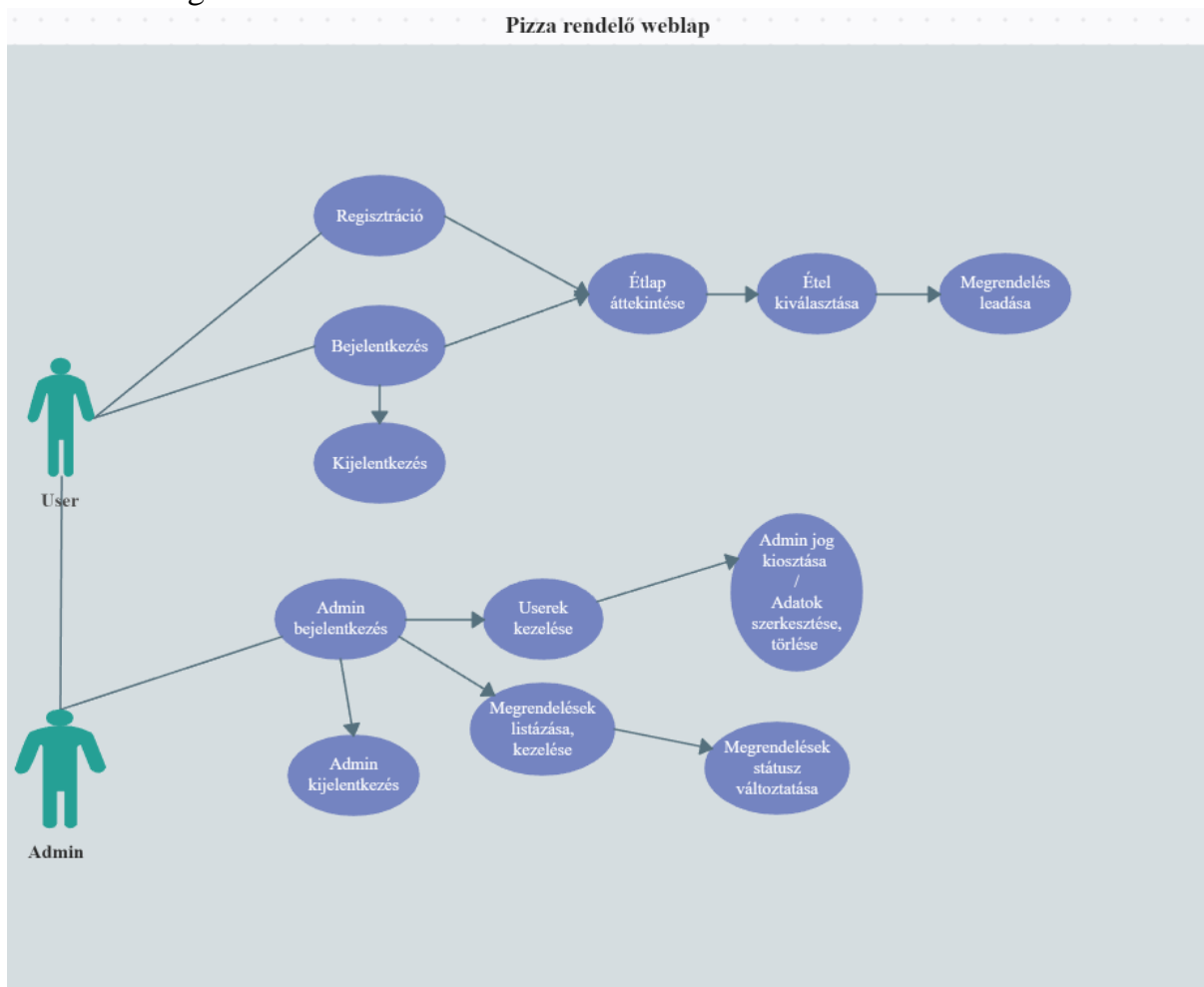
Alapértelmezetten a pizzánk mérete „NORMÁL” méretű, azonban, ha nagyobb pizzát szeretnénk rendelni, arra is van lehetőség, hiszen a „NAGY” gombra kattintással módosítjuk a kiválasztott pizza méretét, ahol a pizza ára is változni fog. Nem csak egy darab pizza rendelésére ad lehetőséget az oldal, hanem akár több terméket is a kosárba tudunk tenni, mivel az elmenti a kosár tartalmát mindaddig, amíg az oldalon maradunk és meg nem rendeljük a termékeket.

2.6. Termék rendelés

Amennyiben a kosár tartalmára kattintunk, akkor a „PÉNZTÁR” gomb jelenik meg, ahol a kosár tartalmát összegzi nekünk a weboldal. Itt tudunk még módosítani a darabszámon, illetve el tudjuk távolítani a nem kívánatos terméket a kosarunkból. Amennyiben minden megfelel

számunkra, akkor kattinthatunk a „TOVÁBB A PÉNZTÁRHOZ” gombra, ami elnavigál minket a megrendelés véglegesítéséhez. Itt meg kell adjuk az adatainkat, amik a megrendeléshez szükségesek és ki kell válasszuk azt is, hogy érte megyünk, vagy kiszállítást kérünk. A vásárlás folytatása gomb megnyomásával véglegesítjük a megrendelést.

2.7. Use Case diagramm



3. Admin felület

Az admin felület elengedhetetlen volt a fejlesztés során, mivel amikor a megrendelő megrendel egy terméket, azt az admin felé valamilyen úton módon el kell juttatni, hogy észlelje a megrendelést. Így arra törekedtem, hogy az admin oldalon megjelenítsük az eddigi megrendeléseket a „megrendelések” menüpont alatt, és amiket le is tudjunk kezelni, legalább olyan szinten, hogy elkészült státuszba tudjuk helyezni az adott rendelés állapotát. Ezzel jelezvén, hogy azzal a megrendeléssel, ami zöld háttérrel kapott már nincs több teendőnk. A megrendelések menüpontban a terméket és a terméken szereplő egységeket részletezzük, például valamely pizzán lévő feltéteket. A megrendelt pizzák darabszámait és méreteit is nyomon tudjuk követni. Fontos szempont volt az is, hogy hova szól a megrendelés, ezért kiírunk minden ahhoz szükséges információt, hogy a pizza megérkezessen ahhoz a személyhez, aki ezt rendelte, abban az esetben, ha a házhozszállítás opciót választotta. A megrendelés közben a szállítási módok között tudunk választani, személyes átvétel és házhozszállítás opció között választhat a megrendelő. Ha valaki a házhozszállítást választja, akkor kiírjuk az admin részére azt is, hogy az adott pizzát hova kéri szállításra a megrendelő. Ha a felhasználó a személyes átvételt választja, akkor azt is megjelenítjük az adminnak, hogy az előkészítésen kívül nincs más teendő, tehát nem kell a futárnak átadni.

Az admin felületen továbbra megtalálható még egy menüpont is, mégpedig a „Userok” gomb. A „Userok” menüponton belül fogjuk látni a regisztrált felhasználóinkat, illetve felhasználóknak tudunk admin jogot is kiosztani. Ennek az a lényege, hogy több embernek is tudjunk kiosztani admin jogot, hogy több személy is hozzá tudjon férni az admin felülethez. Gondoljunk csak bele, ha egy étteremben dolgozók között többen töltik be azt a szerepet, ahol a munkájuk megköveteli a rendelések eljuttatását a számítógéptől a szakácsig, ezt követően a szakácstól a megrendelőig, akkor erre biztosítanunk kell egy lehetőséget. A dolgozók közül, akik ezt a feladatot látják el, admin joggal kell rendelkezniük, hogy a saját felhasználói adataikkal tudjanak belépni a felületre. Az összes admin felületre igaz az a tény, hogy csak abban az esetben érhető el a felhasználó számára, ha rendelkezik admin joggal!

3.1. Login.php

Az admin felületre való belépésre csak annak van joga, aki a „Users” nevű táblában az „is admin” cellán belül el van látva 1-es értékkel, tehát admin minősítéssel rendelkezik az adott profil. Akinek a neve mellett „0” szerepel, nem tud az admin oldalra fellépni. Ezt az admin

minősítést ellenőrzi a kód, mielőtt elkezdené a küldött adatokat vizsgálni. Amikor megnyomásra kerül a bejelentkezés gomb, akkor történik meg a vizsgálat. Megpróbálja megtalálni azt a felhasználót, akinek az e-mail címe és jelszava egyenlő a mezőkbe beírt értékekkel. Mivel a felhasználó jelszava md5-ös formátumban van tárolva, ezért a vizsgálat során a beírt jelszót is átalakítja md5-re és így történik meg az ellenőrzés. A hash-eket párosítja össze, lényegében hash-ekben keres. Megszámolja a találatokat, vagy 1, vagy 0. Ha 0, akkor nem talált ezekkel az adatokkal felhasználót, akkor hibaüzenet rögzítés történik a sessionben, plusz oldal frissítés is történik egyben, azért, hogy ne legyen ott a hibaüzenet. Ha van ilyen találat, akkor ráhelyezi a session user-re a talált felhasználónak az összes adatát és abból lesz a profil.

3.2. Dashboard.php

A dashboard.php az admin kezdőoldala, itt tulajdonképpen el van helyezve kettő darab gomb. Az egyik a „Userék”, a második pedig a „Megrendelések”.

3.3. Orderlist.php

Az orderlist.php-n az admin a megrendelések beérkezését tudja nyomon követni. Az oldalt csak akkor tudja megtekinteni a felhasználó, ha admin joggal rendelkezik, ellenkező esetben elérhetetlen lesz számára a felület. Olvasható, értelmezhető nevet ad a kiszállítási kulcsoknak, valamint az adatbázisban eltárolt kiszállítás kulcsainak. Lekérésre kerülnek a felhasználók korábbi rendelései, amiket meg is jelenít, illetve részletez a kód. Ezt követően áttekintő adatokat fog látni az admin, a megrendelés leadásának dátuma és szállítási módja, valamint a rendelés ára is megjelenítésre kerül. A felület azt is kezelni tudja, ha elkészül egy rendelés, akkor az admin saját maga számára megkülönböztetést tesz azokra a rendelésekre, melyek elkészültek.

3.4. Order-details.php

Az order-details.php az orders.php-n belül található, abban az esetben, ha az admin rákattint egy beérkezett megrendelésre. A felület kijelzi az admin számára, hogy ez hányadik megrendelés, mivel csökkenő sorrendbe van rendezve a táblázat, azaz legfelül mindig a

legutolsó beérkezett rendelés található. A felületen tulajdonképpen az admin a megrendelés részleteit tekintheti át. Lekérésre kerül az adatbázisból a kiválasztott megrendelés ID-ja. Azt is látja az admin, hogy ez a megrendelés mikor lett beküldve, illetve, hogy milyen szállítási módot választott a megrendelő és azt is, hogy az adott megrendelésnek mennyi az ára. Az adott vásárlás részleteinek a lekérdezése konkrétan a kapcsolódó termékeknek a listázása. Ez csak annyit csinál, hogy megkeresi, hogy ahhoz a vásárláshoz mik voltak a megrendelések, milyen termékeket tartalmaz a vásárlás. Arra az esetre létre lett hozva egy checkbox, ha egy rendelés elkészült, akkor az admin azt be tudja pipálni és a rendelés státuszt fog váltani, ami szemmel láthatóan is megtörténik, mivel zöld háttérrel fog kapni az orderlist.php-n szereplő elem. Mindezek mellett az adatbázisban is befrissül az ordersek táblán belül „is_ready” mező és a 0-ás érték 1-es értéket fog kapni, ezzel is jelezve, hogy elkészült a megrendelés.

3.5. Users.php

A(z) users.php-n belül ki listázásra kerülnek az oldalra be regisztrált felhasználók. Az sql nevű változó végig megy a users táblán és ki listázza azokat, akik ott szerepelnek. Arra is ad információt az adminnak, hogy egy adott user rendelkezik-e admin jogosultsággal, vagy sem. A kódban el van helyezve egy úgynevezett „Adatok” gomb is, aminek a segítségével eljuttatja az admint a felhasználók adataihoz, ahol lehetőség van az adatok módosítására.

3.6. Userdata.php

Ezen a felületen van lehetőség arra, hogy az admin szerkeszteni tudja a felhasználók adatait. Csak akkor tudja ezt az oldalt elérni a felhasználó, ha rendelkezik admin joggal. Amennyiben nem rendelkezik vele, akkor elérhetetlen lesz számára az oldal. Tulajdonképpen ezen a felületen történik meg minden szerkesztés, ezért szükséges az adatbázis kommunikáció is, mert ha átírjuk a felhasználó vezetéknévét, akkor az a users táblában frissítésre kerül. Ugyan így igaz a törlésre is, ha bepipálja az admin a checkboxban azt, hogy törölni kívánja a felhasználót, akkor egy „delete” sql parancs fut le, a kiválasztott id alapján. A kódban szerepelnek feltételek, ugyanúgy, mint az űrlapon a módosításnak, meg kell felelnie néhány kritériumnak. Ilyen kritériumok a következők: minimum 2, maximum 30 karakter a vezetéknév és a keresztnév is. Az e-mail cím validálás ellenőrzése is itt található meg. Nem tud az admin olyan e-mail címet beállítani egy felhasználónak, ami már használatban van, mivel ezt is figyeli egy kódsor. Ha az errors tömb értéke 0, akkor megtörténik az adatbázis frissítés és bekerülnek az új adatok az adatbázisba.

