



HambiBambi

Étel rendelő szolgálat

Kertész Ruben,
Molnár Zsolt

Nógrád Vármegyei
Szakképzési
Centrum

Szent-Györgyi
Albert Technikum

Szoftverfejlesztő
és -
tesztelőtechnikus
(szakmai
azonosító: 5-0613-
12-03)

2025. április. 28.

Tartalom

Bevezetés.....	5
Témaválasztás	5
1. Felhasználói dokumentáció	5
1.1. Kezdőlap.....	6
1.2. Étlap	7
1.3. Rólunk	9
1.4. Belépés/Regisztráció	10
1.5. Kosár	11
1.6. Rendelés véglegesítése.....	13
1.6.1 Sikeres rendelés.....	14
1.7. Profil.....	14
1.7.1. Profil adatok szerkesztése	16
1.8. Admin.....	18
1.8.1. Bejelentkezés.....	18
1.8.2. Admin panel	19
1.8.3. Terméklista megtekintése.....	19
1.8.4. Új termék felvitele.....	21
1.9. Reszponzivitás.....	21
1.9.1 Telefonos nézet:	22
1.9.2. Tabletes nézet:.....	23
2. Fejlesztői dokumentáció.....	23
2.1. Felhasznált technológiák	23
2.1.1. VSC	23
2.1.2. HTML.....	24
2.1.3. Angular JS	24
2.1.4. GitHub	25

2.1.5.	JSON	25
2.1.6.	CSS.....	25
2.1.7.	MySQL.....	26
2.1.8.	JavaScript	26
2.1.9.	XAMPP	26
2.1.10.	API	26
2.1.11.	AJAX.....	27
2.1.12.	PHP.....	27
2.2.	Kódmagyarázat.....	27
2.2.1.	Adatbázis kapcsolódás	30
2.2.2.	Kezdőlap.....	30
2.2.3.	Rólunk	32
2.2.4.	Belépés/Regisztráció	35
2.2.5.	Étlap	39
2.2.6.	API	41
2.2.7.	Kosár	43
2.2.8.	Rendelés véglegesítése	44
2.2.9.	Profil.....	46
2.2.9.1.	profile.php	46
2.2.10.	Admin.....	48
3.	Az adatbázis célja.....	51
3.1.	Tervezési lépések	51
3.2.	Egyedek meghatározása	52
3.2.2.	Settlements (Települések)	52
3.2.3.	Counties (Megyék).....	52
3.2.4.	Baskets (Kosarak)	52
3.2.5.	Payments (Fizetési módok)	52

3.2.6. Quantity_units(Mértékegységek)	52
3.2.7. Order_Statuses (Rendelés Státusza).....	52
3.2.8. Orders (Rendelések).....	52
3.2.9. Products (Termékek)	53
3.2.10. Products_Categories (Termékek kategóriái).....	53
3.2.11. Products_Groups (Termékek csoportjai)	53
3.2.12. Admin.....	53
3.2.13. Reviews (Értékelés).....	53
3.3. Kapcsolatok meghatározása	53
3.3.1. 1:N kapcsolatok meghatározása	53
3.3.2 N:M kapcsolatok felbontása.....	54
3.4 Táblák.....	55
3.4.1. Users (Felhasználók).....	55
3.4.2. Settlements (Települések)	56
3.4.3. Counties (Megyék).....	56
3.4.4. Orders (Rendelések).....	57
3.4.5. Baskets (Kosarak)	58
3.4.6. Payments (Fizetési módok)	58
3.4.7. Quantity_units(Mértékegységek)	59
3.4.8. Order_Statuses (Rendelés Státusza).....	59
3.4.9. Products (Termékek).....	60
3.4.10. Products_Categories (Termékek kategóriái).....	61
3.4.11. Products_Groups (Termékek csoportjai)	61
3.4.12. Reviews(Értékelések).....	62
3.4.13. Admin.....	62
3.5. Adatbázis diagram.....	63
3.6. Adatbázis továbbfejlesztési lehetőségek	64

4.	API tesztelése	64
4.1.	GET kérések tesztelése	65
4.2.	GET kérés tesztelése id alapján.....	66
4.3.	POST kérések tesztelése.....	67
4.4.	PUT kérések tesztelése.....	67
4.4.	DELETE kérések tesztelése	69
5.	Összefoglalás.....	70
5.1.	Továbbfejlesztési lehetőségek.....	70
5.1.1.	Nyelv választása:.....	70
5.1.2.	Sötét mód:	70
5.1.3.	Keresés funkció:.....	71
5.1.4.	Adatbázis szerveren történő működtetése:.....	71
5.1.5.	Adatok szerkesztése a rendelés véglegesítésénél:	71
5.1.6.	Bug-ok javítása, optimalizálás, karbantartás:.....	71
5.1.7.	Rendelés visszaigazolás, email küldése:	71
5.1.8.	Felhasználó adatainak erősebb védelme	71
5.1.9.	Előző rendelések listája:.....	71
5.2.	Felmerült akadályok.....	72
5.3.	Források:.....	72
5.4.	Mellékletek.....	73
5.4.1.	Forrásállományok.....	73
5.4.2.	Adatbázis	73
5.4.3.	Xampp	73
5.4.4.	Dokumentáció	73

Bevezetés

Csapatunk két személyből áll: Kertész Ruben és Molnár Zsolt. Projektünket közösen kezdtük el az információk kutatásával. A következő fő lépés az adatbázis kialakítása volt. Az elosztás végül az lett, hogy Ruben csinálta az adatbázis nagyobb részét, míg Zsolt a dokumentációból vette ki a nagyobb részt. Közben a weboldal fő oldalát közösen megcsináltuk, valamint a bejelentkezés és regisztráció oldalba is belekezdünk. A többi oldal elkészítését a lehető legjobban próbáltuk meg egymás között elosztani. A célunk az volt, hogy mind a ketten csináljunk valamennyit a projekt backend és frontend részéből is. Ruben az Étlap, a Profil, a Rendelés véglegesítését, és a Kosár felület kialakítását kapta feladatnak, Zsolt a Kapcsolat, valamint a Kezdőlap és a Bejelentkezés/Regisztráció oldal befejezése volt a feladata. Az Admin felületet gondoltuk utoljára, amely szintén közösen készült el.

Témaválasztás

Az elmúlt időben nagyon sokat gondolkoztunk azon, hogy mi legyen a projektfeladatunk témája. Végül egy nagyon egyszerű téma mellett döntöttünk, amely egy általunk kitalált étterem rendelésének elvégzésére szolgáló weboldal. Ezt a témát azért választottunk, hiszen ez egy olyan dolog, amit mindketten kedvelünk.

A kutatásunk közben sok étterem weboldalát is megnéztük köztük a Balassagyarmati Dolce Vita, valamint az AstaPasta éttermek oldalait. Végül a Farm Burger nevű Váci üzletet vettük példaként, az ő weboldaluk olyan, amiről rendelni is lehet, és mi is egy hasonló oldalt hoztunk létre. Az oldalon több funkció is elérhető, mint például rendelni, bejelentkezni, vagy éppen regisztrálni. Oldalunkon elérhető a rendelés és házhozszállítás opció. Felhasználóbarát, könnyen kezelhető, egyszerűen elérhető és kiválasztható a képpel illusztrált termék. Weboldalunkkal szeretnénk kényelmessé és egyszerűvé tenni az ételrendelést.

1. Felhasználói dokumentáció

A weboldalon ételeket és italokat böngészhetünk, kosárba helyezhetünk, majd meg is rendelhetünk. A felhasználók regisztrálhatnak, majd beléphetnek a profilukkal a weboldalra, értékelést is írhatnak. A bejelentkezés után módosíthatják a profiluk adatait, vagy szükség esetén kiléphetnek. Az oldalakon szereplő képeket, valamint a vármegyék és települések listáját a mesterséges intelligencia segítségével generáltattuk le, és adtuk hozzá a projekthez. A felhasznált színeket azért választottuk, mivel a kutatásunk közben megkerestük azokat a színeket, amelyek étvágygerjesztést mutatnak. Ennek eredményeként az oldal logója is ezekkel a színekkel készült el.

1.1. Kezdőlap

Az "index.php" a kezdőlap, ahol a felhasználó landolni fog. A weboldal csúcán szerepel a navigációs sáv, ahonnan el lehet jutni a weboldal további részeire, pl.: Étlap. A navigációs sáv bal oldalán helyezkedik el a logó.



Következik a tartalom, itt egy rövid bemutatkozás szerepel az étteremről, a szöveg mellett egy kép található. A szöveg alatt megfigyelhető egy újabb bejegyzés, ami az alapanyagokról számol be.



A Hambibambi Étteremben a minőség és az ízélmény találkozik! Kézzel válogatott alapanyagokból, friss és zamatos hozzávalókból készítjük fogásainkat, hogy vendégeink minden falattal egy kulináris élményt élhessenek át. Legyen szó szaftos burgerekről, ropogós sültkrumpliról vagy egy frissítő salátáról, nálunk minden étel szeretettel készül.

Ételeinket úgy álmodtuk meg, hogy mindenki megtalálja a kedvencét: a klasszikus ízek kedvelői és a különleges gasztronómiai élményeket keresők egyaránt. A zamatos húsoktól kezdve a vegetáriánus finomságokon át a házi készítésű szószokig minden fogásunk az autentikus ízeket és a modern gasztronómiai trendeket ötvözi. Gyere és tapasztald meg a Hambibambi élményt!



Ezt követi egy nagyobb kártya, amiben három véletlenül kiválasztott kép követ, ami a kiemelt ajánlatokat szolgálja. Képeken ételek szerepelnek, alattuk a termék neve tartozik.

Kiemelt ajánlataink



Bacon burger



BBQ burger



Gombás burger

Végül, de nem utolsó sorban szintén egy kártya szerepel, ami magáról a személyzetről szól.



A weboldal legalján egy lábléc szerepel, ami ugyan azokat az elemeket tartalmazza, mint a navigációs sáv, viszont itt megfigyelhető az étterem neve, és tükrözve van a navigációs sávhoz képest.



1.2. Étlap

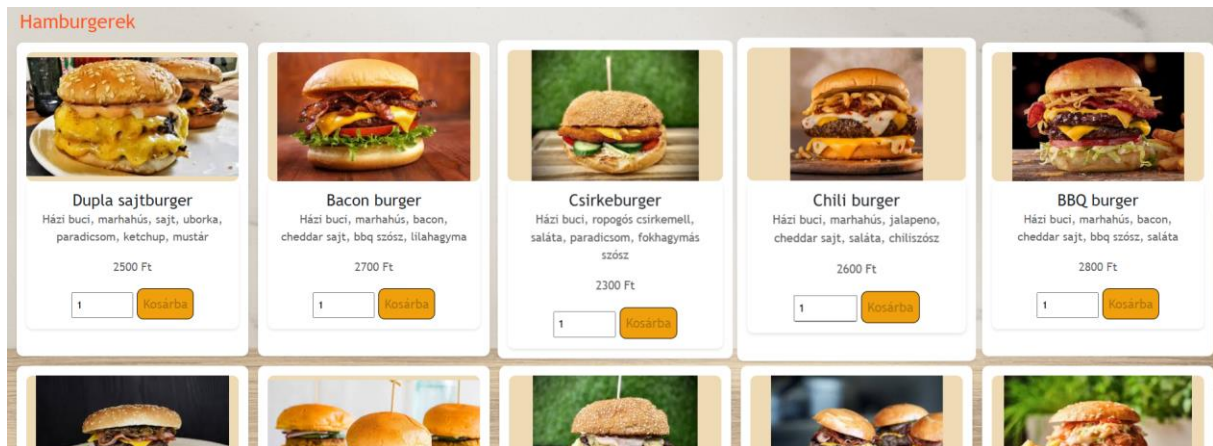
A projektünkben lévő Étlap weboldalon találhatóak az étterem termékei. Az oldal fejléc szekciójában található a menüsor, amely segítségével lépkedhetünk a másik oldalak között. Emellett az étterem logója, és a kosár elérési ikonja látható.