4. Az adatbázis



Az adatbázist pizza adatbázisnak neveztem el, mivel ezt gondoltam a legátfogóbb névnek.

4.1. users tábla

- id: egyedi azonosító a táblán belül auto increment és primary key.

- `is_admin`: Itt tudjuk ellátni admin joggal a felhasználókat. Ha az „`is_admin`” értéke 0, akkor nincsen joga a „`system_admin`” belépéshez, amennyiben 1-es értéket kap, akkor be tud lépni a felhasználó a „`system_admin`” felületére.

Általános adatok:

- `first_name`: `last_name`: `e-mail`: `shipping_*(minden)`: `billing_*(minden, ami billingel kapcsolatos)`: → ez a vásárláskor megadott szállítási és számlázási adatokat tartalmazza.
- `created_at`: a user létrehozásakor automatikusan rögzíti a létrehozási dátumot.
- `updated_at`: a bejegyzés valamely értékének a módosítási dátumát automatikusan rögzíti
- `user` tábla általános info: Amennyiben vendégként vásárol egy felhasználó, bekerülnek a számlázási, szállítási és kapcsolattartási adatai a táblába. A `type` ez esetben `guest` lesz. Ha később valaki regisztrál egy olyan email címmel, amelyhez tartozóan már van rögzítve vendégként leadott vásárlás, akkor a regisztrálni kívánt személy, aki már nem `guest`, hanem egy `user`, a korábbi adatai hozzárendelődnek a regisztrálást követően a fiókjához.
- `email`: Letároljuk a felhasználó e-mail címét, amivel regisztrált az oldalra.
- `new_email`: A regisztrált felhasználó, ha új emailt ad meg, akkor az új email címe ide fog bekerülni ideiglenesen és ezt követően innen írja felül az email field-et. Az email cím visszaigazolását követően innen helyeződik át az email mezőbe és kiürül a `new_email`
- `email_verified_at`: További fejlesztés célja, ha a `mailer.php`-t hozzá rendelnénk a kódoz, akkor itt tudnánk nyomon követni azt, hogy a felhasználó mikor kattintott a visszaigazoló e-mailre. Jelenleg azt a dátumot írja be, amikor regisztrált a felhasználó.
- `created_at`: Mikor lett a regisztráció létrehozva.
- `password`: md5ös hasben tárolt jelszó.
- `remember_token`: ez jelenleg nincs használatban, további fejlesztés célja lehet. Egy titkos kulcsot tartalmazna, amellyel azonosítaná a felhasználót, abban az esetben, ha a felhasználó a megjegyzés opciót bepipálja belépéskor.
- `verification_token`: titkos kulcs, amely bekerül a regisztrációs emailbe, megerősítés céljából. A jelenlegi verzióban ez a funkció nincs használatban.

4.2. Pizza tábla

Itt a pizzák alapadatait tároljuk. Id, ugyan az, amint ami eddig is (lásd usersben).

- Url: az adott pizzához tartozó terméklap hivatkozása.
- Name: A pizza neve.
- Image: A pizzáról készült kép
- Description: Jelenleg nincs használatban.
- Hotdeal: Kiemelt ajánlat a kezdőoldali listák, ezekből állnak össze. 0, vagy 1 lehet az értéke.
- Idő bélyeg táblák: createdat és updtatedat.

4.2.1. Pizzas_toppings tábla

- id: Azonosítóként szolgál.
- pizza_id: Ez a kapcsolja hozzá a pizzas táblában lévő pizzával a feltétet.
- topping_id: ez kapcsolja össze a toppings táblában lévő feltéttel a pizzához társított topping_id-t
- creted_at, updated_at.

Egyes bejegyzések vannak Pizza_id mezőben, ha 1-es szerepel, akkor az azt jelenti, hogy a pizzas táblában lévő 1-es id-ú pizzához van társítva egy feltét. Az, hogy melyik feltét van hozzá társítva, a topping_id alapján van definiálva.

4.2.2. Toppings tábla tábla

- id: Azonosítóként szolgál.
- name: topping neve.
- created_at:
- updated_at

Itt az azonosító és az ahhoz kapcsolódó feltét neve a fontos.

Van két törzs tábla: Az egyik a pizzas, a másik a toppings. Ezek között van egy kapcsolatot definiáló tábla a pizzas_toppings. Ez csak kapcsolatot definiált számokon/azonosítókon keresztül.

4.3. Sizes tábla

- id: Azonosítóként szolgál.
- name: méretnek a neve.
- default: 1es vagy 0ás az értéke, ha 1es akkor az lesz a pizza alapértelmezett mérete
- created_at:
- updated_at

Itt az azonosító és az ahhoz kapcsolódó feltét név a fontos.

4.3.1. pizzas_sizes tábla

- id: Azonosítóként szolgál.
- pizza_id: pizza azonosítója.
- size_id: a méret azonosítója, a sizes táblában található adott méretnek az azonosítója.
- price: az adott pizzának az adott mérethez tartozó árát jelöli.
- created_at:
- updated_at:

A két törzstábla a pizzas és a sizes. Az egyikben a pizzák vannak definiálva a másikban meg a méretek. A közöttük lévő kapcsolatot a pizzas_sizes táblában definiáljuk.

4.4. orders tábla

Ide kerül be a vásárlásnak a globális adatai.

- id: Azonosítóként szolgál.
- user_id: ide kerül be a user id-ja.
- user_data: ide kerül be minden adat, amit a megrendelés közben megadott, vevő neve, szállítási cím, e-mail cím és a kiszállítás módja.
- shipping_price: a szállítás ára.
- delivery_method: hogyan igényelte a felhasználó a kézbesítést.

- total: teljes fizetendő összeget tartalmazza.

Ezt követően még vannak az időbélyegek.

4.4.1. order_pizzas tábla

- id: Azonosítóként szolgál.
- order_id: Az előzőleg az orders táblában létrehozott, új vásárlás azonosítója.
- pizza_id: Az adott vásárlásnak egy terméke. Egy order_idhoz több pizza_id is tartozhat, mert egy vásárláshoz több pizza is tartozhat.
- pizza_name: a megrendelt pizza aktuális időpontjában lévő neve. A vásárláskori név, szóval, ha megváltoztatjuk később a nevet, akkor ez biztosan itt marad és nem fogja lekövetni a változást. Fixen ide be lett írva és nem hivatkozással olvassuk ki.
- topping_list: a feltétek listája is fixen van, mégpedig json encoded stringben.
- size: a méretnek a neve, szintén fixen.
- price: az ár, szintén fixen.
- quantity: az a mennyiség, ami megmutatja, hogy hány darab pizzát szeretne a felhasználó rendelni
- subtotal: szintén fixen van behelyezve. Az adott termék ára, mennyiség szerinti ára.

4.5. Új termék bevitele

1. A Pizzas táblában felvesszük a kívánt adatokat, ennek eredményeként kapunk egy pizza azonosítót, amely az id lesz. A képet el kell helyezni az images/products-ban (VS CODE). Az image mezőbe csak a kép fájl nevét írjuk, mert ezt automatikusan hozzá illeszti a program.
2. A pizzas_toppings táblába beszúrunk annyi rekordot (beszúrás), ahány feltétet szeretnénk adni a pizzához. Beszúrásnál az id értéket üresen hagyjuk a pizza_id-hoz pedig beírjuk az újonnan kinyert id-t. Topping_id-ba beírjuk azt a toppings táblában lévő feltétnek az azonosítóját, amelyet hozzá szeretnénk adni. Annyiszor hozzuk létre, ahány feltétet szeretnénk a kívánt pizzára.
3. pizzas_sizes táblában beszúrunk annyi rekordot, ahány méretbe szeretnénk kínálni a pizzát. Szintén megadjuk az 1-es pontban kinyert pizza azonosítót. Második lépésként a size_id mezőbe annak a méretnek az azonosítóját, amit hozzá szeretnénk adni (sizes

táblában vannak a méretek) + beírjuk a price mezőbe a kívánt ár összeget (ezt is 2x csináljuk meg, mert 2db méret van).

5. Fejlesztői dokumentáció

5.1. A HTML és a CSS

Első lépésként felépítettem az index.php fájlt, amely tartalmazza a HTML kódot. A fájl első soraiban hívom be a function-öket és az adatbázis kapcsolatot is itt létesítem, azaz inicializálom az oldalt.

Ezt követően következett a HTML szerkezet, amelyben ügyelni kell arra, hogy a HTML nyelvezete „hu”, azaz magyar. Létrehoztam benne a CSS kapcsolatot, melyet a materialize.css oldaláról másoltam át. Létrehoztam egy szuper globális javascript változót, amiben azt a terméket tároljuk, melyet a kosárba szeretnénk helyezni. Ezt követte a „currency”, azaz az oldalon használt pénznem, végül pedig egy publikus útvonal, ezért is szuperglobális, mert több javascript is használja, amit a config.php-ben tárolunk. Ebben a fájlban vizsgáljuk, hogy a felhasználó vendégként van-e belépve, avagy sem. Amennyiben a felhasználó nincs belépve, akkor megadjuk neki a regisztráció és a belépés lehetőséget. Ha be van lépve, akkor mutatjuk neki a „profilom” és a „kilépés” linket, ezek mellett helyet kapott a kis kosár ikon is, azaz a shopping cart elérési útvonala is.

A kód további részében jelenik meg a reszponzivitás, azaz a mobilon is elérhető verzió, ahol ugyanazt a folyamatot vizsgáljuk, amit fentebb is említettem.

Ezt követően pedig a lábléc található, ahova szabadon meglehet adni a pizzéria elérhetőségeit, telefonszámát, e-mail címét, illetve az utca nevét is.

Az oldal legalján pedig a szükséges scriptek kaptak helyet, melyek a jquery, „materialize.js” és az „init.js”.

A CSS mappában található a „style.css” fájl, mely egy HTML oldalainak megjelenését befolyásoló egyszerű nyelv. Ennek segítségével határozzuk azt meg, hogy hogyan jelenjenek meg egyes HTML elemek. A CSS segítségével formázhatjuk a címsorokat, paragrafusokat, módosíthatunk színeket, méreteket, margókat. Ha egy stílust gyökér elemre definiáltunk, akkor öröklődnek, hierarchija szerint.

Szintén ebben a mappában kapnak helyet a „materialize.min.css” és a „materilazie.css” nevű fájlok is, melyeket a Materialize.js oldalról az általam választott témával töltöttem le, mint a csomag része.

A CSS kódban leginkább a Materialize.com oldalról letöltött templatekhez találhatóak utasítások, hogy azok az én igényeimnek megfelelően legyenek elrendezve és kevés olyan css elem van létrehozva a fájlban, amelyet magamtól találtam volna ki. A főcím méretét én adtam meg a fájlban, mint ahogy az ikonoknak is a méretét is én formáztam. A szövegezést is én állítottam sorkizárttá. Gyakorlatilag a „Materialize” használatával, ahhoz, hogy olyan témájú weboldalt kapjunk, melyet én használok, ezek a szükséges beállítások, melyek a Materialize.com weboldaltól szabadon letölthetők, mivel, ha ezt a csomagot szeretnénk használni, akkor a csomag részeként tölti le a stíluslapot a felület.

5.2. A képek feldolgozása

A képek, melyeket megjelenítek a weboldalon, azokat én töltöttem le a <https://stock.adobe.com/> oldalról, melyeket a letöltés után kicsinyítettem le. A weboldal hátterét az images mappában tároltam le, viszont az eladó termékeket az „images” mappán belül a „products” mappában kerültek eltárolása és onnan kerülnek betöltésre a weboldalra.

5.3. Fájl szerkezet, a weboldal részei

A „pages” mappában, ahogy a neve is utal rá, az weboldal által használt oldalakat tároltam el, melyek lentebb felsorolásra és bővebb kifejtésre kerülnek.

A „pages” mappán kívül helyett kapott a config.php nevet viselő fájl is. A config.php fájlban található a konstansok definiálása, csak akkor fusson le, ha még nem történt meg a definiálás.

Mik az előnyei a definiálásnak?

Az egész program futása során nem kaphatnak új értéket és így is kezeli őket a program. Ezeket az adatokat beégetjük, memóriát spórolunk meg vele az így definiált konstansok láthatók mindenhol, minden scopeból.

Az oldal relatív útvonalának az elérése a: '/pizzarendelo_bead'-on keresztül történik meg.

A config fájlban belül lentebb található néhány adatbázis adat: „host” név, a „user”, a jelszó, a tábla neve.

A config fájlban van definiálva a pénznemnek a definíciója is. Abban az esetben, ha újra definiálnánk egy define függvényt, akkor az fatális hibát eredményezne és leállna a program.

Az első fájl, amelyet létrehoztam az oldalon az „index.php” nevet kapta, ez gyakorlatilag a weboldal főoldala. Itt található egy rövid bemutatkozás, ahová szabadon választható szöveget tudunk beírni. Alatta található egy úgynevezett „waves-effect”-el ellátott menüsor, ahol kilistázzuk a rendszeresen rendelt pizzákat, úgynevezett „hot dealeket”, melyeket a template_parts.php-ban tárolunk és onnan hívjuk meg azokat. Az egész html kód tartalmát visszaadja, mely magát a terméklistát alkotja, ez mindig ismétlődik. Ennek az az előnye, hogy akárhány helyen módosítjuk a dobozokat, akkor mindenhol módosul, például ha egy képet váltunk. Lentebb található egy make_url funkció, amely azt hajtja végre, hogy az ékezetes karaktereket kicseréli ékezet nélkülire a termékek URLjében. Először kisbetűssé alakítjuk, majd az „str” megvizsgálja, hogy kapott-e olyan karaktert, amely az „str1”-ben szerepel, amennyiben tartalmaz olyan karakter, amit az „str1”-ben talál, azt kicseréli az „str2”-ben található karakter megfelelőjére.