A fejléc alatt egy oldalon belüli menüsor látható, amely segíti a termékek kikeresését a kategóriájuk alapján.

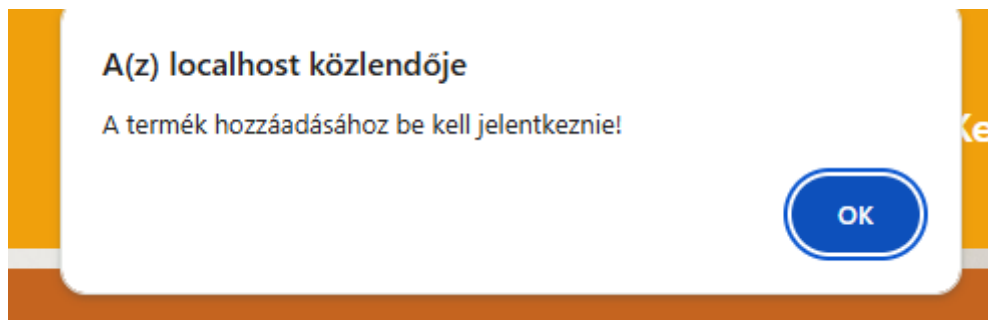


Az oldal törzse tartalmazza a termékeket, amelyek a kategória nevük alapján csoportosítva jelennek meg.



Ha a kurzort rávisszük egy termék kártyájára, akkor a nagysága megváltozik. Minden termék egy saját kártyában jelenik meg, pontosabban a hozzárendelt képük, a nevük, az áruk forintban mérve, valamint a termék részletes leírása. Ezek mellett tartozik még egy beviteli mező, amellyel megadható az adott termékből kívánt mennyiség.

Végül egy gomb is található, "Kosárba" leírással, amely lenyomásával hozzáadjuk a kosárhoz az kiválasztott terméket, amely csak akkor megnyomható, ha a felhasználó be van jelentkezve. Ha viszont bejelentkezés nélkül szeretnénk megtenni a hozzáadást, akkor egy figyelmeztető ablak ugrik fel:



Egy navigációt segítő gomb is megjelenik akkor, amikor az oldalon le lett görgetve. A gomb lenyomásával visszatérünk az oldal elejére, ezzel is gyorsítva az oldalon a navigációt.



Az oldal láblécében egy menüsor található, amely hasonló a fejlécben lévő menüsorhoz. Itt is megjelenik az oldalak közötti navigációt segítő szekció, emellett az étterem neve, valamint a logója.

1.3. Rólunk

A navigációs sáv után a felhasználó öt darab kártyára lehet figyelmes, amikhez fel van tüntetve az elérhetőségek, a nyitvatartás, a közösségi média, az értékelés, amit akkor használhat a felhasználó, miután bejelentkezett a weboldalra. Az utolsó sorba lévő kártyában látható a többi felhasználó által létrehozott értékelés, itt megjelenik a nevük, 1-5-ig hányas értékelést adott, az értékelés leírása, és a dátum, amikor a felhasználó közzé tette az értékelést.

Elérhetőségek

Telefonszám: +36 30 123 4567

E-mail cím: info.hambibambi@gmail.com

Cím: 2660 Balassagyarmat, Hamburger utca 12.

Nyitvatartás

Hétfő: 8:00 - 20:00
Kedd: 8:00 - 20:00
Szerda: 8:00 - 20:00
Csütörtök: 8:00 - 20:00
Péntek: 8:00 - 18:00
Szombat 8:00 - 18:00
Vasárnap: Zárva

Közösségi média

[Tiktok](#)

[Instagram](#)

[Facebook](#)

Értékelés

★★★★★
0/5

Üzenet:

✉ Üzenet küldése

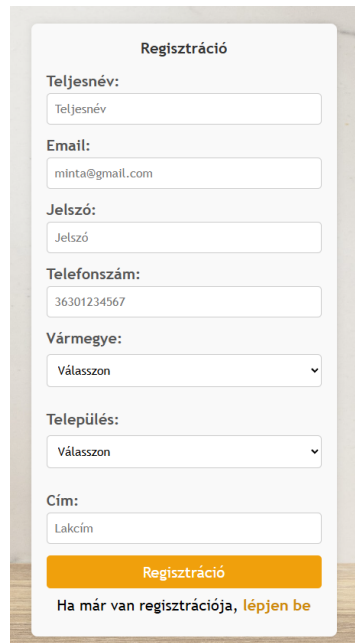
Felhasználói értékelések

Felhasználó	Értékelés	Leírás	Dátum
Minta Péter	5/5	Jó volt minden!	2025-04-26 16:03:25
Kertész Ruben2	4/5	Finom ételek, gyors kiszállítás!	2025-04-26 16:01:24

Lábléc itt is megjelenik az oldal alján.

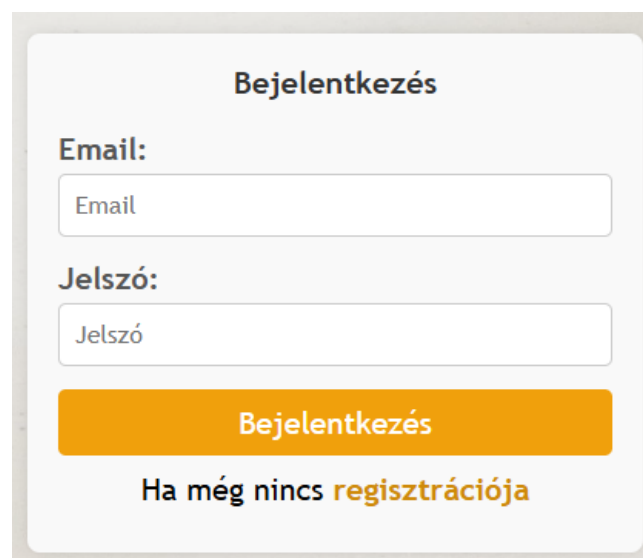
1.4. Belépés/Regisztráció

A felhasználó miután rákattintott a “Belépés/Regisztráció” gombra, alapértelmezetten a regisztráló oldalra irányítja át, ahol a felhasználónak a regisztrálás lehetőséget nyújtunk. Itt megadhatja a teljes nevét, email-címét, egy tetszőleges jelszót, telefonszámát, a megyét, illetve a vármegyéhez megfelelő településeket is be tudja állítani magának a felhasználó.



The image shows a registration form titled "Regisztráció". It contains several input fields: "Teljesnév:" (Full name) with a placeholder "Teljesnév", "Email:" with a placeholder "minta@gmail.com", "Jelszó:" (Password) with a placeholder "Jelszó", "Telefonszám:" (Phone number) with a placeholder "36301234567", "Vármegye:" (County) with a dropdown menu showing "Válasszon", "Település:" (Settlement) with a dropdown menu showing "Válasszon", and "Cím:" (Address) with a placeholder "Lakcím". Below the fields is an orange button labeled "Regisztráció". At the bottom, there is a link that says "Ha már van regisztrációja, lépjen be".

Miután megadta az adatait, a “regisztráció” gombra rá kattintva a felhasználó adatai bekerülnek az adatbázisba, és be tud a későbbiekben lépni a weboldalra. Ha már készített egy profilt magának, lehetőséget kap, hogy a “regisztráció” gomb alatt lévő linkre rányom, ami átirányítja a felhasználót a bejelentkezési felületre. A bejelentkezés is egy külön oldalon történik, ahol a felhasználó az email-címét és a jelszavát kell megadnia, hogy be tudjon lépni.



The image shows a login form titled "Bejelentkezés". It contains two input fields: "Email:" with a placeholder "Email" and "Jelszó:" (Password) with a placeholder "Jelszó". Below the fields is an orange button labeled "Bejelentkezés". At the bottom, there is a link that says "Ha még nincs regisztrációja".

Bejelentkezés gombra kattintva a felhasználó átirányul a fő oldalra.

A gomb alatt található egy link, ami a regisztrációs oldalra irányít vissza.

Ha a felhasználó rosszul adta meg valamelyik adatát, akkor egy hibaüzenet jelenik meg a Bejelentkezés felirat alatt.



1.5. Kosár

A kosár a fejléc menüsorában található bevásárló kosár ikon lenyomásával jelenik meg. Ez a funkció nem érhető el a bejelentkezés és regisztráció, a rendelés véglegesítésénél, az profil adatok szerkesztésénél, és az admin felületen sem.

Ha be van jelentkezve:



Ha nincs bejelentkezve:



Ha rányomunk az említett ikonra, akkor egy felugró ablakként lesz látható az oldalon. Itt találhatóak azok a terméket, amelyeket a felhasználó tett bele a “Kosárba” gomb lenyomásával. Ha a kosárba bekerül valamennyi termék, akkor a kosár megjelenése a következő:

Kosár tartalma				
Termék	Név	Mennyiség	Ár	
	Csirkeburger	1	2300 Ft	Termék törlése
	Bacon burger	2	5400 Ft	Termék törlése
	Dupla sajtburger	1	2500 Ft	Termék törlése
Megrendelés		Végösszeg: 10200 Ft		Minden termék törlése

Ha nem kerül termék a kosárba, akkor pedig:

Kosár tartalma				
Termék	Név	Mennyiség	Ár	
A kosár üres				
Megrendelés		Végösszeg: 0 Ft		Minden termék törlése

A felugró ablak jobb felső ablakában egy X gomb helyezkedik el, amely megnyomásával inaktív lesz a kosár felugró ablaka, így tovább lehet böngészni az oldalt. Az adatok egy táblában vannak megjelenítve, a fejlécében magyarázó szövegek vannak. A megjelenített adatok a termék képe, a termék neve, a darabszáma, és a termék darabszám alapján kiszámolt összára. Az utolsó oszlopban az adatok mellett egy “Termék törlése” szöveg van, amely lenyomásával az adott terméket kiveszi a kosárból. A tábla utolsó sorában a “Megrendelés”, a végösszeg, és

a “Minden termék törlése” található. A végösszeg szöveg mellett a kosárba tett termékek összértéke látható. A “Minden termék törlése” szöveg lenyomásával a kosárban lévő összes terméket kiveszi, ezzel üressé válik a kosár. A “Megrendelés” szöveg kattintásával egy új oldalra kerülünk, amely a rendelés véglegesítéséhez szükséges.

1.6. Rendelés véglegesítése

A Rendelés véglegesítése oldal a kosár felugró ablakában lévő “Megrendelés” szöveg megnyomásával nyílik meg, amely csak akkor elérhető, amikor a felhasználó be van jelentkezve. Az oldal tetején egy “Rendelés véglegesítése” ismertető cím látható.