A kezdőoldalon megjelölt „hotdealek” kezelése is ebben a fájlban kapott helyet. Az adatbázisban a „pizzas” táblában az oszlop címe az, hogy „hotdeal”, ha itt 1-es értéket adunk az adott mezőnek, akkor azt fel fogja venni a „hotdealek” közé, ami azt eredményezi, hogy meg fogja jeleníteni a kezdő képernyőn, mint top 4 pizza. Ha az értéket 0-ra állítjuk, akkor kikerül a menüsorból. A pizzák megjelenítése random sorrendben jelenik meg. Maga a lekérdezés úgy néz ki, hogy lekérdezzük a listát, majd egy ciklus segítségével végig megyünk a listán. A listboxban generált tartalmat hozzáfűzzük a result stringhez.

Az oldalon helyet kapott egy gomb is, melyre az van írva, hogy „Tekintsd át a kínálatunkat”. Ez a link a „product.php” oldalra navigál minket. Itt a product.php-ban ugyan az a folyamat, mint ahogyan azt fentebb leírtam, viszont itt más sql lekérdezést használunk, mivel itt minden rendelhető terméket lekérünk. A „product.php”ban beolvassuk a product_details termék összes adatát és azt meg is jelenítjük.

A terméklapon szereplő adatok lekérése az adatbázisból url alapján történik. A pizzas táblában van egy url, minden pizzának van egy saját url-je, amint egyezést talál a rendszer, akkor megjeleníti. Amint rákattintunk az általunk szimpatikus pizzára, akkor generálódik egy url és egy új oldalon betölti nekünk azt a pizzát a nevével, képpel és az adataival. Készítünk egy tömböt és annak a pizza nevű kulcsára rákerül a kiolvasott pizza alap adata, ez egy kétdimenziós tömböt használ.

A pizzák rendelkeznek feltétekkel, melyeket toppings néven tároltam az adatbázisban. A pizzákhoz tartozó feltéteket lekérjük az sql-ből és ezt behelyezzük a toppings nevű tömbbe. A tömbön belül létrehozunk egy toppings indexet és erre elhelyezzük az előzőleg kigyűjtött toppingsokat, feltéteket. A tömbnek létrehozunk egy prices nevű indexet, amire elhelyezzük azokat az árakat, amik tartoznak a különböző méretű pizzákhoz. Végül ezt a tömböt visszaadjuk.

Lekérjük a termék adatait „id” alapján, ez a lekérés két lépésben történik. Az első, amikor lekérjük az id alapján az URLjét, majd az URL alapján lekérjük a pizza minden adatát. Ezt követően az „amount” megformázza a mögé adott számértéket pénz formátumra és elhelyezi a pénznem szimbólumát.

Az oldalon szerepel egy funkció „only_guest” néven, amely egy olyan div, amely gyakorlatilag egy nézet, ami információt nyújt a felhasználónak, abban az esetben, ha olyan részre téved, amely csak a regisztrált vendégek számára érhető el. Ilyen például a „profile”, hiszen értelmetlen lenne egy vendég felhasználó számára a profil oldal, abban az esetben, ha még nem regisztrált saját magának profilt.

Ennek megfelelően az oldalon szerepel egy olyan funkció is, melyet „only_members”-nek neveztem el, amely szintén egy nézet, ami információt nyújt a belépett felhasználóknak abban az esetben, ha egy olyan oldalra téved, amely csak a regisztrálatlan felhasználók számára érhető el pl.: regisztráció, vagy a login. Mivel egy olyan személy nem fog tudni regisztrálni, aki már be van lépve, vagy nincs lehetőség a belépés után egy újbóli belépésre sem. Ez önmagában nem ellenőrzi, csak egy információt ad vissza, formázásban segít, azért kellett a function, mert sok helyen használjuk, akkor mindig a html-ben kellett volna ezt ismételtetni. Ha valamit módosítunk, akkor mindenhol elvégezzük azt az adott módosítást.

A Functions fájl tartalmazza a publikus elérési útvonalat, melyet visszaad a mappa név szerint. Ezen keresztül generáljuk a végleges urlt. Ennek a segítségével generálódik az oldal elérési útvonala.

Itt behívjuk a „connection” függvényt, amely az adatbázis kapcsolat felépítéséért felelős, az includes/config.php-ban megadott konstansok segítségével. Visszaadja az adatbázisból a size táblából az alapértelmezett méretet, ami akkor kerül érvényesítésre, amikor még nincsen méret választva. Az adatbázisban a size táblában a default nevű oszlop tartalmazza ezt az értéket.

Visszaadja nekünk az adott pizza alapértelmezett mérethez tartozó árát. Mivel az adatbázisban az van beállítva, hogy az alapértelmezett méret az a normál, ezért a normál pizza árát kapjuk meg. A `pizzas_sizes` táblában megkeresi a pizza azon ár értékét (`pizzas_sizes` tábla, `price` mező), ahol a `size_id` azonos a `size` tábla értékével, ahol a default oszlop 1. Ezt könnyedén át lehet állítani a „`sizes`” táblában, ha az 1-es értéket a „Nagy” méret mellé helyezzük el a default mezőben, akkor alapértelmezetten a nagy pizzák méretét mutatja először.

A kódban a méretek közül az adott méret idhoz tartozó adatokat adja vissza a `sizes` tábla rekordja. Egy adott pizzához társított összes méretet és árat visszaadja a „`getAppPrice`” függvény. A visszaadott értékek az ár, a méret neve alapértelmezetten.

A „`getOptionalPrice`” függvény pedig egy adott pizza méret azonosítójához tartozó árat ad vissza.

5.3.1. Registration.php

Ez a felület biztosítja a regisztrációs felületet az oldalnak. Első lépésben változóba helyezzük a postba érkező adatokat. A `trim` funkció levágja a whitespaceket, azaz ne legyen szóköz az elején se a végén. Pl.: valaki beírja a nevét kettő darab szóközzel az elején és a végén, akkor azt a program kód le fogja vágni.

A felületen helyet kapott egy a hibaüzenetek tárolásához inicializált tömb, ha valami üresen marad, vagy a jelszó nem megfelelő, vagy az e-mail nem megfelelő kontextusban lett beírva, akkor ez figyelmezteti a felhasználót. Egy-egy hibaüzenet teljesülésekor az `errors` tömbbe, asszociatív tömbként bekerül az adott hibaüzenet. Ez az jelenti, hogy egy kulcshoz pontosan egy érték tartozik. Az asszociatív tömb kulcsa lehet karakter összefűzés is. Az asszociatív tömb kulcsa megegyezik annak az inputnak a nevével, amihez tartozik, ezek össze vanna kötve, szóval, ha hibás egy input, akkor az alatt fog megjelenni a hibaüzenet is, hogy egyértelmű legyen, hogy melyik beviteli mezővel van a probléma.

A kód tartalmaz egy beépített függvényt is, ami a „`filter_validate()`”, ami az email formátumát, azaz a „`@`” meglétét vizsgálja. Ezt követően le ellenőrzi, hogy létezik-e a megadott e-mail címmel felhasználó az adatbázisban. A lekérdezés során az eredmény, amit hozhat az „0”, vagy pedig „1”. Abban az esetben, ha egy, akkor hiba van és nem engedi a regisztrációt végrehajtani, de amikor nullás értékkel tér vissza, akkor engedi a regisztrációt, mivel nincsen olyan e-mail cím az adatbázisban, mint amivel a felhasználó regisztrálni próbál. Az adatbázisban az e-mail mező egyedi indexel van ellátva, azaz „`UNIQUE KEY`”-el van ellátva.

Lentebb történik annak a ténynek a megállapítása, hogy a felhasználó be van lépve, akkor információt lát arra nézve, hogy ez az oldal csak vendégek számára érhető el, mármint maga a regisztrációs folyamat, tehát a felhasználó bejelentkezve nem fog tudni ismét regisztrálni, ahhoz ki kell jelentkeznie. Abban az esetben, ha a felhasználó nincsen bejelentkezve, akkor megjelenik a regisztrációs űrlap, ahol regisztrációs folyamatot tud kezdeményezni. Ezt a kódot követik az űrlap elemei. Az asszociatív tömb is itt van kiírva. Az errors tömbön hivatkozni tudunk az aktuális hibaüzenetre. A jelszó megerősítésnél nem használjuk az errors tömböt, mivel, ha nem egyezik a megerősítő jelszó az alap jelszóval, azt az alap jelszónál kommunikáljuk. A regisztráció gombbal pedig elküldésre kerül az űrlap.

5.3.2. Login.php

Azt a feltételt vizsgáljuk, hogy a küldésben szerepel-e a submitted index, el van-e küldve az űrlap, vagy nincs, ez azt jelenti, hogy rányomunk a bejelentkezésre, vagy sem. Akkor történik meg a vizsgálódás, amint rányomunk a gombra. Megpróbáljuk megtalálni azt a felhasználót, akinek az e-mail szerepel az adatbázisban és a megadott jelszó is, amit eltároltunk md5-ös hasheléssel. Mivel md5-ös formátumban tároljuk a regisztrált felhasználó jelszavát, így az összehasonlítás előtt az aktuálisan küldött jelszót szintén kódoljuk md5-el. A hasheket párosítja össze, hashben keres lényegében. Ezt követően megszámlálja a találatokat, amely vagy 1, vagy 0. Ha 0, akkor értelem szerűen nem talált ezekkel az adatokkal felhasználót, ebben az esetben hibaüzenet rögzítés történik a sessionben. Ezzel együtt egy oldalfrissítés is történik azért, hogy ne legyen ott a hibaüzenet. Amennyiben az értékünk 1 akkor, az azt jelenti, hogy volt találat és ráhelyezzük a session userre a talált felhasználónak az összes adatát és abból lesz a profil.

A login felületet csak azok a személyek láthatják, akik nincsenek belépve az oldalra, viszont, ha be van lépve a felhasználó, akkor megjelenítjük számára a főoldalt. A kódban a „??” dupla kérdőjel az valójában egy opcionális érték, ami arra hivatott, hogy elkerüljük az undefined variable hibaüzeneteket. Ha valami van az errors e-mail indexén, akkor kiírja nekünk pirossal, hogy mi a gond, ezért van a „??” dupla kérdőjel és üres string. Amennyiben nincs definiálva, akkor ne jelenjen meg hiba. Amikor megnyitjuk az oldalt, akkor azonnal ne írjon ki olyan változót, ami még nincs itt beírva. Ezt a műveletet használjuk a jelszó mezőnél is, csak magát a mezőt nem emailnek hívjuk, hanem passwordnek.

A kód alján a script, az Ajaxos kosár működésének a biztosítására szolgál.

5.3.3. Logout.php

A logout fájlban annyi történik, hogy megsemmisítjük a session szuperglobális indexén található adatot, azaz kilépünk a profilunkból. Ezt követően továbbítja a felhasználót a főoldalra.

5.3.4. Account.php

Az account oldalon két lehetőségünk van. Az egyik az az e-mail cím módosítása, a másik pedig a jelenleg használt jelszó megváltoztatása. A kódban azt vizsgáljuk, mind a két értéknél, hogy el lett-e küldve a módosító űrlap.

A jelszó esetében azt vizsgáljuk, hogy el lett-e küldve az űrlap és a jelszó megerősítés ellenőrzése is ki lett töltve. Továbbá vizsgáljuk azt is, hogy a jelszó hossza megfelelő-e, azaz megfelel-e a megadott kritériumoknak, tehát 6 karakternél nem kisebb és 30 karakternél nem nagyobb. A jelszó módosításnál figyelembe kell venni azt is, hogy a jelszó megerősítő mezőbe beírt érték egyezik-e az új jelszó mezőbe beírt értékkel. Amennyiben nem, akkor hibát dob azzal a felszólítással, hogy a „A két jelszó nem azonos!”. Amennyiben van hibánk, akkor a hiba rögzül az errors tömbbe. Ha nincsen hiba, akkor frissítjük a user rekordját az új jelszóval, tehát felülírjuk a régi jelszót azzal, amit a felhasználó megadott a felületen. Amennyiben nincs hiba a felület tájékoztat bennünket, hogy a jelszóváltoztatás sikeresen megtörtént.

Az e-mail cím módosításánál azt vizsgáljuk, hogy formailag megfelelő-e az, amit beírtunk és valóban egy e-mail címet írtunk be a mezőbe. Ezt a vizsgálatot úgy ellenőrzi a rendszer, hogy az új mezőbe, amit beírtunk szerepelnie kell a „@” jelnek. Amennyiben nem megfelelő az e-mail cím, akkor a hibaüzenet belekerül ugyan úgy az errors tömbbe és hibát fog nekünk kiírni, „Nem megfelelő formátum!” címmel. Amennyiben helyes a formátum, akkor még mindig tovább ellenőrizzük, hogy valaki más használja-e a használni kívánt e-mail címet, a jelenlegi felhasználók közül. Abban az esetben, ha más is használja az e-mail címet, akkor ki fogja írni nekünk a rendszer, hogy „Foglalt e-mail cím”, mivel az adatbázisban végrehajt egy sql lekérdezést. Ha talált rekordok száma nem 0, akkor az azt jelenti, hogy valaki már ezt az e-mail címet használatba vette.

Ha minden rendben van és senki nem használja az e-mail címet és a beírt cím formailag is megfelelő, akkor készítünk egy verification token, 2db md5-ös hash formájában, az egyik az e-mail cím címéből lett csinálva, a másik az aktuális időből plusz véletlen számból. Ezek nem összegként ként, hanem stringként fűződnek hozzá az inthez. Az int a jelenlegi idő és a string a véletlen szám, ami egymillió és kilencmillió között van. Ezekből lesz md5-ös hash és így mind a 2 string lesz. Ahogy ezzel elkészült a token, beillesztjük a felhasználó new email mezőjébe a verification token a verification token mezőben az sql-en belül és megjelenítjük azt az adatbázisban, hogy mikor lett megváltoztatva az e-mail cím.

Ahogy ez megtörtént, akkor rögzítjük a session[success]-ben a sikeresség tényét, tehát a rendszer közli velünk, hogy „Sikeres módosítás!” történt.

A header location-nél pedig csinálunk egy oldal frissítést. Az oldal frissítéskor a session successben lévő sikerességre utaló üzenetet kiíratjuk és kiíratás után azonnal megsemmisítjük azt. A kiíratott tájékoztatás addig lesz ott, amíg a felületet el nem hagyom, vagy meg nem frissítem. Az oldal alján szintén szerepel az ajaxos kosár működésének biztosítására szolgáló script.

5.3.5. Profile.php

A profil oldalon megjelenítjük főcímként azt, hogy ez a profil, illetve a felületen helyet kapott még kettő darab gomb is. Az egyik a „Megrendelésem” nevet viseli, a második pedig a „Belépési adataim” nevet kapta. A felületen ellenőrzésre kerül az, hogy be van-e lépve a felhasználó, ha nincs tájékoztatást kap, hogy csak belépve láthatja a profilját. Szóval, ha valakinek nincsen regisztrációja, akkor nem is látja a profilját. Csak abban az esetben nyer rálátást a felhasználó erre a felületre, miután regisztrálta magát az oldalra.

A két darab gomb Materialize.com oldalról letöltött ikonokat használ. Az egyik az az „add_shopping_cart” a második pedig a „verified_user”.

5.3.6. Product_list.php

A `product_list.php` tulajdonképpen az étterem menüje, itt kerülnek felsorolásra azok a pizzák, melyeket megvásárolhat az oldalról a felhasználó. A termékek kilistázásra kerülnek a felületen, de valójában a `template_parts.php`-n van definiálva a kód és a meghívásuk itt történik meg.

Az oldalon egy lapozót láthatunk, amit a Material.com oldalról szereztem be.

5.3.7. Product.php

Ezen a felületen beolvasásra kerül a `product_detail` termék összes adata és az kerül kiírásra. Az oldalon kilistázásra kerülnek a termékekhez kapcsolódó feltételek, melyik terméken mi található pl.: Mexicoi pizza: bab, sajt, csirke. Ezt követően kilistázásra kerülnek a termékek árai, illetve méretei is. Tekintettel arra, hogy egy pizzának két mérete lehet, ezért kettő ára is van. Alapesetben a default adattal rendelkező pizza jelenik meg, ami a normál méretet takarja és ezt lehet módosítani a nagy méretre. A default classal ellátott gomb kap egy kattintást, ennek köszönhetően van defaultként kiválasztva az alap pizza. A méretváltoztatás kódolását illetően így került megoldásra:

```
data-size="'. $price["size"]->size_id. "'  
data-id="'. $product["pizza"]->id. "'>'.  
$price["size_name"]->name. '</a> ';
```

Végezetül a kód alján található a kosár ikonja, aminek a href része el lett látva egy „#” jellel, erre azért volt szükség, hogy ne történjen oldal újra töltés.

5.3.8. Checkout.php

A `ceckout.php`ban lehet megadni a szállítási adatokat és itt történik a megrendelés véglegesítése is. Az egész oldal feldolgozása AJAX-on keresztül történik, tehát a `checkout.js`-ben van benne az a kód, ami a funkcionalitást végrehajtja. A `checkout.php` oldalon hozzuk létre a labeléket és az input mezőket is, illetve itt lehet kiválasztani a legördülő menüből a kiszállítás módját, ami alap helyzetben azt írja ki, hogy „Adja meg az átvétel módját!”. A kódban a szállítási adatokat elláttam copy-formal, erre azért volt szükség, mivel a szállítási adatokat le tudjuk másolni egy gombnyomásra abban az esetben, ha a szállítási adatok megegyeznek a számlázási adatokkal

is. Ezt azért tartottam hasznosnak, hogy ne kelljen kétszer ugyan azokat az adatokat begépelni a megrendelés véglegesítésekor.

A php kódban a „fulltotal” értéknél szerepel egy int, ami azt a célt szolgálja, hogy egész számmá konvertálja az értéket, ha nem biztos, hogy van a kosárba valami, akkor ne hibát dobjon, hanem írjon ki egy 0-át. Alatta található meg a szállítás díja felirat is, ami akkor fog megjelenni, ha kiválasztottunk egy szállítási módot, mivel a házhozszállításnak külön díja van, és ha azt választjuk az hozzá adódik a pizza árához, tehát a pizza ára plusz a szállítás ára lesz az az összeg, ami a végén megjelenik. Azonban ez akkor is megjelenik, ha nem választottunk ki semmit, mivel az értéke lehet nulla Ft. Abban az esetben, ha azt az opciót választjuk, hogy a vásárlást folytatjuk, akkor visszaléptet minket a pizzákhoz, hogy újabb pizzát tudjunk a kosarunkba tenni, viszont, ha a megrendelés véglegesítése gombot választjuk, akkor továbbít minket a megrendelés véglegesítése felületre.

5.3.9. Checkout.js

Ez még a checkout.php része, viszont itt történnek meg a javascriptes műveletek.

Az első sora a kódnak az az űrlap, ahol véglegesíti a vásárló a rendelést. Ezt követően a checkboxra való kattintást követően fut le a kód, ha másolni szeretnénk a számlázási adatokhoz a cím adatokat. Amennyiben be lett kattintva a boks, akkor másolja az adatokat. Ha nincsen bepipálva, akkor törlésre kerülnek az értékek a számlázási mezőkből A „copyShipping2Billing” funkció maga a másolás funkció. A számlázási név mezőbe belekerül a szállítási vezetőnév és keresztnév. Ebből a kettőből adódik össze, mivel lehet cég névre is igényelni a számlázási adatokat. A kód végig megy a mezőkön és beazonosítja a hozzá tartozó számlázási mezőt és a szállítási értékek átkerülnek a számlázásiba, azaz átmásolásra kerül. Itt rendeljük hozzá, hogy a focus label felülete az input fölé menjen.

Ezt követően meghívásra kerül a copyform, ami másolni fogja a szállítási adatokat a számlázási adatokhoz. A for ciklus végigmegy mindazon elemeken, amiknek az az ID-juk, hogy „copyform” és ezeket az ID-ju elemeket fogja másolni.

Az onblur, az az esemény, amikor a focusban volt, de mellé kattintunk, így veszíti el a focust. Ez mindegyik elemen végig megy és mindegyikre érvényes az onblur. Lentebb helyeztem el a gomb funkcióját, ami véglegesíti a vásárlást kattintásra. Itt csak belerakjuk az inputokat kiolvasás nélkül. A foreach segítségével végig megyünk az input listán és az array tömböt

feltöltjük ID-val és a hozzá tartozó értékekkel. Ezt követően elindítunk egy fetch hívást, ezzel küldjük az AJAX PHP-nak a kiolvasást, felhasználó által beírt adatokat küldjük meg. A küldés postal történik meg és végül JSON string-é alakítjuk. Azért kell a JSON stringgé alakítás, mert csak szöveg formában tudjuk küldeni. Változtatások nélkül változót nem tudunk küldeni, át kell alakítani szöveggé. Küldést követően megkapjuk a válasz adatokat az AJAX PHP-től, a küldött adatokat feldolgozta és ha hibát tapasztalt, azt vissza küldte nekünk. Az a lényeg, hogy ezen a ponton visszakaptuk a válasz adatokat az AJAX PHP-től és további feldolgozás előtt az összes hibaüzenetet töröljük.

Ezt követi a törlés. Törli az előzőleg esetlegesen generált hibaüzeneteket. Megvizsgáljuk, hogy az „if”-nél, sikeres volt-e a küldés, vagy sikertelen. Ha undefined, akkor sikertelen, mert hiba volt a küldött adatokban. Itt végig megy a hibaüzenetek listáján, ahol megvizsgálja a program, hogy van-e benne adat, vagy sem. Belehelyezzük az adott inputhoz tartozó, esetlegesen visszakapott hibaüzenetet és azt kapjuk vissza. Az „error spambe” helyezzük bele az üzenetet. Így jelennek meg egyes „labelek” alatt a hibaüzenetek.

Az „else” részében pedig a sikeres küldést helyezzük el. Minden rendben volt és megtörtént a rögzítés, nem csinálunk mást, minthogy az „order completedre” továbbítjuk a felhasználót, ahol tájékozik arról, hogy minden rendben volt a rendelés leadásával kapcsolatban.

Lentebb található az a kódrész, ahol kitudjuk választani a szállítási módot. ID egyenlő a „delivery-methoddal”. Ennek van egy úgynevezett „onchange” eseménye, amelyhez hozzá kötünk egy folyamatot. A „const that = this;” kód az arra értendő, hogy „select” mező rövidített hivatkozása kívül érhető el és a „fetchen” belül nem, ezért tesszük változóba, hogy a fetchen belül elérhető legyen. Ezt követően „postoljuk” a kiválasztott értéket, kér szállítást vagy nem?

A válaszatot visszaalakítjuk változóba. A szállítási árat megjelenítjük, a termékek árait „fulltotal displayben” jelenítjük meg, plusz a szállítást is és ez a kettő adódik össze a „full total price”-ra. Az ár és az ár összegzés el vannak látva „hideable” class-al. Ezeket végig megyünk és ha valami ki lett választva szállítási módnak, akkor láthatóvá tesszük. Láthatóvá tesszük az árakat, az ár összegzést csak akkor, ha lett kiválasztva szállítási mód. Ezt a megjelenítést az árra kell érteni, illetve a tovább gombra. Nem jelenik meg a szállítás ára, a szállítási árral növelt összár és a véglegesítés gomb sem. Ha semmi nem lett kiválasztva, akkor az imént felsoroltak eltűnnek és nincsen véglegesítési gomb sem. Tehát, ha nem választunk ki szállítási módot nem tudunk tovább haladni a folyamatban.

5.3.10. init.js

Az init.js nevű fájlban tároljuk a „sidenav”-ot, ami a mobil menü összeállításához szükséges a materialize egyik komponense. Ettől fogja megkapni a mobilnézetet az oldal. A referencia oldal: materialize.com/css/grid/responsivelayouts elérési úton található. Itt találhatóak a listaelemek kép diveinek a háttérképei is. Először összegyűjtjük a „diveket”, utána a „data background” attribútumban lévő értékeket átalakítjuk háttérképpé, maga a megjelenés a „template_pars.php”ban található. Létrehoztam egy „divet”, amit átalakítottam háttérképpé és ennek adtam egy „covert”. A cikluson belül történik a „data background” átalakítás háttérképpé, azért van erre szükség, mert eltérő méretű képeink vannak és egyszerűbb, ha itt adok meg egy fix méretű „divet”, mintsem egyenként a képeket méretre vágom. Alternatívaként használhattam volna a „css object-fit” tulajdonságot is. A kódban szerepelnek még olyan sorok is, melyeket a material frameworktől kaptam, ahol a select egyedi megjelenítését végzi el.

Az init fájlban történik meg a kosár kiolvasása. Maga a kosár funkciók is, gondolok itt arra, hogy a kosár tartalmának az elérése, kattintásra elő hívódik vagy eltűnik. A jobb oldalt felül található kis kosarat megjeleníti, vagy eltünteti, az attól függ, hogy milyen állapotban van a kosár. Itt találhatóak a méretet jelző gombok is. A pizza méreteinek meghatározó gombok listáját begyűjti a „calcbtns”-be, aztán végigmegy rajtuk egy ciklussal. Mindegyik gombhoz ad egy „click” eseményfigyelőt, amely aktualizálja a pizza mérethez kapcsolódó árat. Szóval a nagy pizza X ezer forint, míg a kis pizza Y ezer forint. Ez a változás „clickelésre” változik. Ez alatt történik az ármódosítás, mivel a fentebbi csak inicializálja az árat, azaz csak kiolvassa az értékeket.

A terméklapon belül azt csinálja, hogy a „default classsal” rendelkező méret gombot alapértelmezés szerint „clickeljük”, tehát, amikor megnyitjuk ezt az oldalt alapértelmezettként ki van választva a normál méretű pizza. Alább található az ár módosításáért felelős rész, amikor a gombok árait aktualizálja a kód.