Rendelés véglegesítése

Kosár tartalma

Bacon burger	10800 Ft
Mennyiség: 4	
Csirkeburger	4600 Ft
Mennyiség: 2	
Dupla sajtburger	2500 Ft
Mennyiség: 1	
Végösszeg:	17900 Ft

Számlázási adatok

Teljesnév
Kertész Ruben2

Email
teszt10@gmail.com

Telefonszám
36312102121

Vármegye
Heves

Település
Bükkszentersébet

Lakcím
Rákóczi út 24.

Fizetési mód

☒ Készpénz
☐ Bankkártya
☐ SZÉP-kártya

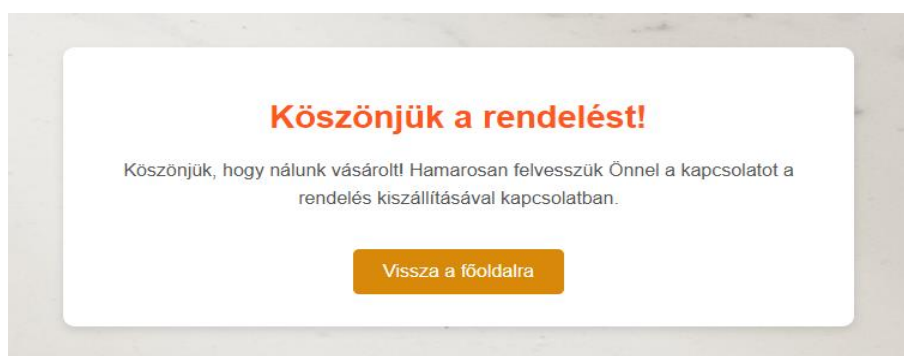
Rendelés véglegesítése <<< Vissza

Az törzs két részre osztható. A bal oldalt a kosár végleges állapota látszik. Ide tartozik a termék neve, a kosárba tett mennyisége, a termék darabszám alapján kiszámolt összára. Az összes kosárba helyezett termék alatt, látható a végösszeg, amely a kosárba tett termékek összértéke alapján kiszámolt érték. A törzs jobb oldalán a számlázási adatok címmel a felhasználó regisztrációkor megadott személyes adatai jelennek meg. Az adatok felhasználó bejelentkezése miatt automatikusan az oldal indulásakor betöltődnek a beviteli mezőkbe. Ide tartozik például a felhasználó e-mail címe, telefonszáma és lakcíme. A személyes adatok alatt rádió gombok segítségével ki lehet választani a fizetési módot, amely lehet készpénz, bankkártya, valamint SZÉP-kártya is.

Az oldal végén egy “Rendelés véglegesítése” gomb található, amely lenyomásával egy másik ablakra kerülünk át. Ha esetleg nem szeretnénk még végezni a rendeléssel, hanem szerkeszteni akarnánk azt, akkor a vissza gombbal visszatérünk az előző oldalra.

1.6.1 Sikeres rendelés

Ha a Rendelés véglegesítése oldalon a “Rendelés véglegesítése” gombra nyomunk, akkor térünk át a Sikeres rendelés oldalra. Az oldalon egy fehér háttérű konténerben egy köszönő szöveg és egy gomb látható, amely segítségével visszatérünk a kezdőlapra.



1.7. Profil

A Profil felület csak akkor lesz elérhető, amint a felhasználó sikeresen bejelentkezik. Ennek következtében az oldalak fejlécében található menüsorban a “Bejelentkezés / Regisztráció” szövegű link egy “Profil” linkre cserélődik, ezzel segíti a profil felület elérését.

Az oldal fejlécében a navigációt segítő menüsor helyezkedik el, amely segíti az oldalak közötti ugrálást és a kosár felület elérését. A törzsben egy magyarázó címsor és egy táblázat látható.

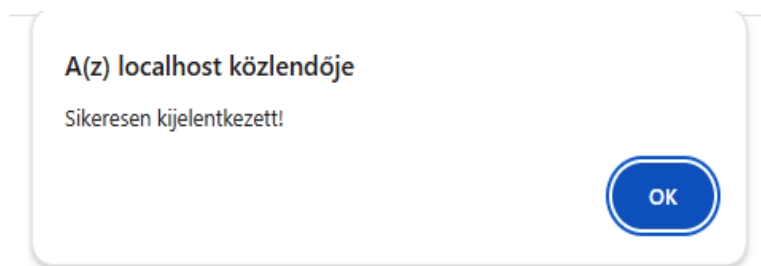
Felhasználó adatai

Teljesnév:	Kertész Ruben
Telefonszám:	+36205462311
E-mail:	rubenkertesz@gmail.com
Megye:	Nógrád
Település:	Balassagyarmat
Lakcím:	Dózsa György út 14.
<div style="display: flex; justify-content: space-around; margin-top: 10px;"><div style="background-color: red; color: white; padding: 5px 10px; border-radius: 5px;"> Kijelentkezés</div><div style="background-color: green; color: white; padding: 5px 10px; border-radius: 5px;"> Adatok módosítása</div></div>	

A bal oldali oszlopban magyarázó szövegek, míg a jobb oldaliban azok az adatok láthatóak, amelyeket a felhasználó adott meg akkor, amikor regisztrált. A tábla alsó sorában két gomb helyezkedik el, amelyek az “Adatok módosítása” és a “Kijelentkezés” szövegekkel vannak ellátva. Az “Adatok módosítása” zöld hátterű gomb lenyomásával átnavigálunk egy másik oldalra, ahol tudjuk szerkeszteni a személyes adatainkat.

A “Kijelentkezés” piros hátterű gomb segítségével ki tudunk jelentkezni a fiókunkból. Így visszakerülünk a kezdőoldalra és újra elérhetővé válik a Bejelentkezés és Regisztráció felület.

A gomb lenyomásával egy figyelmeztető ablak ugrik fel:



1.7.1. Profil adatok szerkesztése

A Profil adatok szerkesztése a profil felületen lévő “Adatok módosítása” gomb lenyomásával nyitódik meg. Az oldalon fejlécében nem jelenik meg a hagyományos menüsor, hanem egy “Adatok módosítása” szöveg.



Adatok módosítása

Teljes név:
Kertész Ruben

E-mail:
rubenkertesz@gmail.com

Telefonszám:
36205462311

Vármegye:
Nógrád

Település:
Balassagyarmat

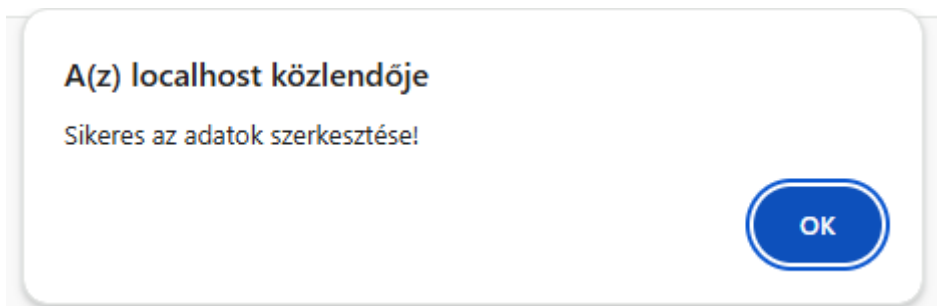
Lakcím:
Dózsa György út 14.

The image shows a user interface for changing a password. It consists of four vertically stacked sections, each with a label and a text input field:

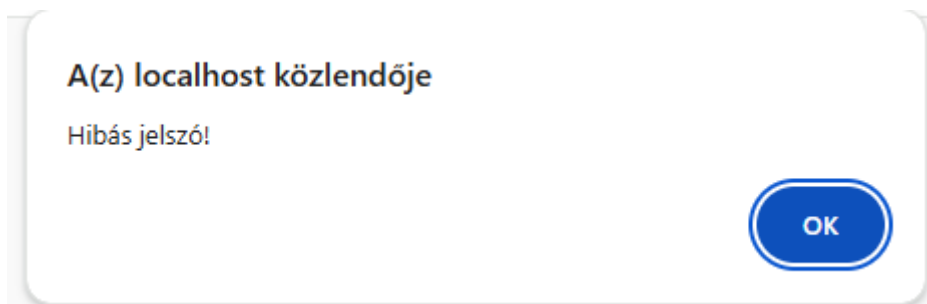
- Jelszó:** The first section with the label "Jelszó:" and a text input field containing "Jelenlegi jelszó".
- Új jelszó (opcionális):** The second section with the label "Új jelszó (opcionális):" and a text input field containing "Új jelszó".
- Új jelszó megerősítése:** The third section with the label "Új jelszó megerősítése:" and a text input field containing "Új jelszó megerősítése".
- Buttons:** The fourth section contains two orange buttons. The top button is labeled "Módosítások mentése". The bottom button is labeled "< < < Vissza".

A törzsben a profil oldalon lévő táblázathoz hasonlóan jelennek meg az adatok. A bal oldali oszlopban szintén magyarázó szövegek jelennek meg, míg a jobb oldaliban beviteli mezők, amelyekben a felhasználó által megadott adatok szerepelnek alapmértékben a jelszón kívül. Itt lehet lecserélni a felhasználói adatokat, mint például lakcímünket, vagy éppen a telefonszámunkat. Ha a jelszót cserélnénk le, akkor meg kell adni az új jelszót, majd újra beírni megerősítésként. A táblázat utolsó sorában két gomb látható. A "<<<Vissza" gomb arra az esetre való, ha mégsem szeretnénk módosítani az adatainkat. Ennek lenyomásával visszatérünk a profil felületre. A "Módosítások mentése" gomb segítségével mentésre kerülnek a módosított adatok és kicseréli a régi adatokat.

A művelet csak akkor történik meg, ha a felhasználó beírja a jelenlegi jelszavát. A lenyomása után figyelmeztető ablak jelenik meg:



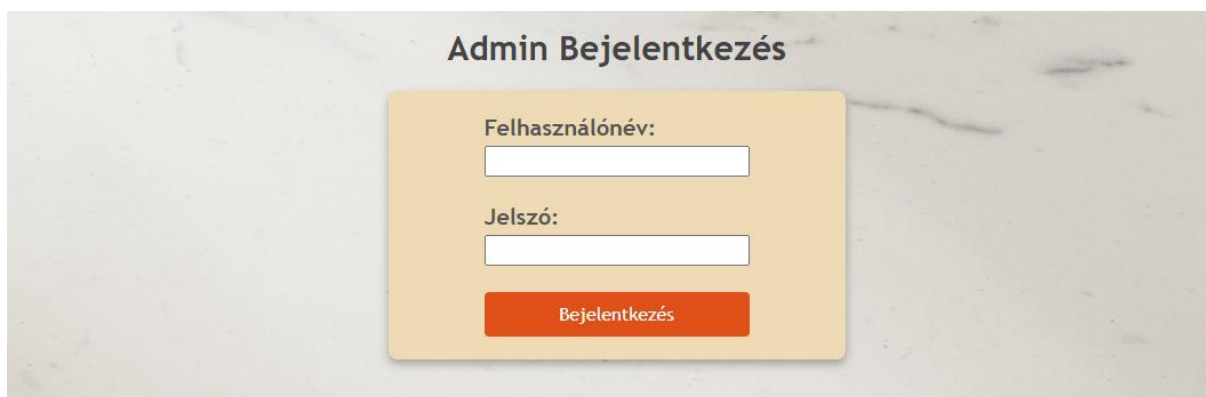
Az esetleges hibás adat megadásakor is egy figyelmeztetés ugrik fel, majd a gomb megnyomásával visszatérünk az oldalra. Példa:



1.8. Admin

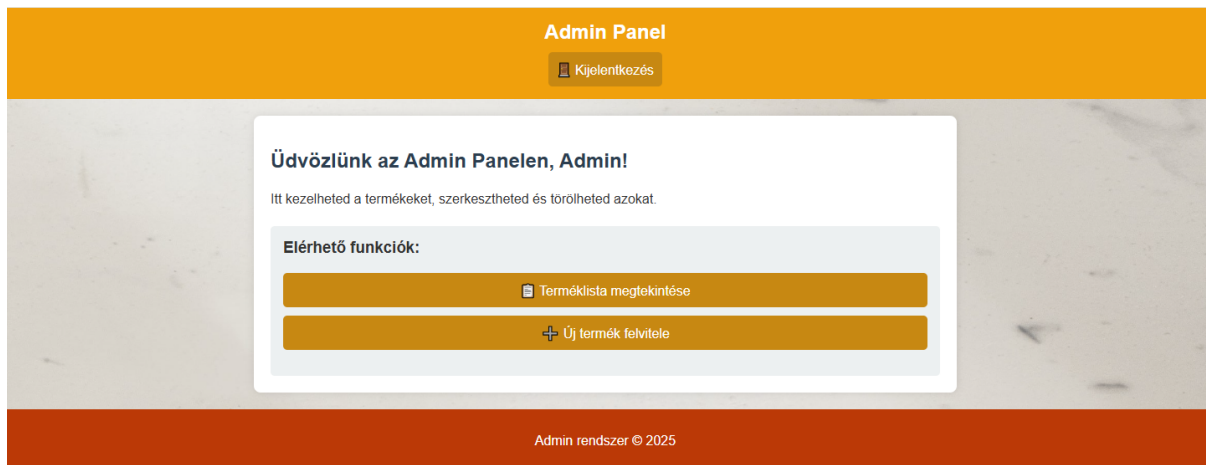
1.8.1. Bejelentkezés

Az admin felület elérése előtt egy bejelentkezési felület látható. Itt helyezkedik el a bejelentkezés struktúra, ahol meg kell adnunk a felhasználónevet és a jelszót.



1.8.2. Admin panel

Ha sikeres a bejelentkezés, akkor átkerülünk az admin panelre.



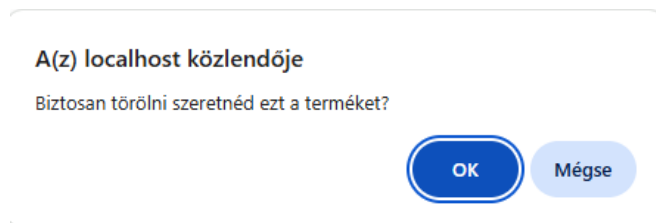
Itt a fejlécben egy “Kijelentkezés” szövegű gomb látható, amely lenyomásával kijelentkezünk a fiókból, és visszakerülünk a bejelentkezési felületre. A láblécben egy copyright szöveg jelenik meg. Az oldal törzsében magyarázó szövegek és két gomb jelenik meg. A “Terméklista megtekintése” gomb megnyomásával átkerülünk a termékek listájára, míg az “Új termék felvétele” gombbal a termék hozzáadása felületre kerülünk.

1.8.3. Terméklista megtekintése

Az oldal tetején egy sárga háttérű vissza gomb helyezkedik el, amely visszavisz az admin panelre. A weboldalon szereplő termékek adatai egy táblázatban jelennek meg. Pontosabban szerepel a termék id-je, neve, ára, részletes leírása, és a képe. Emellett az utolsó oszlopban látható két gomb, amelyek segítségével módosítani, valamint törölni lehet az adott terméket. A táblázat rendezhető id, név, és ár alapján növekvő, vagy csökkenő sorrendben úgy, ha a táblázat fejlécében rákattintunk valamelyikre.

Terméklista					
< < < Vissza					
A termékek id, név és ár szerint rendezhetők, kattintson a fejlécre!					
ID	Név	Ár	Leírás	Kép	Műveletek
39	Cézár saláta	2500 Ft	Grillezett csirkemell, római saláta, parmezán, kruton, Cézár öntet		Módosítás Törölés
40	Görög saláta	2300 Ft	Paradicsom, uborka, feta sajt, olívbogyó, lilahagyma, olívaolaj		Módosítás Törölés
41	Tonhal saláta	2600 Ft	Tonhal, saláta, kukorica, lilahagyma, paradicsom, olívaolaj		Módosítás Törölés
42	Gulyásleves	2200 Ft	Marhahús, burgonya, répa, csipetke, pirospaprika		Módosítás Törölés

Törlés esetén egy figyelmeztető üzenet ugrik fel, majd visszatérünk a terméklistához.



Ha módosítani szeretnénk a terméket, akkor a módosítás gomb lenyomásával egy új felületre térünk át.

Itt módosíthatjuk a termék nevét, árát, leírását, valamint a képét is. Ha végeztünk, a “Módosítás” gomb lenyomásával elmentődnek a módosított adatok, és szintén visszatérünk a terméklistához. Szükség esetén visszaléphetünk a “Vissza” gombbal.

1.8.4. Új termék felvitele

Az új termék felvitelénél egy dobozban megjelenő beviteli mezők jelennek meg. Itt meg kell adni az új termék nevét, az árát, a leírását, és a hozzátartozó kép nevét. Ezek mellett ki kell választani egy legördülő listából a feltölteni kívánt termék kategóriáját, és mennyiségi egységét.

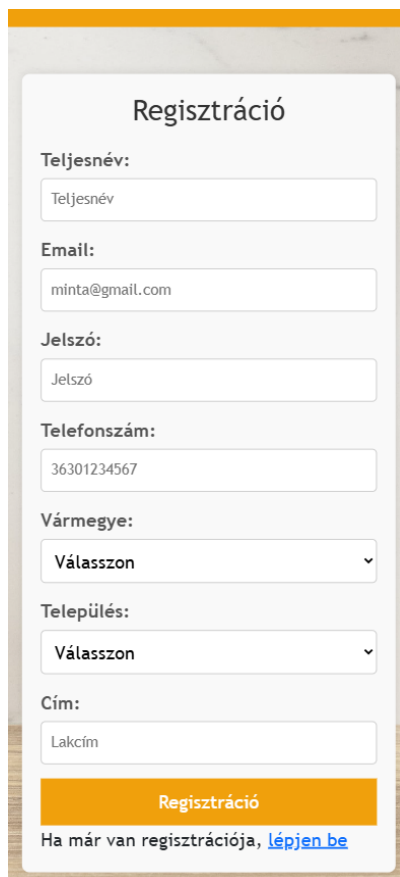
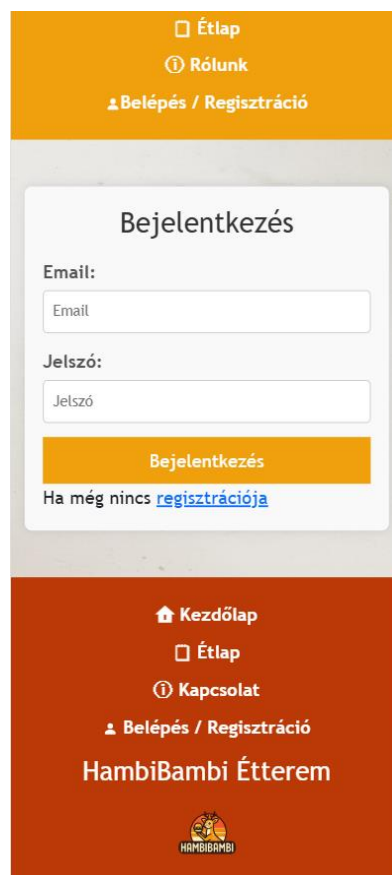
Ha minden ki lett töltve, akkor a “Hozzáadás” gombbal feltöltődik az adatbázisba és visszakerülünk a terméklistához. Ezek után meg fog jelenni az új termék a táblázatban.

78	Torony burger	2050 Ft	Torony burger		Módosítás	Törölés
----	---------------	---------	---------------	---	---------------------------	-------------------------

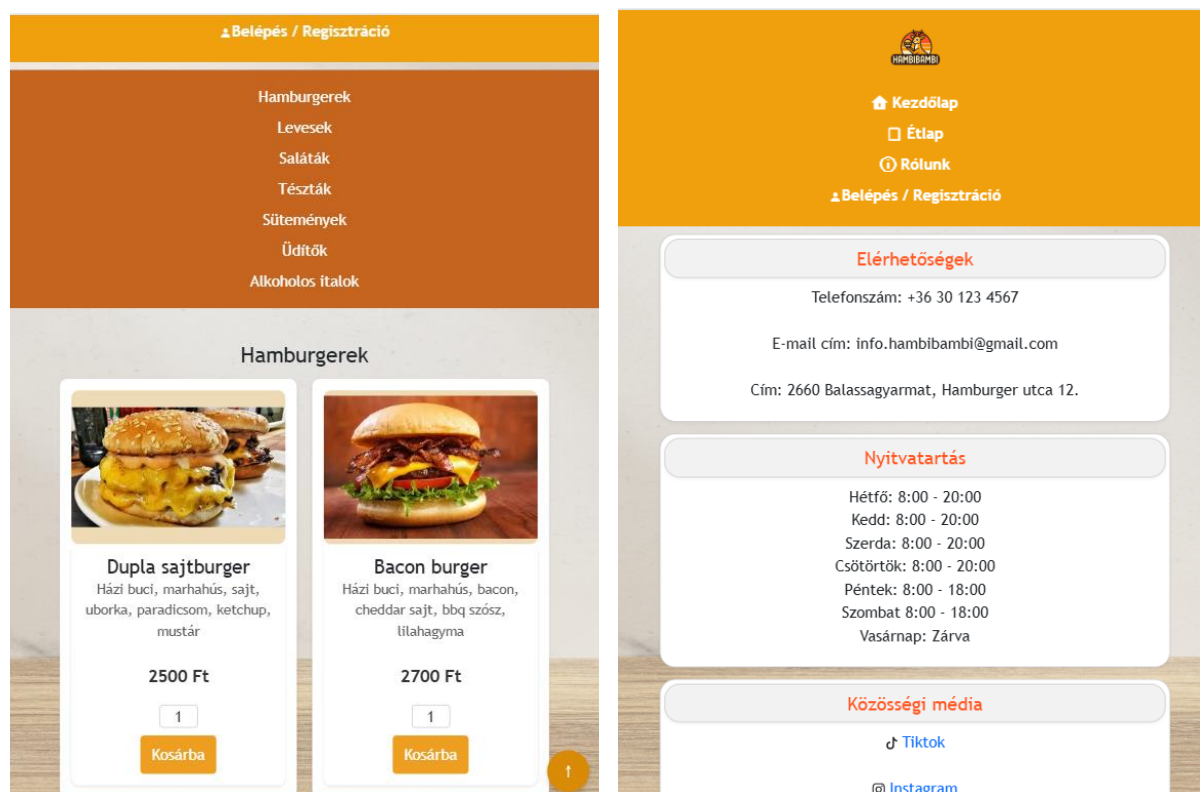
1.9. Reszponzivitás

A rezponzivitás az oldalakon tabletre és mobilra is elkészült a flex kódolás segítségével. Telefonra az Iphone 14 Pro Max eszköz képméreteire készítettük el, míg tableten iPad Mini-re. Néhány oldalon nem teljesen működik, így továbbfejlesztési lehetőségnek írtuk be a kijavítását.