Az onload az AJAX kéréseknek egy úgynevezett „call back” ága, ami akkor teljesül, amikor komplett lett a lekérés és a válasz is visszaérkezett a PHP-től, itt egy konkrét lekérés van a termék mérethez tartozó árat illetően, konkrét információ van. A „result” nevű változóban tároljuk a termék méretéhez tartozó információkat. A termékid és méretid. A termékid a terméket határozza meg, míg a méretid, pedig azt, hogy egy adott terméknek hány féle változata van, illetve ezeknek a változatoknak milyen áruk van. Külön egy sonkás pizzát nem tudunk

adni, mert mindenképp ki kell választani azt, hogy mekkora legyen a pizza, mivel az határozza meg magát a terméket, amit megvásárol a vevő.

Lentebb végig megy egy ciklussal a gombokon, mind a két darab gomb először zöld és csak a lap betöltése után vált át a „nagy” méretű gomb szürkére, mivel a kicsi a „default”. Megkeresi, hogy melyik gomb tartozik a kiválasztott mérethez, eltávolítjuk róla a zöld class-t és ezáltal válik szürkévé, tehát kiválasztott. Ezzel szimbolizáljuk azt, hogy nem kattintható. Ennek a segítségével van az is meghatározva, hogy egyszerre két különböző méretű terméket ne lehessen bekattintani egyszerre. Magát a kiválasztási metódust hajtja végre. Nem lehet egyszerre kettőt belerakni egy eseménybe. Ha valaki kicsit akar először a kicsit teszi a kosárba, aztán visszamegy és beleteszi a nagyot is. Méret választás mindig van, bekerül a kiválasztott terméknek a méretazonosítója, az azonosító azonosítja be azt, hogy melyik termékről van szó és a mennyisége is, hogy hány darabot szeretne a felhasználó az adott termékből. Erre akkor van szükség, amikor a kosárban kattint a felhasználó, akkor az ebben tárolt infókat helyezi el a kosárba. Lentebb az adatok kiolvasása és küldése történik, a „select size” visszaadja a terméknek a kiválasztott méret változathoz tartozó árát és utána történik meg a küldés.

A toCart függvény a kosárba helyezést végzi el. Inicializálja a küldés adatait, termékid és méretid. Az append hozzá teszi a formdata-hoz az id kulcson az objektum azonosítóját. Formdata objektumban létrehozza az id kulcsot a termék id-val. Definiálva lett egy úgynevezett „quantity” változó, alapértelmezett mennyiséget jelent. Ha a „quantity” null, akkor nem a terméklapon vagyunk, hanem a lista oldalon, ekkor a mennyiség a paraméterben átadott objektumban definiált mennyiség lesz. Ha a terméklapon vagyunk, akkor pedig a „quantity” mennyiségbeviteli mező értéke lesz. Definiálásra kerül a mennyiség. Ezt a mennyiséget hozzákapcsolja a formdata objektum „quantity” attribútumára, ezt a hozzáfűzést azért kell megtenni, mert küldést szeretnénk végre hajtani. Ezt követően csatornát nyit az AJAX PHP felé a küldés céljából a program kód és végül megtörténik a küldés. A formdata küldéskor tartalmazza a termékid-t a méretid-t és a mennyiséget is.

Az AJAX ezt követően kiolvassa a kosár tartalmát és megvalósul az AJAX hívás.

Bizonyos funkciók, változók magyarázata a kódban, hogy mi mit hajt végre:

- function readCart(obj) – Kiolvassa az ajax kosár tartalmát.
- const xhr = new XMLHttpRequest – Xmlhttp request objektumnak a példányosítása, gyakorlatilag az ajax kérés inicializálása.

- `xhr.open('POST', public_path + 'ajax.php? operation=read_cart', true)` – Itt történik meg az ajax hívás megvalósítása.
- `xhr.send();` - Itt kerül elküldésre az ajax kérés, lentebb az onloaddal pedig a betöltése fog megvalósulni.
- `xhr.onload = function()` – Megtörtént a kiolvasás és a visszatérő értékek itt kerülnek kiolvasásra. Az összes adat ebben a response textben (válasz szöveg) van benne.

A „resultstring”-ben összegyűjtjük a HTML kód részletet egyesével. Minden termék képe, neve, darabszáma.

Lentebb megkapjuk a vásárlás komplett összegét, tehát a „subtotalok” (rész ár) összegét, mert az esetleges szállítási költséget nem tartalmazza. Abban az esetben, ha a „fulltotal” (végösszeg) egyenlő nullával, az azt jelenti, hogy a kosár üres.

Ezt követően a kódban kiolvassuk az „ajax-cart” idval ellátott divet, és a resultstringet belehelyezzük ebbe az elembe, aminek az lesz az eredménye, hogy a kosárba belekerülnek a termékek.

A kód alján pedig a tördelés látható és az ár formázása.

5.3.11. shopping-cart.php

A shopping-cart.php valójában az úgynevezett nagyobb kosár, ahová úgy tud eljutni a felhasználó, ha kiválasztotta a számára megfelelő terméket, akkor jobb felül ez bele fog kerülni a kosárba, amit jobb felül a bevásárló kosár ikon szimbolizál, amint rákattint a felhasználó lenyílik az ablak és megtalálja benne a megrendelni kívánt terméket. Ezt követően a kosár alján található egy úgynevezett „Pénztár” gomb, amely továbbítja a felhasználót a shopping-cart.php felületére. Az oldalon kiírás történik, ahol tájékoztatjuk a felhasználót, hogy ez a kosár, alatta pedig tájékoztató szöveg a gombok használatával kapcsolatában. Az oldal közepére pedig behívodik a shopping-cart.js fájl.

5.3.12. shopping-cart.js

A shopping-cart.js-ben találhatóak azok a funkciók, melyek magát a kosár funkcionális működését látják el. Ez a kosár az oldal törzsében van beolvasva, ennek az első sora, hogy

kiolvassuk a „cart-page-content”-et és ez fogja tartalmazni az úgynevezett nagy méretű kosarat. AJAX kérésen keresztül kiolvassa a PHP Sessionből, hogy mit helyeztünk kosárba és a behelyezett adatokat fogja kiolvasni. A kódsorban megtalálható az úgynevezett „await fetch”, ami azért került felhasználásra, mert ez a futás nem futás időben történik, valamennyivel több idő kell a kiolvasásra. Végül vissza tér a kód a kiolvasott adatokkal, amit a „GetCartData”-ból olvas ki.

A kódban helyet kapott egy függvény, melynek az a neve, hogy „fillCardContent”, ami annyit tesz, hogy feltölti, aktualizálja az előzőleg deklarált „cart-page-content” divet. Kiolvassa a „get-cart-data” által visszaadott adatokat és utána a shopping cart containernek, azaz a nagy kosárnak az innerHTML-jébe belehelyezi. Lentebb a diven belüli kód az egy termékre fog vonatkozni és a map pedig ciklusként fogható fel és ezen keresztül kerül ismétlésre. Itt lesz benne a termék képe, neve, mérete, darabszáma, ami itt változtatható, az a termék egység ára és a darab ára, azaz hányszor lesz az egységár. Ezt követően hozzá fűzünk egy teljes totál árat, ami a subtotalok árából jön létre és ezek képezik a termékek összárát. Alattuk el van helyezve két darab gomb, az egyikkel vissza tudunk lépni a vásárlás folytatásához, a másik pedig a „checkouthoz” visz, ahol véglegesíthető a vásárlás.

Az oldalon mivel van lehetőség a törlésre, ezért helyet kaptak az erre a célra szolgáló gombok, melyek funkciója az, hogy a törlést valósítják meg, kattintásra pedig, a „remove_from_cartot” fogja meghívni.

Az oldalon továbbá van lehetőség a megrendelni kívánt termék darabszámának a módosítására is. A kívánt darabszámot manuálisan kell a megrendelőnek begépelnie, amely egy Ajax kérésen keresztül történik. A kiolvasáshoz fetch metódust használ és annak a segítségével olvassa ki az összes jelenlévő mennyiségi mezőt. A mennyiség beállítása akkor lesz aktív, ha felemeljük a billentyűt. A kosár tartalma valós időben fog frissülni és ahányszor módosítunk a mennyiségen, annyiszor frissül az adat. Meghívjuk a „fillCardContent()” metódust, amikor az oldal frissül, akkor a kosár tartalma legyen belehelyezve. Továbbá a kód tartalmaz a W3School dokumentációt is, amely segítséget nyújtott az elkészítésre.

5.3.13. Orders.php

Az orders.php fájlban találhatjuk a megrendeléseket. Ez az oldal csak a regisztrált felhasználók számára érhető el, abban az esetben, ha valaki regisztráció nélkül rendelt valamit egy adott e-

mail címről és a rendelést követően beregisztrál magának egy profilt a korábban használt e-mail címére, akkor visszamenőleg is van lehetősége arra, hogy rátekintést nyerjen a korábban, még vendégként leadott rendeléseire is. A kód tetején biztosítva van az, hogy ne lehessen megtekinteni az oldalt akárki számára, amelyre csak akkor van lehetőség, ha a felhasználó bejelentkezett és hitelesítette magát. A feltétel másik része akkor jelenik meg, ha be van lépve a felhasználó és listázásra kerülnek az általa leadott rendelések. Megjelenítésre kerül az is, hogy a korábban leadott rendeléseknél milyen termékek lettek leadva. Felsorolásra kerülnek az előző vásárlások, illetve azoknak a részletei, áttekintő adatok, a megrendelés leadásának a dátuma és a szállítási mód. Végül megjelenik a fizetendő végösszeg. Kilistázza a felhasználó számára a megvásárolt pizzákat, sőt még azt is a felhasználó tudtára adja, hogy milyen feltételeket tartalmazott a pizzája.

5.3.14. Order-completed.php

Amikor sikeresen leadtunk egy megrendelést, akkor fog minket erre az oldalra továbbítani a rendszer, ahol kiírja nekünk a megrendelésünk sikerességére utaló üzenetet, azaz „Sikeres megrendelés!” és „A megrendelésed feldolgozása megkezdődött, munkatársunk hamarosan megkeres!”. Innen vissza tudunk térni az index oldalra, azaz a főoldalra.

5.3.15. Ajax.php

Elsőként behívódnak a „functionok” és adatbázis kapcsolatot létesítünk. Az első változó a kódban a kiszállítás áraira vonatkozik, minden egyes kiszállítási módhoz rendelve van egy ár és egy értelmezhető név is. Ezt követően definiálásra kerülnek a hibaszintek és eltüntetésre kerülnek a figyelmeztetés szintű hibák. Esetekre van bontva, hogy mikor fusson le az ajax, mivel az ajax php operation paraméterétől függ, hogy mi fog lefutni. Méretet tud váltani az ember (pizzaméret -> selec size). Küldésre kerül egy termékid és egy sizeid is, ezekből az adatokból kalkulálja ki a program az adott termékhez tartozó árat, ami a „getoptional price”-al valósul meg. Ez után azt a kódot láthatjuk, hogy mi az a válasz, amit visszaad méretváltáskor, az egy ár, egy termékid és a méretid kerül küldésre tehát ezeket adja vissza válaszként. Ezek az értékek behelyeződnek egy globális változóba, amikor a kosárba kerülnek. A session cart indexére ráhelyeződik egy újabb index, amely a termék azonosító és a méret azonosítóból áll.

Sessionben helyeződnek el a termékek és ezeknek a kulcsa a termékidből és a méretidből tevődnek össze. Az alábbi információk vannak egy tömbbe rendezve: termékid, méretid és mennyiség és ezek az információk mennek bele a kosarat szimbolizáló session változóba. Kiírásra kerül a session cart értéke. A „read_cart” minden egyes olyan alkalommal lefut, amikor ki kell olvasni a kosár tartalmát, amit visszaad az pedig egy tömb lesz a termékekkel. Inicializálásra kerül egy üres tömb, ahova elhelyeződik a kosárban tárolt, megjeleníteni kívánt tartalom. A session cartban elhelyezett termékek feldolgozásra kerülnek egy „foreach” ciklus segítségével. Az ár alapján ki olvasásra kerül a termék konkrét adata, amiket le is tárol a kód. A termék ára mellett a mérete is ki olvasásra kerül. A lentebb látható adatokkal kerül feltöltésre a tömb. Az aktuális mennyiség * termékár mindig hozzáadódik a subtotal értékhez, amint vége a ciklusnak a termék végleges ára belekerül a „fulltotal” változóba. Szövegformátummá konvertálódik a kosár tartalma, erre azért van szükség, hogy a felhasználó felé szövegesíteni lehessen, hogy frontend oldalon megtudjon jelenni. Megjelenik az is, hogy milyen termékkel van feltöltve a kosár, mennyi a fizetendő összeg és a pénznemet is az összeg mögé helyezi.

A „remove_from_cart” a termék törlésére vonatkozik. A kosár termékazonosítójából és méretazonosítójából összeállt indexét fogja törölni. Egy termék két féle mérettel is szerepelhet egy kosárban és törölhetőek is külön-külön.

A „change_quantity” a mennyiség változtatására szolgál. A session cartnak a termékazonosítójából és a méretazonosítójával definiált indexét módosítja és abból lesz a quantity az új érték. Gyakorlatilag egy felülírás történik. Ezt követően a frontendnek egy válasz, hogy sikerült a módosítás, ami csak a hálózati monitorozásban látszódik, nem íródik ki a felhasználó elé.

A „verify-checkout” akkor kerül használatba, amikor a vásárlást véglegesíti a felhasználó és rákattint a vásárlás véglegesítése gombra, ide érkezik az a sok űrlapadat, számlázási cím, számlázási adat stb. Előfeldolgozásra kerülnek az adatok, inicializálódik az a tömb, amiben tárolni fogja a később felmerülő hibákat a program. Lentebb helyet kapott egy segédfüggvény is, amely azt a célt szolgálja, hogy a beírt adatok hosszúságát tudja ellenőrizni. A „converted” nevű változó lesz az, ahova bekerülnek a felhasználó által írt adatok. Rengeteg kikötés szerepel az űrlap kitöltésekor, ilyen kikötés például, hogy a vezetéknév hány karakter lehet tól-ig meghatározva, hány karakter lehet az irányítószám, illetve az is, hogy a szállítási mód kiválasztása kötelező.

Miután a felhasználó kitöltötte az űrlapot megszámolja a program, hogy hány hiba keletkezett az errors tömbben, ha nagyobb, mint nulla, akkor hiba van. Json encodeban visszaadja a frontendnek és megjeleníti a felhasználó számára a hibásan kitöltött inputmezőket. Amennyiben nulla lett a hibák száma, akkor adatbázisban eltárolja az űrlapon megadott adatokat. A fulltotal nevű változót arra használja a program, hogy kiszámolja a felhasználónak a full total árat. Belehelyeződik a kosár összes eleme egy tömbbe és közben kiszámolásra kerül a végösszeg. Ezt követően elhelyezésre kerül az orders táblában a vásárlások általános adatai. Végül rögzítésre kerül a sikeres vásárlás ténye és ezt meg is kapja a felhasználó a frontend oldalon is a Json segítségével.

A „calculate-shipping” fogja kiszámolni azt, hogy mennyi lesz az adott szállítási ár, illetve a teljes árat is megadja. A checkout.jsből akkor jön adat, amikor megtörténik a kiválasztása az átvételi módnak. Akkor történik egy ajax kérés, ami ide küldi az adatokat. Csak azt a nevet küldi el, ami ki lett választva az optionból a szállítási mód megadásakor. Ez a küldés post-al történik. Az első kulcs a price, ami visszaadja az adott szállítási módnak az árát. A full_price nevű kulcs pedig összeadja a termékek árát, illetve a szállítási díjat. A teljes fizetendő végösszeget fogja visszaadni. A sessionben eltárolásra kerülnek az információk és az összegszerűségek, mennyi volt a szállítási díj, annak a módnak a neve és a díja.

6. Az oldal elérése

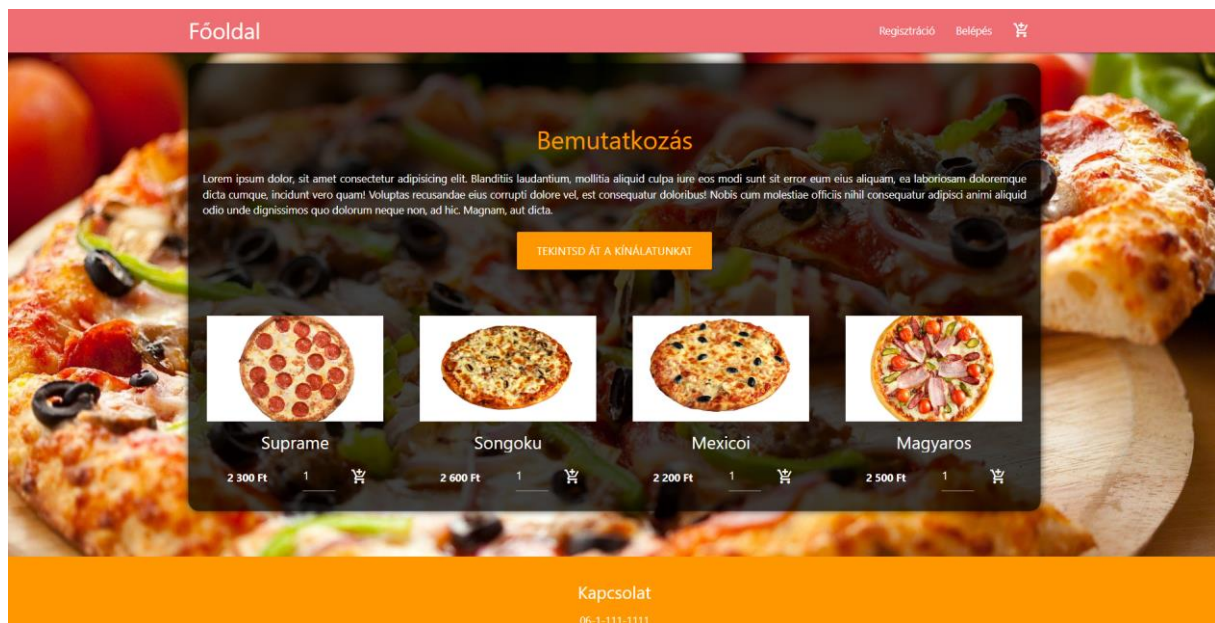
A weboldal lokális hálózaton érhető el, ehhez szükség lesz egy szerverre, ami az én esetemben a XAMPP nevű virtuális szerver volt. Miután telepítjük a XAMPP-ot ki kell keressük a számítógépünkön a program mappáját. Ez általában a („C:”) meghajtón van. A mappán belül keressük ki a „htdocs” mappát és helyezzük bele a Pizzarendelő mappát. Ezt követően nyissuk meg a XAMPP-ot és indítsuk el rajta az „Apache” és a „MySQL” modulokat a „Start” gombok segítségével. A MySQL modul mellett van egy olyan gomb, hogy „Admin”. Erre kattintsunk rá. Itt válasszuk ki bal felül az „Új” lehetőséget. Ezt követően azt fogjuk látni, hogy „Adatbázis létrehozása”, alatt nevezzük el az adatbázist „pizza” néven. Fontos, hogy ez legyen az adatbázis neve, mivel a kódon belül a „pizza” nevű adatbázist fogja keresni a programsor. Felül a menüsorban válasszuk ki az importálást. Válasszuk ki az „SQL” fájlt a számítógépről, ami a Pizzarendelő mappán belül van és feltöltésre kerül az adatbázis.

Nyissuk meg a Google Chrome böngészőt és gépeljük be az oldal „http:” részébe azt, hogy: http://localhost/pizzarendelo_bead/ vagy pedig: http://127.0.0.1/pizzarendelo_bead/ . Mind a két elérési útvonal működik.

7. Felhasználói dokumentáció:

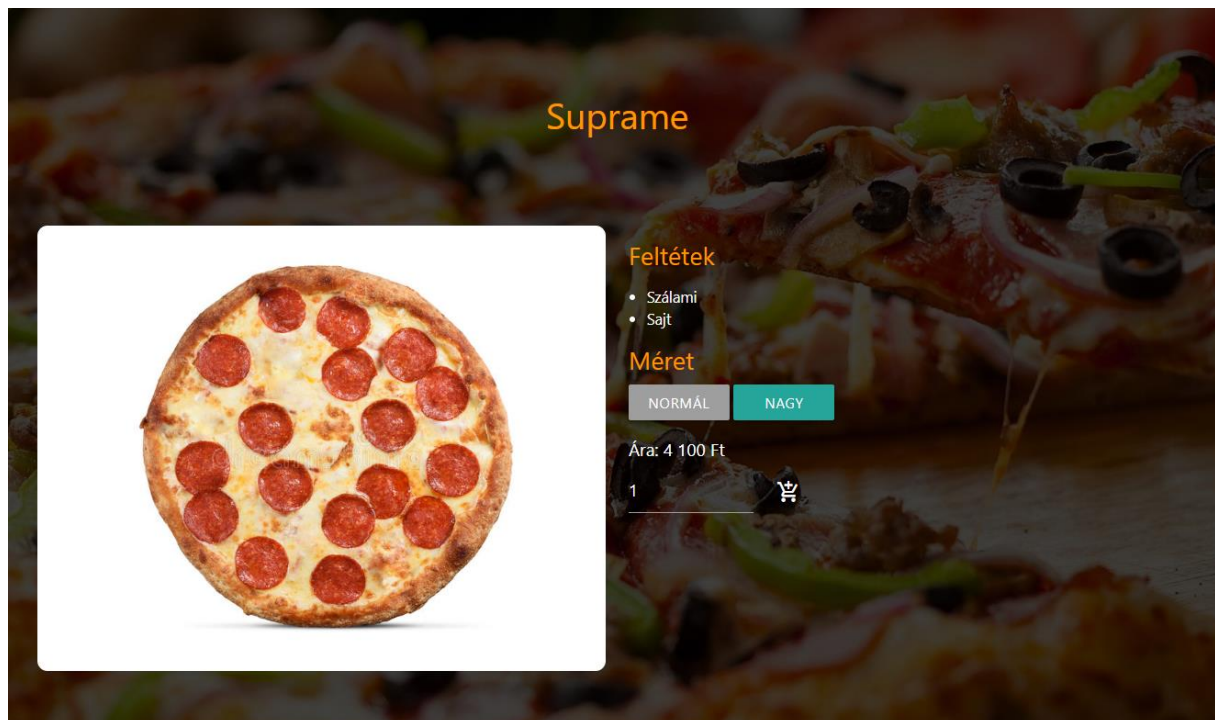
7.1. Rendelés

Amint megnyitjuk az oldalt, akkor a bemutatkozás szöveggel és a legjobb ajánlatokkal a főoldal fog minket, felhasználókat fogadni.



A felhasználó a vásárlását megteheti vendégként, illetve regisztrálhatja magát az oldalra és akkor létrejön a számára egy profil, aminek a segítségével a rendszer listázza az eddigi rendeléseket.

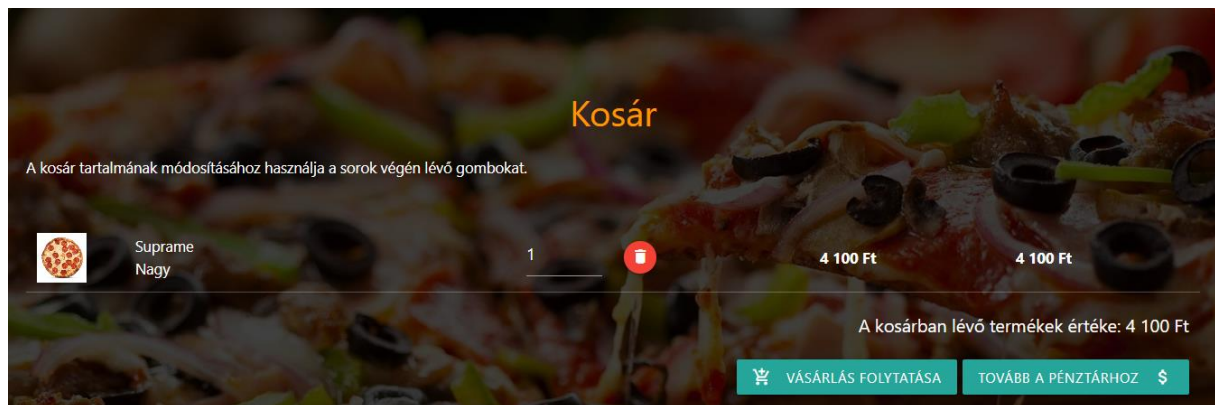
Ha a felhasználó úgy dönt, hogy nem fog regisztrálni és megtetszik neki egy pizza a „hotdeals”-ek közül, akkor rákattint az általa kiválasztott pizzának a nevére, ahol egy részletes tájékoztatót kap arról, hogy az adott pizzán milyen feltétek szerepelnek. Ha a felhasználónak szimpatikus a pizza, akkor kiválasztja az általa preferált méretet és mennyiséget, ezt követően a kosár ikonra kattint.



A kattintást követően a jobb felső sarokban meg fog jelenni egy legördülő ablak, ami magát a kosár tartalmát fogja megmutatni a felhasználónak. A legördülő ablakban helyet kapott egy gomb is, ami a pénztár elnevezést kapta, ha rákattint erre a felhasználó, akkor itt még tudja módosítani a megrendelését.



Lehetőség van a megrendelni kívánt termék mennyiségének a módosítására, illetve a termék kosárból való eltávolításra is.



Amennyiben a felhasználónak eszébe jutott, hogy még egy más fajta terméket is vásárolna, akkor a „vásárlás folytatása” gombra kattintva vissza fog jutni a kezdőoldalra. Azonban, ha a felhasználó nem kíván több terméket rendelni, akkor a „tovább a pénztárhoz” gombot választva el fog jutni a megrendelés véglegesítése felületre.

A megrendelés véglegesítése!

Az alábbi úrlapon adja meg a szállítási adatait, vagy válassza ki az átvétel módját, azután kattintson a megrendelés véglegesítése gombra!

Szállítási adatok

Vezetéknév _____ Keresztnév _____

Irányítószám _____ Város _____

Utca _____ Házzszám _____

☐ A számlázási adatok megegyeznek a számlázási adatokkal

Számlázási adatok

Név / Cégnév _____ Adószám (cég esetén) _____

Irányítószám _____ Város _____

Utca Hátszám

Account adatok

E-mail

Az átvétel módja

Adja meg az átvétel módját

A kosárban lévő termékek értéke: 4 100 Ft

VÁSÁRLÁS FOLYTATÁSA

Kapcsolat

06-1-111-1111
pizzarendelo@gmail.com
XY utca 1

Made by Materialize

Itt értelemszerűen ki kell tölteni a felhasználó adataival a megfelelő mezőket. Az oldal alján lehetőség van arra is, hogy a felhasználó kiválassza az átvétel módját. Amennyiben személyes átvételt szeretné választani a felhasználó, akkor azt választja ki, de ha inkább a házhozszállítást preferálja, akkor meg azt.