1.9.1 Telefonos nézet:



1.9.2. Tabletes nézet:



2. Fejlesztői dokumentáció

A projekt mappa jól lekövethetőre lett létrehozva, és könnyen lehet keresni benne. A gyökérben a fő oldal php kódja, az adatbázishoz való kapcsolódás php kódja, valamint négy különböző mappa helyezkedik el. A „databases” mappa az elkészült adatbázisok sql fájljait tárolja, a „documents” pedig a projekt bemutatóját és a prezentációját foglalja magába.

Az „assets” mappa három belső mappát tartalmaz, amelyekben a CSS, JavaScript fájlok, és az oldalon szereplő képek találhatóak.

Az „application” két mappát tárol. A „controller” mappában az api, bejelentkezés és regisztráció kezelésének a kódjai, valamint a termék hozzáadásának a kódja van benne. A „view” a projekt többi oldalának mappáit és fájljait tárolja el.

2.1. Felhasznált technológiák

2.1.1. VSC

A Visual Studio Code (rövidítve: VSCode vagy VS Code, VSC) ingyenes, nyílt forráskódú kódszerkesztő, melyet a Microsoft fejleszt Windows, Linux és macOS operációs rendszerekhez. Támogatja a hibakeresőket, valamint beépített Git támogatással rendelkezik, továbbá képes az intelligens kódkiegészítésre az IntelliSense segítségével. A Visual Studio

Code-ban a felhasználók megváltoztathatják a kinézetet (témát), megváltoztathatják a szerkesztő gyorsbillentyű-kiosztását, az alapértelmezett beállításokat és még sok egyebet. Támogatja a kiegészítőket, melyek segítségével további funkciók, testreszabási lehetőségek érhetők el. A Visual Studio Code az Electron nevű keretrendszeren (amellyel asztali környezetben futtatható Node.js alkalmazások fejleszthetők), illetve a Visual Studio Online szerkesztőn alapszik (fejlesztési neve: "Monaco"). A VSC-t bővítményekkel lehet bővíteni, ami segítségre lehet a produktivitás növelésére, tesztelés, fejlesztési folyamat felgyorsítása, és egyéni fejlesztői élmény növelése. A Live Server egy népszerű Visual Studio Code bővítmény, amely segíti a webfejlesztést azáltal, hogy egy lokális fejlesztői szervert indít, és automatikusan frissíti a böngészőt a fájlok módosításakor. A Live Share egy bővítmény, amely lehetővé teszi, hogy valós időben együtt dolgozz más fejlesztőkkel ugyanazon a kódbázison anélkül, hogy fájlokat kellene megosztanod. Az Open PHP/HTML/JS in Browser bővítmény egy egyszerű és hasznos bővítmény, amely lehetővé teszi a fejlesztők számára, hogy gyorsan és könnyedén megnyissák a PHP, HTML és JavaScript fájlokat közvetlenül a böngészőben anélkül, hogy manuálisan kellene azt megnyitni. A Path Autocomplete egy hasznos bővítmény, ami segít a fájl- és mappautvonalak gyorsabb és pontosabb kiegészítésében, a kódban. Különösen akkor hasznos, ha sok fájl és mappa található a projektben, és gyakran kell elérni különböző fájlokat a kódban.

2.1.2. HTML

A HTML (angolul: HyperText Markup Language, „hiperszöveges jelölőnyelv”) egy leíró nyelv, melyet weboldalak készítéséhez fejlesztettek ki, és mára már internetes szabvánnyá vált a W3C (World Wide Web Consortium) támogatásával. Az aktuális verzió száma 5, mely az SGML általános jelölőnyelv egy konkrét alkalmazása (azaz minden 5-ös HTML dokumentum egyben az SGML dokumentumszabványnak is meg kell, hogy feleljen). HTML általában szöveges állományokban található meg olyan számítógépeken, melyek az internethez kapcsolódnak. Ezek az állományok tartalmazzák azokat a szimbólumokat, amelyek a megjelenítő programnak leírják, hogyan is kell megjeleníteni, illetve feldolgozni az adott állomány tartalmát.

2.1.3. Angular JS

Az AngularJS egy megszűnt ingyenes és nyílt forráskódú JavaScript-alapú webes keretrendszer egyoldalas alkalmazások fejlesztésére. Főként a Google és egy magánszemélyekből és vállalatokból álló közösség tartotta fenn. Célja az volt, hogy egyszerűsítse mind az ilyen

alkalmazások fejlesztését, mind a tesztelését azáltal, hogy keretrendszert biztosít a kliensoldali modell-nézet-vezérlő (MVC) és modell-nézet-nézet-modell (MVVM) architektúrákhoz, valamint a webes alkalmazásokban és a progresszív webes alkalmazásokban általánosan használt komponensekhez.

2.1.4. GitHub

GitHub egy web alapú platform, amelyet főként verziókezelésre és együttműködő szoftverfejlesztésre használnak. Többféle funkciót biztosít a kód-tárolók kezeléséhez, mint például forráskód-kezelés és verziókezelő eszközök. A fejlesztők többnyire a GitHub-ot használják, hogy meg tudják osztani kódjukat egymással, együttműködjenek projekteken és hozzájáruljanak az open-source szoftverekhez. A GitHub 2008-ban alakult és azóta az egyik legnagyobb kódtároló platformmá vált a világon. Egyéni fejlesztők milliói használják, illetve rengeteg szervezet, köztük néhány legnagyobb tech cég is, mint a Microsoft, a Google és a Facebook.

2.1.5. JSON

A JSON egy egyszerű, ember által is olvasható szöveg alapú szabvány számítógépek közötti adateserére. A JavaScript szkriptnyelvből alakult ki egyszerű adatstruktúrák és asszociatív tömbök (a JSON-ban „objektum” a nevük) reprezentálására. A JavaScripttel való kapcsolata ellenére nyelvfüggetlen, a legtöbb nyelvhez van értelmezője. A JSON hivatalos MIME-típusa application/json. Fájlként a kiterjesztése json. A JSON-t legtöbbször egy szerver és egy kliens számítógép közti adatátvitelre használják (legtöbbször AJAX technológiával). Általánosságban strukturált adatok tárolására, továbbítására szolgál. A téradatok leírására és tárolására szolgáló változata a GeoJSON. Az XML legfőbb alternatívája.

2.1.6. CSS

A CSS-t a weboldalak szerkesztői és olvasói egyaránt használhatják, hogy beállítsák vele a lapok színét, betűtípusokat, elrendezéseket, és más megjelenéshez, esztétikához kapcsolódó elemeket. A tervezése során a legfontosabb szempont az volt, hogy elkülönítsék a dokumentumok struktúráját (melyet HTML vagy egy hasonló leíró nyelvben lehet megadni) a dokumentum megjelenésétől (melyet CSS-sel lehet megadni). Az ilyen elkülönítésnek több haszna is van, egyrészt növeli a weblapok használhatóságát, rugalmasságát és a megjelenés kezelhetőségét, másrészt csökkenti a dokumentum tartalmi struktúrájának komplexitását. A CSS ugyancsak alkalmas arra, hogy a dokumentum stílusát a megjelenítési módszer

függvényében adja meg, így elkülöníthető a dokumentum formája a képernyőn, nyomtatási lapon, hangos böngészőben (beszédszintetizátorral olvassa fel a weboldal szövegét).

2.1.7. MySQL

Az SQL (Structured Query Language) egy standard programozási nyelv, amelyet adatbázisok kezelésére használnak. Az SQL lehetővé teszi az adatok lekérdezését, manipulálását és kezelését relációs adatbázisokban. Számos adatbázis-kezelő rendszer támogatja, például MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle Database és egyém más.

2.1.8. JavaScript

A JavaScript programozási nyelv egy objektumorientált, prototípus-alapú szkriptnyelv, amelyet weboldalakon elterjedten használnak. Ebből fejlődött ki a TypeScript, ami a JavaScript típusos változatának tekinthető. A JavaScript a három alapvető webfejlesztési nyelv közé tartozik, a HTML és a CSS mellett. Míg a HTML a tartalmat, a CSS a megjelenést határozza meg, addig a JavaScript a viselkedést (interaktivitást). A JavaScript általában a böngészőkben fut, lehetővé téve az interaktív weboldalak létrehozását. Ilyen például az űrlapok validálása, dinamikus tartalomfrissítés (AJAX), animációk, és egyéb felhasználói élményt javító funkciók. Adatbázis

2.1.9. XAMPP

Az XAMPP egy ingyenes és nyílt forráskódú szoftvercsomag, amely tartalmazza az Apache web szerveret, a MySQL adatbázis-kezelő rendszert, valamint az PHP és a Perl programozási nyelveket. A neve a csomagban található komponensekre utal: "X" az operációs rendszer (Cross-platform), "A" az Apache, "M" a MySQL, "P" pedig a PHP rövidítése. Az XAMPP célja, hogy egyszerűen telepíthető és használható környezetet biztosítson a webfejlesztéshez. Általában lokális számítógépeken használják, hogy teszteljék és fejlesszék a weboldalakat, mielőtt azokat az internetre feltöltik. Az XAMPP telepítése és konfigurálása viszonylag egyszerű, sok dokumentáció és támogatás áll rendelkezésre az interneten. Az XAMPP használatával a webfejlesztők könnyen kialakíthatnak egy komplett fejlesztési környezetet a saját gépükön, amelyben tesztelhetik és finomíthatják a webalkalmazásokat. Jelen esetünkben, mi a XAMPP-ot nem lokális gépen használtuk, hanem a phpMyAdmin-t összekötöttük a szerverünkkel.

2.1.10. API

Az API az "Application Programming Interface" rövidítése. Az API egy olyan szoftver interfész, amely lehetővé teszi a különböző szoftveralkalmazások közötti kommunikációt és

adatsere lehetőségét. Az API-k általában előre meghatározott szabványokat és protokollokat használnak, amelyek lehetővé teszik az alkalmazások közötti adatátvitelt. Az API-k lehetővé teszik, hogy az alkalmazások különféle funkciói és adatai elérhetők legyenek más alkalmazások számára, hogy azok használni tudják őket. Például egy webhely, mint például a Facebook, rendelkezik egy API-val, amely lehetővé teszi a harmadik feleknek, hogy az alkalmazásukba integrálják a Facebook bejelentkezési rendszert vagy a Facebook felhasználók adatait. Az API-k használata széles körben elterjedt a szoftverfejlesztésben, és lehetővé teszi az alkalmazások közötti integrációt, ami növeli az alkalmazások funkcionalitását és használhatóságát.

2.1.11. AJAX

Az Ajax (Asynchronous JavaScript and XML) interaktív webalkalmazások létrehozására szolgáló webfejlesztési technika. Segítségével a weblap kis mennyiségű adatot cserél a szerverrel a háttérben, így a lapot nem kell újratölteni minden egyes alkalommal, amikor a felhasználó módosít valamit. Ez növeli a honlap interaktivitását, sebességét és használhatóságát.