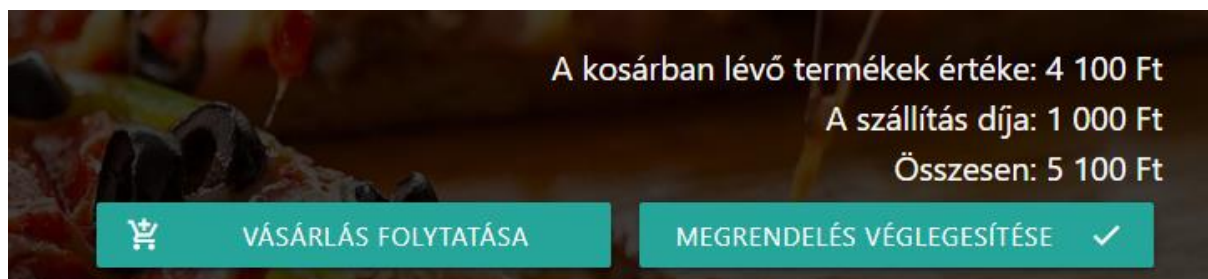
Az átvétel módja

Adja meg az átvétel módját

Személyes átvétel

Kiszállítás

Abban az esetben, ha a felhasználó a kiszállítást választja, akkor a rendszer automatikusan fel fogja számolni a végösszeg mellé a szállítási költséget is.



Ha minden rendben van és megfelel a rendelni kívánt termék, akkor a rendelés véglegesítése gombbal le fogja tudni adni a megrendelését a felhasználó.

A sikeres megrendelésről egy tájékoztató oldal fog megjelenni.



7.2. Regisztráció

Ha a felhasználó szeretne magának egy profilt, akkor a „Regisztráció” gombra kell kattintania. Ezen a felületen meg kell adnia, hogy mi az ő neve, e-mail-címe és, hogy milyen jelszóval szeretne belépni a későbbiekben az oldalra.

Ezt követően rá kell nyomin a regisztráció gombra, ha ez megtörtént és minden adat rendben van, akkor a „Sikeres regisztráció!” felirattal találkozik a felhasználó.



7.3. Belépés

A felhasználónak a belépéshez be kell írnia a regisztrációkor megadott e-mail címet, illetve a jelszavát is.

A belépést követően a „Profilom” felirat fogja fogadni a felhasználót, ahol megnézheti a korábbi rendeléseit, illetve módosítani tudja a belépésre szolgáló adatait.

A screenshot of a web application showing the user's order history. The background is a blurred image of a pizza. The title "Megrendeléseim" is in large orange text. Below it, there is a table with columns for date, delivery status, item name, quantity, and price. The table shows two orders: one for "Magyaros" (2 500 Ft) and one for "Mexicói" (2 200 Ft). At the bottom, there is a button labeled "BELÉPÉSI ADATAIM" with a checkmark icon.

2022.10.21	kiszállítás			4 700 Ft
Magyaros	Normál	2 500 Ft	1db	2 500 Ft
Sajt, Kolbász, Bacon, Paradicsom, Cse...				
Mexicói	Normál	2 200 Ft	1db	2 200 Ft
Bab, Sajt, Csirke				

BELÉPÉSI ADATAIM 

Belépési adataim

E-mail megváltoztatás

Jelenlegi e-mail cím
farkasbence@gmail.com

Új e-mail cím

MENTÉS! ✓

Jelszó változtatás

Jelszó

Jelszó ismét

MENTÉS! ✓

MEGRENDELÉSEIM

A „Kilépés” gombbal fog tudni kilépni a profilból a felhasználó.

7.4. Admin bejelentkezés

Ezen a felületen, ugyan azt a metódust kell végrehajtani, mint a „Belépés”-nél, annyi különbséggel, hogy itt nincsen mindenkinek jogosultsága a belépéshez. Ezt az oldalt http://localhost/pizzarendelo_bead/system_admin útvonalon érheti el az adminisztrátor.

Belépés

E-mail cím

Jelszó

BELÉPÉS ✓

Abban az esetben, ha admin joggal rendelkezünk és helyesen begéptük a bejelentkezéshez szükséges adatokat, akkor az „Admin kezdőoldal” felírat fog minket köszönteni.

Admin kezdőoldal

USEREK

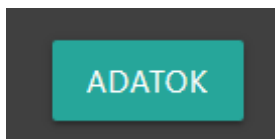
MEGRENDELÉSEK

7.5. Userek

Az oldalon lehetőség van a felhasználók áttekintésére. Ehhez a funkcióhoz kattintsunk a „userek” gombra.

Userek Admin				
Id	Név	Admin	E-mail	Művelet

A táblázatban fel vannak sorolva azok a felhasználók, akik regisztrálták magukat a weboldalra. Az „Adatok” gombbal tudunk rámenni az adott felhasználó adatainak az áttekintésére.



Az adatok szerkesztésénél lehetőségünk van a felhasználó adatainak a szerkesztésére, tudjuk a felhasználó fiókját törölni a „Törlésre jelöl” checkbox segítségével.



Illetve lehetőségünk van az adminisztrációs jogok kiosztására is.

Adminisztrátor
Igen
Nem

Ha a nemet át állítjuk igenre, akkor az adott felhasználónak lehetősége lesz belépni az admin oldalra, ugyan úgy, mintha a megrendelő oldalra lépne be, tehát a belépési adatok ugyan azok lesznek.

7.6. Megrendelések

A megrendelések felületen kilistázza nekünk a rendszer a korábban leadott eddigi összes megrendelést.

Megrendelések		
2022.10.21	kiszállítás	4 700 Ft
2022.10.21	kiszállítás	4 100 Ft
2022.10.21	kiszállítás	2 600 Ft
2022.10.17	személyes átvétel	2 300 Ft

Abban az esetben, ha egy megrendelés szürke, az azt jelenti, hogy még feldolgozás alatt áll, de ha rákattintunk a mezőre, akkor tudjuk a státuszát elkészültté tenni.

Megrendelés részletei: #22									
2022.10.21	kiszállítás				4 700 Ft				
Magyaros	Normál	2 500 Ft	1db	2 500 Ft					
Sajt, Kolbász, Bacon, Paradicsom, Cse...									
Mexicói	Normál	2 200 Ft	1db	2 200 Ft					
Bab, Sajt, Csirke									
Szállítási adatok									
Nagy Aladár	1010 Budapest				Teszt 1				
Számlázási adatok									
Nagy Aladár	1010 Budapest				Teszt 1				
<input type="checkbox"/>	Elkészült								
MENTÉS! ✓									
KEZDŐOLDAL 🏠									

Kattintsunk rá az „Elkészült” checkboxra és ezt követően a mentés gombra.

☒ Elkészült

MENTÉS! ✓

Miután ezt megtettük, kattintsunk a kezdő oldalra és válasszuk ki ismét a „Megrendelések” opciót.

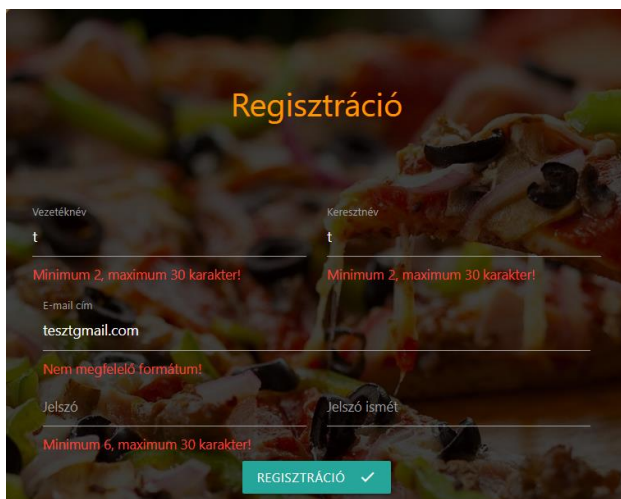
Megrendelések		
2022.10.21	kiszállítás	4 700 Ft
2022.10.21	kiszállítás	4 100 Ft
2022.10.21	kiszállítás	2 600 Ft
2022.10.17	személyes átvétel	2 300 Ft

Itt látható lesz, hogy a megrendelés státusza sikeresen átváltozott.

8. Tesztelési dokumentáció

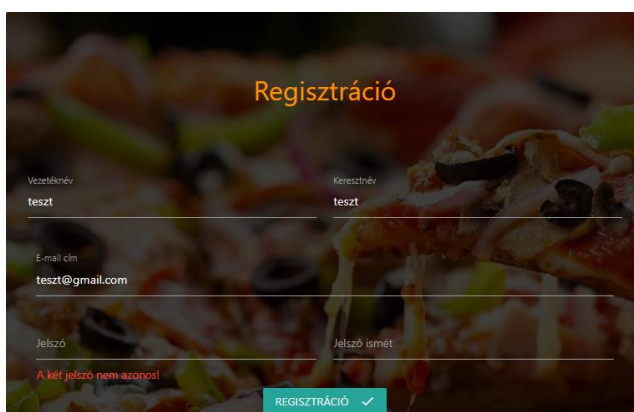
Regisztráció tesztelése: Amennyiben nem rendelkezik a felhasználó regisztrációval, de szeretnénk, hogy rendelkezzen, abban az esetben regisztrálnunk kell a felületre. A regisztrációnál adtunk meg kritériumot. Ezeket a kritériumokat fogom tesztelni.

1. Ha a vezetéknév kevesebb mint 2 karakter:
2. Ha a keresztnév kevesebb mint 2 karakter:
3. Ha az e-mail cím nem megfelelő formátumú, hiányzik belőle a @:
4. A jelszó értéke kevesebb mint 6 karakter:



The screenshot shows a registration form titled "Regisztráció" with a background image of a pizza. The form has four input fields: "Vezetéknév" (Last name) with the value "t", "Keresztnév" (First name) with the value "t", "E-mail cím" (Email address) with the value "tesztgmail.com", and "Jelszó" (Password) with the value "teszt". The "Jelszó ismét" (Repeat password) field is empty. Below the "Vezetéknév" and "Keresztnév" fields, there is a red error message: "Minimum 2, maximum 30 karakter!". Below the "E-mail cím" field, there is a red error message: "Nem megfelelő formátum!". At the bottom, there is a green button labeled "REGISZTRÁCIÓ" with a checkmark icon.

5. Ha a két jelszó nem megfelelő:



The screenshot shows the same registration form as before, but with the "Jelszó" (Password) field filled with "teszt" and the "Jelszó ismét" (Repeat password) field also filled with "teszt". A red error message appears below the "Jelszó" field: "A két jelszó nem azonos!". The "REGISZTRÁCIÓ" button is still visible at the bottom.

6. Ha egy olyan e-mail címmel próbál meg regisztrálni a felhasználó, ami már használatban van:

Lépése	Teszteset	Elvárt eredmény	Sikeresség
1.	Regisztrációs felület vezetéknév ellenőrzése	A rendszernek hibát kell írnia, mivel minimum 2, maximum 30 karakter	Sikeres
2.	Regisztrációs felület vezetéknév ellenőrzése	A rendszernek hibát kell, írnia, mivel minimum 2, maximum 30 karakter	Sikeres
3.	Regisztrációs felület e-mail cím ellenőrzése	A rendszernek hibát kell, írnia, nem megfelelő formátum miatt.	Sikeres
4.	Regisztrációs felület e-mail cím ellenőrzése. Nem írunk bele „@”- ot	A rendszernek hibát kell, írnia, nem megfelelő formátum miatt.	Sikeres
5.	Regisztrációs felület jelszó mezők ellenőrzése. Kevés értéket adunk meg.	A rendszernek hibát kell írnia, minimum 6, maximum 30 karakter	Sikeres

6.	Regisztrációs felület jelszó mezők ellenőrzése. Sok értéket adunk meg.	A rendszernek hibát kell írnia, minimum 6, maximum 30 karakter	Sikeres
7.	Regisztrációs felület jelszó mezők ellenőrzése. A két mezőbe különböző értékeket írjunk	A rendszernek hibát kell írnia, mivel a két formátum nem megfelelő	Sikeres
8.	Belépés felület. A két mezőt üresen hagyjuk	A rendszernek hibát kell írnia, mivel nem kapja meg a megfelelő formátumot	Sikeres
9.	Belépés felület. A jelszó mezőt üresen hagyjuk	A rendszernek hibát kell írnia, mivel üres az egyik mező	Sikeres
10.	Account felület. Olyan e-mail címet és jelszót választunk, amivel nem lett még regisztrálva	A rendszernek hibát kell dobnia, mivel az adatbázisban nem szerepel az az adat, amivel megpróbálunk bejelentkezni	Sikeres
11.	Account felület, belépési adataim módosítása. Az új e- mail cím nem megfelelő formátumú, hiányzik a „@”	A rendszernek hibát kell dobnia, mivel nem megfelelő az új e-mail cím formátuma, ezért nem fogja engedni a felülírást	Sikeres