2.1.12. PHP

A PHP (Hypertext Preprocessor) egy nyílt forráskódú, szerveroldali programozási nyelv, amelyet kifejezetten webfejlesztéshez hoztak létre. Az egyik legnépszerűbb nyelv a dinamikus weboldalak és webalkalmazások készítéséhez. A PHP kódot a webkiszolgáló (pl. Apache vagy Nginx) futtatja, mielőtt az oldal megjelenik a felhasználónál, így dinamikusan generálhatók a weboldalak tartalma. A PHP szoros integrációval rendelkezik a legnépszerűbb adatbáziskezelő rendszerekkel, mint például MySQL, MariaDB, PostgreSQL és mások.

2.2. Kódmagyarázat

A projektünkben szereplő valamennyi oldalhoz létre lett hozva külön egyedi header.php és footer.php, ezzel is segíteni az átláthatóságot.

Ezekben a fájlokban szerepelnek a HTML alapvetőbb kódrészletei, mint például a karakterkódolás, az oldal nyelve, vagy a láblécben lévő navigációs sor. Emellett itt kerülnek meghívásra az oldalhoz tartozó CSS fájlok, a külső könyvtárak (Bootstrap, FontAwesome), és a JavaScript kódállományok is.

Láblécben található menüsor hasonló a fejlécben szereplővel.

Ezek mellett az oldalak jelentős részéhez hozzá van csatolva a navbar.php, amely dinamikusan generálja a navigációs sávot. A PHP kód felelős az aktuális útvonal alapján az elérési utak

meghatározásáért, valamint a felhasználó bejelentkezési állapotának ellenőrzéséért. A HTML rész tartalmazza a navigációs sáv vizuális elemeit.

A fájl elején ellenőrzi, hogy a PHP munkamenet (session) elindult-e, és ha nem, akkor elindítja:

```
<?php
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}
?>
```

Ez biztosítja, hogy a felhasználó bejelentkezési állapotát kezelni tudja.

A fájl az aktuális URL (\$_SERVER["REQUEST_URI"]) alapján határozza meg a relatív elérési utat a szükséges fájlokhoz.

Bejelentkezés elérése:

```
<?php
$Path = "./";
if($_SERVER["REQUEST_URI"] == "/hambibambi/application/view/logreg/loginreg.php") {
    $Path = "../..../";
}
```

Ez a logika biztosítja, hogy a navigációs sávban szereplő hivatkozások mindig helyesen működjenek, függetlenül attól, hogy az alkalmazás melyik részén tartózkodik a felhasználó.

A navigációs sáv tartalmazza az alkalmazás logóját. A logó képének elérési útja dinamikusan kerül meghatározásra a \$Path változó alapján az src-ben. Ha a kép nem jelenne meg akkor egy alternatív szöveg jelenik meg.

A navigációs sáv tartalmazza az oldal különböző részeire mutató hivatkozásokat:

```
<nav>
<ul class="navbar">
  <li><a href=<?= $Path . "index.php"; ?>><img alt="Home icon" data-bbox="468 698 492 718"/> Kezdőlap</a></li>
  <li><a href=<?= $Path . "application/view/menu/menu.php"; ?>><img alt="Menu icon" data-bbox="468 728 492 748"/> Étlap</a></li>
  <li><a href=<?= $Path . "application/view/contacts/contact.php"; ?>><img alt="Contact icon" data-bbox="468 758 492 778"/> Kapcsolat</a></li>
```

A navigációs sáv dinamikusan változik a felhasználó bejelentkezési állapotától függően:

Ha a felhasználó be van jelentkezve:

```
<?php if (isset($_SESSION['user_id'])): ?>
    <li><a href=?=$Path."application/view/profile/profile.php";?><svg
xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi
bi-person-fill" viewBox="0 0 16 16">
    <path d="M3 14s-1 0-1-1 1-4 6-4 6 3 6 4-1 1-1 1zm5-6a3 3 0 1 0 0-6 3 3 0 0 0 0 6"/>
</svg>Profil</a></li>
<li>
    <div class="kosarikon">
        <p>0</p><i class="fa fa-shopping-cart"></i>
    </div>
</li>
<?php else: ?>
```

Ha a felhasználó nincs bejelentkezve:

```
<li><a href=?=$Path."application/view/logreg/loginreg.php";?><svg
xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi
bi-person-fill" viewBox="0 0 16 16">
    <path d="M3 14s-1 0-1-1 1-4 6-4 6 3 6 4-1 1-1 1zm5-6a3 3 0 1 0 0-6 3 3 0 0 0 0 6"/>
</svg>Belépés / Regisztráció</a></li>
<?php endif; ?>
```

2.2.1. Adatbázis kapcsolódás

Az adatbázis kapcsolódás php-val valósul meg. Ha a kapcsolódással hiba történt, akkor egy hibaüzenet jelenik meg.

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "hambibambi";

$conn = mysqli_connect($servername, $username, $password, $dbname);

if (!$conn) {
    http_response_code(500);
    echo json_encode([
        'status' => 'error',
        'message' => 'Adatbázis kapcsolódási hiba: ' . mysqli_connect_error()
    ], JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
    exit();
}
echo json_encode([
    'status' => 'success',
    'message' => 'Sikeres kapcsolódás az adatbázishoz.'
]);
?>
```

2.2.2. Kezdőlap

2.2.2.1. index.php

Az index.php egy PHP alapú weboldal, ami tartalmazza az étterem bemutatkozását, néhány kiemelt ajánlatot, és alapvető weboldal struktúrát. Először a weboldalra beillesszük a kellendő fájlokat.

Itt konténerekben találhatóak képek, illetve szövegek, ami az étterem bemutató szövegeit tartalmazza. Két külön div-ben vannak elhelyezve.

A szekcióban kiemelt ajánlatok jelennek meg.

```
<section class="kiemelt-ajanlatok">
  <h2>Kiemelt ajánlataink</h2>
  <div class="ajanlatok">
    <?php
      $sql = "SELECT product_name, picture FROM products ORDER BY RAND() LIMIT
3"; // 3 véletlenszerű termék
      $result = $conn->query($sql);

      if ($result->num_rows > 0):
        while ($row = $result->fetch_assoc()): ?>
          <div class="ajanlat">
            ">
            <h3><?php echo htmlspecialchars(string: $row['product_name']); ?></h3>
          </div>
        <?php endwhile;
      else: ?>
        <p>Nincsenek ajánlatok jelenleg.</p>
      <?php endif; ?>
    </div>
  </section>
```

Három véletlenszerűen kiválasztott kép alapján. Az adatbázisból lekérdez három véletlenszerű terméket, a products táblából. Minden termékhez megjeleníti a termék nevét, és hozzá megfelelő képet. Hibakezelésként üzenetet jelenít meg, ha nincsenek ajánlatok. Ez a bemutatkozás szekció, ez az étterem további információit tartalmazza.

```
<?php $conn->close(); ?>
<?php include("./application/view/footer.php"); ?>
```

A végén lezárja a kapcsolatot, ezt követően beilleszti a footer.php-t vagyis a lábléceket.

2.2.3. Rólunk

2.2.3.1. contact.php

Ez a PHP alapú weboldal az étterem elérhetőségi információit és értékelési rendszerét valósítja meg. Az oldal lehetővé teszi a bejelentkezett felhasználók számára, hogy értékeljék az éttermet, valamint megjeleníti a korábbi értékeléseket.

A "session_start()" kezeli a felhasználói munkamenetet, a connect.php pedig az adatbáziskapcsolatot létesítő file.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['rating'],  
$_POST['message'])) {  
    if (!isset($_SESSION['user_id'])) {  
        die("Hiba: Be kell jelentkezni az értékeléshez.");  
    }  
}
```

Értékelések feldolgozása, POST kérés kezelése. Ez a kódrészlet ellenőrzi, hogy a felhasználó be van jelentkezve. Megakadályozza a többszöri értékelést úgy, hogy ellenőrzi, hogy a felhasználó értékelt-e.

```
$user_id = $_SESSION['user_id'];  
$review_value = intval($_POST['rating']);  
$description = trim($_POST['message']);  
$review_date = date('Y-m-d H:i:s');
```

Ez a kódrészlet a felhasználótól érkező értékelése adatokat dolgozza fel, és készíti elő az adatbázisba történő mentésre.

```
$check_query = "SELECT * FROM reviews WHERE user_id = ?";  
$stmt_check = $conn->prepare($check_query);  
$stmt_check->bind_param("i", $user_id);  
$stmt_check->execute();  
$result_check = $stmt_check->get_result();
```

Ez a kódrészlet egy már létező értékelés ellenőrzését valósítja meg a felhasználó azonosítója alapján, hogy a felhasználót megakadályozza a többszöri értékeléstől.

```

if ($result_check->num_rows > 0) {
} else {
    $stmt = $conn->prepare("INSERT INTO reviews (user_id, review_value, description,
review_date) VALUES (?, ?, ?, ?)");
    $stmt->bind_param("iiss", $user_id, $review_value, $description, $review_date);

    if ($stmt->execute()) {
        header("Location: " . $_SERVER['PHP_SELF']);
        exit;
    } else {
        die("Hiba történt az értékelés mentésekor: " . $stmt->error);
    }
}
}

```

Ez a kódrészlet kezeli az értékelések adatbázisba történő mentését, és figyeli, hogy a felhasználó egyszer értékelhet. Ha a felhasználónak már van értékelése, akkor nem történik semmi. Sikeres mentés esetén az oldal újra töltődik, és megszakítja a kód végrehajtását. Hiba esetén, leáll a weboldal egy hibaüzenettel.

Az oldal öt fő területre oszlik, “area1”, ahol az elérhetőségek találhatóak, “area2”, ahol az étterem nyitvatartása található, “area3”, ahol az étterem közösségi média platformjaihoz link található, “area4”, ahol pedig a felhasználó az értékelési űrlapot kitöltheti, az “area5” ahol a felhasználó megtekintheti a mások által létrehozott értékeléseket.

```

$sql = "
SELECT u.full_name, r.review_value, r.description, r.review_date
FROM reviews r
JOIN Users u ON r.user_id = u.user_id
ORDER BY r.review_date DESC

```

Ez a kódrészlet adatokat kérdez le az adatbázisból, a felhasználó teljes nevét, értékelés értéke, az értékelés tartalma, illetve az értékelés létrehozásának dátumát.

```
// Ellenőrzés, hogy van-e eredmény
if ($result->num_rows > 0) {
    // Táblázat megjelenítése
    echo '<table border="1" cellpadding="10" cellspacing="0" style="width: 100%; margin-top: 20px;">';
    echo
'<tr><th>Felhasználó</th><th>Értékelés</th><th>Leírás</th><th>Dátum</th></tr>';

    // Eredmények feldolgozása és megjelenítése
    while($row = $result->fetch_assoc()) {
        echo '<tr>';
        echo '<td>' . htmlspecialchars($row['full_name']) . '</td>';
        echo '<td>' . htmlspecialchars($row['review_value']) . '/5</td>';
        echo '<td>' . htmlspecialchars($row['description']) . '</td>';
        echo '<td>' . htmlspecialchars($row['review_date']) . '</td>';
        echo '</tr>';
    }

    echo '</table>';
} else {
    // Ha nincs értékelés
    echo '<p>Nincsenek értékelések.</p>';
}
```

Ez a kód az adatmegjelenítést szolgálja, táblázatosan jelenik meg, biztonsági intézkedésként minden kiírt adat “htmlspecialchars()”-on keresztül megy, ami megakadályozza az XSS támadásokat. Ha nincs értékelés üzenet jelenik meg.

2.2.3.2. contact.js

Ez a JavaScript kód kezeli a contact.php űrlap küldés gombjának állapotát egy üzenetmező tartalma alapján.

```
document.getElementById("message").addEventListener("input", function() {  
    let sendButton = document.getElementById("sendButton");  
    if (this.value.trim() !== "") {  
        sendButton.disabled = false;  
    } else {  
        sendButton.disabled = true;  
    }  
});
```

Ez a kód figyeli az üzenetmezőt, értékeli a mező tartalmát, alkalmazza a küldés gomb állapotát, lehet tiltott, illetve aktív. Megakadályozza, hogy a felhasználó üres mezőt, illetve adatot küldjön. Azonnali visszajelzést ad a felhasználónak a gomb állapotán keresztül.

2.2.4. Belépés/Regisztráció

2.2.4.1. loginreg.php

Ez a PHP kód egy felhasználókezelési rendszert valósít meg, ami tartalmazza a regisztrációt, bejelentkezés, és település/vármegye választási lehetőséget.

```
// Ellenőrzi, hogy a felhasználó már be van-e jelentkezve  
if (isset($_SESSION['user_id'])) {  
    header("Location: ../loginreg.php");  
    exit();  
}
```

Ez a kódrészlet ellenőrzi, hogy a felhasználó már be van-e jelentkezve.

```
$sql = "SELECT counties.county_id, counties.county_name, settlements.settlement_name,  
settlements.settlement_id  
FROM counties  
LEFT JOIN settlements  
ON counties.county_id = settlements.county_id  
ORDER BY county_name, settlement_name";
```

Ez a kódrészlet két tábla összekapcsolását szolgálja, tömbé rendezi vármegye szerint.

```

$counties = [];
if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $countyName = $row['county_name'];
        $settlementName = $row['settlement_name'];

        // Ellenőrizzük, hogy a vármegye már létezik-e a tömbben
        if (!isset($counties[$countyName])) {
            $counties[$countyName] = [];
        }

        // Hozzáadjuk a települést a vármegyéhez
        if (!empty($settlementName)) { // Csak akkor adjuk hozzá, ha a település neve nem üres
            $counties[$countyName][] = $row;
        }
    }
}

```

Ez a kódrészlet egy hierarchikus adatszerkezetet épít fel, ahol vármegyékhez rendeli a hozzájuk tartozó településeket az adatbázisból lekérdezett eredmény alapján.

Üres tömb létrehozása, ami majd a vármegyéket és azok területét fogja tárolni. Ellenőrzi, hogy van-e visszaadott sor az adatbázis-lekérdezésből. While ciklussal végigmegy az összes eredmény soron.

```

$countyName = $row['county_name'];
$settlementName = $row['settlement_name'];
if (!isset($counties[$countyName])) {
    $counties[$countyName] = [];
}

```

Kinyeri az aktuális sor vármegye nevét, ha a vármegye még nem szerepel a tömbben, létrehoz egy új bejegyzést üres tömbbel.

```
if (!empty($settlementName)) {
    $counties[$countyName][] = $row; }
```

Csak akkor adjuk hozzá, ha a település neve nem üres.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['full_name'],
$_POST['email'], $_POST['password'], $_POST['phone_number'], $_POST['address'],
$_POST['settlement_id'])) {
```

POST kérés, ellenőrzése, csak űrlapküldésre reagál. Megvizsgálja, hogy az alapvető adatok léteznek-e.

```
$full_name = trim($_POST['full_name']);
$email = trim($_POST['email']);
$password = $_POST['password'];
$phone_number = trim($_POST['phone_number']);
$address = trim($_POST['address']);
$settlement_id = intval($_POST['settlement_id']);
```

Ez a kódrészlet az adatfeldolgozást valósítja meg.

```
$stmt = $conn->prepare("SELECT user_id FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$stmt->store_result();
```

Ez a kódrészlet, megakadályozza, hogy a felhasználó duplikált regisztrációt hozzon létre.

```
$stmt = $conn->prepare("INSERT INTO users (full_name, email, password,
phone_number, address, settlement_id, registration_date)
VALUES (?, ?, ?, ?, ?, ?, NOW());
$stmt->bind_param("sssssi", $full_name, $email, $hashed_password, $phone_number,
$address, $settlement_id); // A paraméterek típusa: 's' (string)
$stmt->execute();
```

Ebben a kódrészletben az adatokat mentjük az adatbázisba, dátumkezelés a “NOW()” függvényt használja, hogy a jelenlegi dátumot megkapja.

Sikeres regisztráció után átirányítás a bejelentkezéshez.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST' && isset($_POST['email'],
$_POST['password'])) {
```

Ez a kódrészlet csak a POST kérést fogadja el, és ellenőrzi, hogy az email, illetve a jelszó mező meg-e lett adva.

```
$stmt->execute();  
$stmt->store_result();  
$stmt->bind_result($user_id, $hashed_password);  
$stmt->fetch();
```

Az eredményt köti a “\$user” és “\$hashed_password” változókhoz. A további részben található egy div konténer, ezen belül egy “area1” osztályú div, és ezen belül egy felhasználóregisztrációs űrlapot valósul meg.

2.2.4.2. loginreg.js

Ennek a JavaScript kódnak az a szerepe, hogy felhasználói bejelentkezéseket kezel, adatokat küld a szerverre és választ fogadja. A függvény első lépésben kiírja az eseményobjektumot a konzolra “console.log(e)”, ami segít debugolni.

```
const options = {  
  method: 'POST',  
  headers: {  
    'Content-Type': 'application/json'  
  },  
  body: JSON.stringify(jsonData)  
};
```

Ez a kódrészlet beállítja HTTP kérés metódusát, a fejléceket, és a body-t.

```
fetch('../controller/loginController.php', options)
```

A “fetch” hívja a loginController.php file-t.

```
if(data.message != "") {  
  document.querySelector(".statusMessage").innerHTML = data.message;  
  document.querySelector(".statusMessage").classList.remove("hide");  
  document.querySelector(".statusMessage").classList.add("show");  
} else {  
  document.querySelector(".statusMessage").classList.remove("show");  
  document.querySelector(".statusMessage").classList.add("hide");  
  document.querySelector(".statusMessage").innerHTML = "";  
}
```

Ebben a kódrészletben a válasz fogadás történik, a válasz JSON formátumban érkezik, ha van üzenet, akkor megjeleníti a felhasználónak. Ha nincs, elrejtí a státuszmezőt. Hiba esetén hiba üzenetet küld a konzolra. Ha a hálózati kérés sikertelen, a hibát naplózza. Ez a kódrészlet egy telefonszám formátum-ellenőrzést valósít meg. A “preg_match” függvény ellenőrzi a telefonszám formátumát. Ha a telefonszám nem felel meg az elvárásoknak, egy “alert()” figyelmeztető ablakban üzenetet küld a felhasználónak.

2.2.5. Étlap

2.2.5.1.menu.php

A menu.php a weboldalon lévő Étlap oldal kódját tartalmazza.

A fájlban meghívásra kerül több másik fájl, pontosabban a product.php, a hagyományos navbar.php, valamint az egyedi header.php, és a footer.php amelyek segítségével létrejön az oldal.

Az insideNavbar osztályú doboz az oldalon látható belső navigációs mező megvalósítása. A “nav-link” osztály lehetőséget biztosít a hivatkozások egységes stílusának meghatározására.

Az href attribútum az oldal adott szekciójára mutat, amelyet az id attribútum határoz meg. Például a href="#hamburgers" a hamburgerek szekciójára ugrik.

Végül meghívásra kerül a product.php is, amely segítségével megjelennek a termékek az étlapon.

2.2.5.2. product.php

A product.php fájl célja, hogy különböző kategóriákba sorolt termékeket jelenítsen meg, például hamburgereket, leveseket, salátákat stb. A termékek adatai egy külső API-ból érkeznek, amelyet az AngularJS \$http szolgáltatása kezel.

A kód elején egy PHP kód található, amely ellenőrzi, hogy a felhasználó be van-e jelentkezve. Az információ a HTML data-is-logged-in attribútumán keresztül kerül átadásra az AngularJS alkalmazásnak:

```
<body ng-controller="ProductController" data-is-logged-in="<?= $isLoggedIn ? 'true' : 'false' ?>">
```

Ez lehetővé teszi, hogy az AngularJS alkalmazás a felhasználó bejelentkezési állapotától függően különböző funkciókat biztosítson.

Az oldal HTML szerkezete több szekcióra van osztva, amelyek mindegyike egy termék kategóriát képvisel. A hamburgerek szekciója példaként:

```
<div class="hamburgers con">
  <legend>Hamburgerek</legend>
  <div class="container" id="hamburgers">
    <div class="product" style="width: 18rem;" ng-repeat="product in products |
filter:{product_category_id: 3}" data-id="{{product.product_id}}">
      
      <div class="card-body product-body">
        <h5 class="card-title">{{ product.product_name }}</h5>
        <p class="card-text">{{ product.description }}</p>
        <p class="ar">{{ product.price }} Ft</p>
        <input type='number' id='quantity' name='quantity' min='1' max='9' value='1'>
        <button type='button' class="kosarhoz">Kosárba</button>
      </div>
    </div>
  </div>
</div>
```

Az AngularJS ng-repeat direktívája iterál a products tömb elemein, és csak azokat a termékeket jeleníti meg, amelyek kategóriaazonosítója (product_category_id) megegyezik a szekcióhoz tartozó értékkel.

A termékek adatai, például a név (product_name), leírás (description), ár (price) és kép (picture), az API-ból érkeznek, és AngularJS interpolációval kerülnek megjelenítésre.

Minden termékhez tartozik egy "Kosárba" gomb, amely lehetővé teszi a felhasználó számára, hogy a terméket a kosárhoz adja.

A fájl végén az AngularJS alkalmazás aktiválása található, amely a termékek betöltéséért felelős:

```
<script>
  let app = angular.module('HambiBambiApp', []);
  app.controller('ProductController', function($scope, $http) {
    $http.get('http://localhost/hambibambi/application/controller/api.php')
      .then(function(response) {
        $scope.products = response.data;
      })
      .catch(function(error) {
        console.error("Hiba történt az API hívás során:", error);
      });
  });
</script>
```

2.2.5.3. product.js

A kód elején egy button változó van definiálva, amely a "top" azonosítójú HTML elemre hivatkozik. A *window.onscroll* eseményfigyelő figyeli az oldal görgetését. Ha a felhasználó 100px-nél lejjebb görget, a gomb láthatóvá válik (*button.style.display = "block"*), ellenkező esetben elrejtí az (*button.style.display = "none"*).

A kód egy *document.addEventListener* eseményfigyelőt is tartalmaz, amely az "input" eseményre reagál. Ha az esemény célpontja egy "quantity" azonosítójú input mező, akkor ellenőrzi a bevitt értéket. Ha a felhasználó több karaktert ír be, csak az első számjegy marad meg (*input.value.slice(0, 1)*).

2.2.6. API

Az *api.php* fájl egy REST API-t valósít meg, amely a termékek kezelésére szolgál. Az API támogatja a CRUD (Create, Read, Update, Delete) műveleteket, és JSON formátumban kommunikál.

A fájl beállítja a válasz tartalomtípusát JSON-ra, és engedélyezi a CORS-t (Cross-Origin Resource Sharing), hogy más domaineokról is lehessen hozzáférni az API-hoz. Az engedélyezett HTTP-módszerek: GET, POST, PUT, DELETE, OPTIONS.