12.	Account felület, belépési adataim módosítása. Az új jelszóhoz kettő darab karaktert írunk be	A rendszernek hibát dobnia, mivel a minimum karakterek száma 6	Sikeres
13.	Account felület, belépési adataim módosítása. Az új jelszóhoz 31 darab karaktert írunk be	A rendszernek hibát dobnia, mivel a maximum karakterek száma 30	Sikeres
14.	Admin felület. Ha olyan adatot adunk meg, amely nem rendelkezik admin joggal.	A rendszernek ki kell írnia, hogy hibás adatok	Sikeres
15.	Admin felület. Ha az e-mail cím mezőt üresen hagyjuk	A rendszernek ki kell írnia, hogy a mező üres	Sikeres
16.	Admin felület. Ha a jelszó mezőt üresen hagyjuk	A rendszernek ki kell írnia, hogy a mező üres	Sikeres
17.	Admin felület. Ha mind a kettő mezőt üresen hagyjuk, akkor is hibát kell dobnia	A rendszernek ki kell írnia, hogy a mező üres	Sikeres

18.	Admin felület. Userek-> Adatok. Abban az esetben, ha egy user e-mail címét nem e-mail formátumban adjuk meg	Hibát kell kiírnia, mivel nem megfelelő a formátum.	Sikeres
19.	Admin felület. Userek-> Adatok. Abban az esetben, ha egy user e-mail címét kitöröljük és úgy mentenénk	Hibát kell kiírnia, mivel nincsen megadva e-mail cím és az e-mail címet nem engedhetjük törölni	Sikeres
20.	Admin felület. Userek-> Adatok. A user vezetéknévét kitöröljük és úgy mentjük	A rendszernek fel kell hívnia a figyelmet arra, hogy minimum 2, maximum 30 karakter	Sikeres
21	Admin felület. Userek-> Adatok. A user keresztnévét kitöröljük és úgy mentjük	A rendszernek fel kell hívnia a figyelmet arra, hogy minimum 2, maximum 30 karakter	Sikeres

9. Továbbfejlesztési lehetőségek

9.1. Elfelejtett jelszó

Ha a felhasználó elfelejti a belépéshez szükséges jelszavát, akkor legyen egy elfelejtett jelszós link, ahol meg kell adnia a felhasználónak a regisztrációkor használt e-mail címét és ha a megadott e-mail cím létezik az adatbázisban, akkor küldjön ki arra a címre egy linket, aminek a segítségével meg lehet változtatni a korábbi jelszót, az adatbázisban pedig egy jelszó update történjen, php mailer segítségével valósuljon meg az automatikus e-mail küldés.

9.2. Rendelés esetén e-mailes tájékoztató kiküldése a felhasználó felé

Amikor a felhasználó elküld egy megrendelést a weblapon, akkor kapjon egy visszaigazoló e-mailt, amit a rendszer automatikusan generál, melyben szerepel egy olyan szöveg, hogy köszönjük a rendelését, az ön rendelése sikeresen beérkezett éttermünkbe, hamarosan megkezdjük annak a feldolgozását. Ezt követően, amikor az admin átállítja a megrendelés státuszát elkészültre, akkor menjen egy újabb e-mail azzal a szöveggel, hogy a megrendelése elkészült. Csak itt figyelembe kell venni azt is, hogy milyen átvételi móddal érkezett be a megrendelés és annak függvényében megy ki a megfelelő e-mail.

9.3. Új pizza felvitele adatbázis használat nélkül

Egy olyan felület kiépítése a system_admin oldalon, ahol lehetőség van egy új termék feltöltésére. A feltöltés valójában egy INSERT INTO utasítást hajtana végre a megfelelő táblázatokban az adatbázisban. Az admin felületen lenne lehetőség arra, hogy az új pizzáról fényképet feltöltsön az admin és a pizzához való leírást, feltétek megadása is ott történne. Ezt követően, ha ezzel elkészült az új pizza, amit felvett az bekerülne a többi rendelhető pizzák közé.

10. Összegzés

A tanulmányaim során rengeteg új és hasznos dologgal találkoztam. Mindeközben a szoftver írása alatt lehetőségem nyílt elsajátítani a saját magam által talált segítségnek felkeresését az interneten. Ide sorolnám, hogy mit és hol keressek, milyen kulcsszavakat használjak egy adott keresés közben, illetve mik azok az oldalak, melyek biztos segítséget tudnak nyújtani akkor, amikor szükség van a segítségre. A projektem készítése alatt úgy érzem, hogy a programozói tudásom sokat fejlődött, mivel a programkód írása közben felmerülő problémákat sikerült valamilyen úton-módon megoldanom és a megoldást implementálni a programkódba. Mindezek mellett, sokkal magabiztosabbnak érzem magam és a tudásomat is ebben a témában. Számomra fontos volt az, hogy megértsek egy másik ember által írt programkódot, amit ezelőtt nem nagyon tudtam értelmezni. Az órák előrehaladtával éreztem, hogy én is napról napra jobban értem azt, amit esetleg másik csoporttársam megmutat, illetve beszél róla. Korábban az adatbázis lekérdezések írása is nehézséget jelentett, mivel számomra eléggé átláthatatlannak tűnt az, hogy miből mi következik, mostanra viszont ez megváltozott, mivel sokkal jobban tudom értelmezni a kódokat ebben a témában is.

A téma választásom is segítséget nyújtott, mivel sikerült a gondolataimat és az elképzeléseimet megvalósítanom, úgy, ahogyan azt elképzeltem. A családi körben, akinek hasznos lehet egy hasonló oldal, számára is megnyerő formát tudtam mutatni az elkészült projektemmel kapcsolatban. Úgy gondolom, hogy a feladattal kapcsolatban az elképzeléseimet megvalósítottam, és nem kizárt, hogy a jövőben az általam kigondolt funkciókból valamennyi részlet felhasználásra fog kerülni.

11. Forrásjegyzék

1. képek elérése:

- https://www.google.com/search?tbs=simg:CAQSgAEafgsQsIynCBpiCmAIAxIooRajFqIWoBalFrQLqwukFqoFnxaXJ6QizynSN84pviHRN5ArpyfMKRowmRAv9UPT-ahqpdfJ_1Oj8XSr4UCZsmRJDU81QnytPh5I4bYHqyCI_1GxMMruzlgtNPIAQMCxCORv4IGgoKCAgBEgTTPKIYDA&sxsrf=ALiCzsZ4-YV1_5G9InnC8fyF0LeKFEJsKA:1665912717754&q=alimentos+que+da%C3%B1an+la+salud&tbm=isch&sa=X&ved=2ahUKEwj9wJK-uOT6AhUNh_0HHUXaC4IQwg56BAgHEAE&biw=1920&bih=969&dpr=1#imgsrc=kgy1tAMCasBPem
- https://lens.google.com/search?p=AU55jv2rigNSAGgn00hN9pKLoR7-o2bdkEMuKoMwSYO-c9O1EOSfNGpkCyUKq8Bbvm5_MoMazuQ5V5pdvS4rgzOP9jRDDg6RIjauyIYkHy mN5jMPGHfOD8WOnTCJXpRn9Ox6zIUPLZaIqsLCZpkSUiXOnahi0AHWt7XrKe_KgMz-IlcqZjezh3hARRntAXsblCINHjeepKhzzAr9AkOxSePVgMka5bvWNBOI9V4W4e_s-FYgwVvvfEs_bF1DBKvhkjodSdlcuqB0qchyc2M2-vA8al1EXxKFARGkypviHro-ofCys0QcjVoysJcZ8pWnDxAB232bbLxczugXV-aNK8x42n5HZ0KdQ31ijQlaj1glsobtrCluiT8lzQ%3D&ep=gisbubb&hl=hu#Ins=W251bGwsbnVsbCxadWxsLG51bGwsbnVsbCxadWxsLG51bGwsIkVrY0tKRfUwTVRWbU5qazJMV1kwTVRFdE5HVm1NUzFoTVRRNUxUUTNPVEk0T0dZM1pEUXdZaElmZHpKdFl6VnlURVpTYjBWbFVVUjFWV2RKVkUxaWJEZDNla1ZOUTFCb1p3PT0iXQ==
- https://lens.google.com/search?p=AU55jv1V4Cg8tSmXwVYOWyvvzUdtsF1R-golTz1mVQvNtZmFCAMHqpuJbwBVyUNc-VudGXuBxID-77c8kfECVn-oABxyKoqiY5OeUApz8bLNnNYUs9um0UsD_K8e_AlKbgLxFXo2tdZRhPCjEdrxoTJjcO0zzkyPU2c0f69ufG4P5RiPJvocO-HFp3Ynia0hwm4w064NfNcFdm7eKJfXIPDNwHX00RYGWBZtfni6Lg4QJWCjyDt xgIiaxS2F4FT8V47sDjnGLJPBz-b7H1DLKTgGm477eZ7QpAnfLbT7-3q8aUOIEj0MdJBUI0S9v16ytcBccuLpKxaoMmWG36HbNjh-WIBZfdxVSQdE0ev0eqwLuYC-Jqtb2TrsGx0%3D&ep=gisbubb&hl=hu#Ins=W251bGwsbnVsbCxadWxsLG51bGwsbnVsbCxadWxsLG51bGwsIkVrY0tKR0U1WmpReE1XSTJmVEZrWXpRdE5HUm1aaTA0TXpjeUxUQmxNMk16T0RBMk1qQmxZaElmWXpoeVpHNVBhbK5LYVVsa1VVUjFWV2RKVkUxaWJEUIJNbGh6UTFCb1p3PT0iXQ==
- https://lens.google.com/search?p=AU55jv2VeikAnQVpfvYrMCg9cIktwAZ5h2SYdoJOidXOzvkJRSak5p3AfHbFM2XhJH3zGS6oUb1_pW4SaS9Digv61Ui6rvlhZjOF240bF4zflhoFLY1P9YoLi68zRBiBPqElhcaEKFA8KwfiWhwc-Eh8eOrJH_acADLY3tANRcg_uJqU5K7Oxfjk4VbPd1HGUFaG5XHUBA8l061E2AymJp0Ehve9ABDiEiGZqFjq8CXJX-VjFZWO-gdXTvCdA5PDrDioZ-y00AXYAzwBnYr1U984C5zPqm-ZefgJZ6743z6nY_Oy-sIDsgOpoy7AtsO2CjLOp_6Sipz7vZAMko0lQ1DIaHdaEB1ruV8b-

[XppCi4aR1pIJNCW1z-3&ep=gisbubb&hl=hu#lms=W251bGwsbnVsbCxdWxsLG51bGwsbnVsbCxdWxsLG51bGwsIkVrY0tKR1U1TXpneU1ETXhMVEUzWldFdE5EaGlOaTFpTmPZeUxUZGxOekl3TnpjM05qSXdoQklmUIRGa2NXb3phbXBSVWxsbVWVUjFWV2RKVKUxaWJEVm5hVzlGUTFCb1p3PT0iXQ==](https://xppCi4aR1pIJNCW1z-3&ep=gisbubb&hl=hu#lms=W251bGwsbnVsbCxdWxsLG51bGwsbnVsbCxdWxsLG51bGwsIkVrY0tKR1U1TXpneU1ETXhMVEUzWldFdE5EaGlOaTFpTmPZeUxUZGxOekl3TnpjM05qSXdoQklmUIRGa2NXb3phbXBSVWxsbVWVUjFWV2RKVKUxaWJEVm5hVzlGUTFCb1p3PT0iXQ==)

- https://lens.google.com/search?p=AU55jv012etXt3aKcZcVYNreyC5GpICZiN_wqKU_2KhkVfm73VxuXql42fQoNk7hIq1sNeC3B-dku6kJRRx9rCiCtiwugjgg4td2f50uaInu8_M973qtEg-LIGQfI_oROcCIj_UZZSOS7WtomzQhULO6hYcirYx90t3e7La5NsWzFROCWm2fOzdIbdy2xhNUfGmoEq0tjPsR3sJsdDaZce3co018OKqHMgSMuZNPjhfy9wJwUXuKMtHamP72BBfxN9imCy6JwIsDiulsFnn0pzhbUBAbrDTisIlaeyEoDFj8ixtUYJnAh74Usgl892vG9aemJPphSgW1Fx4PbC5gB35k_5ivM7Wug79yDC6WsGUOj8ln1wHWDWx9&ep=gisbubb&hl=hu#lms=W251bGwsbnVsbCxdWxsLG51bGwsbnVsbCxdWxsLG51bGwsIkVrY0tKR1ExTTJJelpUZGpMVEJoTW1ZdE5EQTVZaTFoTXpGbUxUYzBaRE01T0dSaVpESXhOeElmWnpKa05VbE5UWFpSTWtsWFVVUjFWV2RKVKUxaWJEZEJjVmxWUTFCb1p3PT0iXQ==
- <https://materializecss.com/icons.html>
- https://hu.wikipedia.org/wiki/Visual_Studio_Code
- <https://www.php.net/manual/en/function.json-decode.php>
- https://www.w3schools.com/php/php_forms.asp
- https://www.w3schools.com/php/php_includes.asp
- https://www.w3schools.com/php/php_sessions.asp
- <https://www.php.net/manual/en/function.session-start.php>
- https://www.w3schools.com/php/filter_validate_email.asp
- <https://materializecss.com/getting-started.html>
- <https://materializecss.com/color.html>
- <https://materializecss.com/grid.html>
- Robert C Martin - Tiszta kód