Az OPTIONS metódusra az API 200-as státuskódot küld vissza, és kilép, mivel ez a CORS előzetes ellenőrzéséhez szükséges.

A fájl a “.././connect.php” fájlon keresztül csatlakozik az adatbázishoz. A kapcsolatot a *global \$conn* változóval éri el minden függvény.

```
header("Content-Type: application/json");
require ".././connect.php";
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");
header("Access-Control-Allow-Headers: Content-Type, Authorization");
if ($_SERVER["REQUEST_METHOD"] === "OPTIONS") {
    http_response_code(200);
    exit();
}
```

A kód a `$_SERVER["REQUEST_METHOD"]` változó alapján azonosítja a beérkező HTTP-kérések típusát, és a megfelelő műveletet hajtja végre.

GET

Ha az URL-ben van id paraméter, akkor egy adott terméket kérdez le a `getProduct($id)` függvény segítségével.

Ha nincs id paraméter, akkor az összes terméket lekéri a `getProducts()` függvénnyel.

POST

A `addProduct()` függvény felelős egy új termék létrehozásáért.

A bemeneti adatokat a `php://input-on` keresztül JSON formátumban várja, és ellenőrzi, hogy minden szükséges mező meg van-e adva.

Az adatokat az `INSERT INTO SQL` utasítással menti az adatbázisba.

PUT

A `updateProduct($id, $data)` függvény frissíti egy meglévő termék adatait.

A bemeneti adatokat JSON formátumban várja, és ellenőrzi, hogy minden szükséges mező meg van-e adva. Az adatokat az `UPDATE SQL` utasítással módosítja az adatbázisban.

DELETE

A `deleteProduct($id)` függvény törli a megadott id-vel rendelkező terméket az adatbázisból.

Az adatokat JSON formátumban várja, és az id meglétét ellenőrzi.

Hibakezelés

Ha a kérés nem támogatott HTTP-módszert használ, az API egy {"error": "Invalid request method"} üzenetet küld vissza.

Sikeres lekérdezés esetén:

```
[{ "product_id": 1,  
  "product_name": "Example Product",  
  "price": 100,  
  "description": "Example description",  
  "picture": "example.jpg" }]
```

Hiba esetén:

```
{ "error": "Product ID is required" }
```

2.2.7. Kosár

2.2.7.1. cart.html

A cart.html fájl a kosár felugró ablakának alapvető szerkezetét foglalja magába.

Az oldal tartalmát a “cartBox” osztályú konténer tartalmazza. Az ablak jobb felső sarkába helyezett egy "X" ikon, amely bezárja a kosár ablakát.

A törzs egy üres táblázat, amely dinamikusan frissül a kosár tartalmával.

2.2.7.2. cart.js

A cart.js fájl tartalmazza a kosár működéséhez szükséges logikát. A kosár adatait a localStorage tárolja, így az adatok megmaradnak az oldal frissítése után is.

Ha kosár felugró ablak megjelenik, a görgetés letiltásra kerül, amint bezárjuk kosár ablakot, a görgetés visszaáll az oldalon.

A "Kosárba" gomb (.kosarhoz) kattintásakor a termék adatai bekerülnek a localStorage-ba.

Ha a felhasználó nincs bejelentkezve, egy figyelmeztető üzenet jelenik meg.

Ha a termék már szerepel a kosárban, a mennyisége növekszik. Ha nem, új elemként kerül hozzáadásra.

```
let existingItem = cartItems.find(data => data.id === item.id);  
if (existingItem) {  
  existingItem.quantity += item.quantity;  
} else {  
  cartItems.push(item); }
```

A kosárban lévő összes termék mennyiségét frissíti és megjeleníti a kosár ikonon.

```
cartItems.forEach(data => totalQuantity += data.quantity);  
cartIconCounter.innerText = totalQuantity;
```

A kosár tartalmát dinamikusan frissíti a táblázatban. Ha a kosár üres, egy üzenet jelenik meg.

Egy adott termék törölhető a kosárból az "Termék törlése" linkre kattintva.

```
window.Delete = function(itemId) {  
    cartItems = cartItems.filter(data => data.id !== itemId);  
    localStorage.setItem('cartItems', JSON.stringify(cartItems));  
    updateCartCounter();  
    updateCartBox();  
};
```

Az összes termék törölhető a "Minden termék törlése" linkre kattintva.

```
window.clearAll = function() {  
    localStorage.removeItem('cartItems');  
    updateCartCounter();  
    updateCartBox();  
};
```

A kosárban lévő termékek árainak összegét kiszámítja, és megjeleníti a táblázat alján.

```
let totalSum = cartItems.reduce((sum, data) => sum + data.price * data.quantity, 0);
```

2.2.8. Rendelés véglegesítése

2.2.8.1. checkout.php

Ellenőrzi, hogy a felhasználó be van-e jelentkezve a `$_SESSION['user_id']` segítségével. Ha nincs bejelentkezve, 401-es hibát küld vissza JSON válaszban.

A bejelentkezett felhasználói adatok az adatbázisból kérődnek le (users, settlements, counties táblákból). Az adatbázisból külön lekéri a megyék listáját.

A rendelés feldolgozása POST kérés esetén ellenőrzi a kosár tartalmát és a fizetési módot, létrehozza a rendelést az orders táblában. Emellett hozzáadja a kosár tételeit a baskets táblához.

A felhasználó adatai (név, email, cím stb.) előre kitöltve jelennek meg, de nem módosíthatók. Három fizetési mód közül választhatunk: készpénz, bankkártya, SZÉP-kártya. A választott érték a rendelés adatainak eltárolásához kerül. A kosár tartalmát dinamikusan jeleníti meg.

2.2.8.2. checkout.js

A localStorage-ból betölti a kosár elemeit. Ha a kosár üres, megjeleníti az üzenetet: "A kosár üres". Ha vannak elemek, akkor megjeleníti a termékek nevét, mennyiségét és az összesített árat.

```
const cartItems = JSON.parse(localStorage.getItem("cartItems")) || [];
cartItems.forEach((item) => {
  const itemTotal = item.price * item.quantity;
  totalSum += itemTotal;
  cartList.innerHTML += `
    <li class="cart-item">
      <div>
        <h3>${item.name}</h3>
        <small>Mennyiség: ${item.quantity}</small>
      </div>
      <span>${itemTotal} Ft</span>
    </li>`;
});
```

A kód ellenőrzi azt, hogy a kosár nem üres, a fizetési mód ki van-e választva, és azt, hogy minden termék érvényes ID-vel rendelkezik. Ha bármelyik feltétel nem teljesül, hibaüzenetet jelenít meg.

Egy AJAX kérést használtunk a rendelés feldolgozásához, amely POST kérést küld a checkout.php fájlra a fizetési mód és a kosár tartalmával.

Sikeres válasz esetén törli a kosár tartalmát a localStorage-ból, és átirányítja a felhasználót a sikeres rendelés oldalára.

Hibás válasz esetén hibaüzenetet jelenít meg.

```

const xhr = new XMLHttpRequest();
xhr.open("POST", "checkout.php", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.onload = function() {
    if (xhr.status === 200) {
        const response = JSON.parse(xhr.responseText);
        if (response.success) {
            localStorage.removeItem("cartItems");
            window.location.href = response.redirect;
        } else {
            alert("Hiba: " + response.message);
        }
    } else {
        alert("Hiba történt a rendelés feldolgozása során: " + xhr.responseText);
    }
};

```

Hibás JSON válasz vagy hálózati hiba esetén figyelmeztető ablak jelenik meg a felhasználónak.

```

xhr.onerror = function() {
    console.error("Hiba történt a kérés küldése során.");
    alert("Hiba történt a kérés küldése során.");
};

```

2.2.9. Profil

2.2.9.1. profile.php

A kód elején ellenőrzi azt, hogy a felhasználó be van-e jelentkezve. Ha nem, átirányítja a bejelentkezési oldalra. Alapméretezetten nem érheti el a profil felületet a felhasználó bejelentkezés nélkül.

A felhasználó adatai lekérésre kerülnek az adatbázisból SQL lekérdezéssel, beleértve a település és vármegye nevét.

Az adatokat alapértelmezetten „Nincs adat” értékekkel tölti fel, ha hiányoznak.

A felhasználó adatait táblázatban jeleníti meg, kijelentkezési és módosítási lehetőséggel. Ezt a \$output változóba tároljuk el.

```

$output = "
<div class=\"profile_container\">
  <div class=\"user_details\">
    <table>
      <tr><th>Teljesnév:</th><td>{$full_name}</td></tr>
      <tr><th>Telefonszám:</th><td>+{$phone_number}</td></tr>
      ....
      <tr>
        <td><a href=\"profileLogout.php\">📄 Kijelentkezés</a></td>
        <td><a href=\"profileUpdate.php\">📄 Adatok módosítása</a></td>
      </tr>
    </table>
  </div>
</div>";

```

A profiloldal HTML struktúrájában meghívásra kerül a \$output változó.

```
<?php echo $output; ?>
```

2.2.9.2. profileUpdate.php

A kód elején a profil oldalhoz hasonlóan lekéri a megyéket és településeket, majd egy tömbbe rendezi őket.

Ellenőrzi a bemeneti adatokat, frissíti az adatbázist, és kezeli az esetleges hibákat.

```

<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $update_query = "UPDATE users SET ... WHERE user_id = '$user_id'";
    mysqli_query($conn, $update_query);
}

```

Az űrlap tartalmazza a felhasználói adatok módosításához szükséges mezőket. A települések listája dinamikusan frissül a kiválasztott megye alapján.


```
function updateSettlements() {
    const county = document.getElementById('county').value;
    const settlementDropdown = document.getElementById('settlement');
    settlementDropdown.innerHTML = '<option value="">Válasszon</option>';
    if (counties[county]) {
        counties[county].forEach(settlement => {settlementDropdown.innerHTML += `<option
value="${settlement.settlement_id}">${settlement.settlement_name}</option>`;
        });
    }
}
```

Ellenőrzi a jelenlegi jelszót, és opcionálisan frissíti az új jelszót, ha az megfelel a feltételeknek. Eltérés esetén hibaüzenetet ad.

2.2.10. Admin

2.2.10.1. login.php

Az admin egy megadott felhasználónév, illetve egy jelszó segítségével tud bejelentkezni a vezérlőpult felületére. A felhasználóneve: admin.

Jelszava: \$2y\$10\$fW8WQZpg/OVrDZH9XUL6buXp5J6Rau4x8n.NJOTIizhozK YUzhfee
ami biztonsági okok miatt ilyen formátumban tekinthető meg az adatbázisban, de jelen esetben a jelszó adminPass.

2.2.10.2. login.php

Ez a php file egy admin bejelentkezési rendszert valósít meg, ami ellenőrzi az admin hitelesítő adatait. Sikeres bejelentkezés esetén átirányít majd az admin vezérlőpultra (dashboard.php).

Hibás jelszó, illetve hiányos adatok esetén hibaüzenettel jelez.

Ha az admin már bejelentkezett, akkor átirányít a vezérlőpultra.

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST['username']);
    $password = trim($_POST['password']);

    if (!empty($username) && !empty($password)) {
        $sql = "SELECT admin_id, username, password FROM admin WHERE username = ?";
        $stmt = $conn->prepare($sql);
        $stmt->bind_param("s", $username);
        $stmt->execute();
        $result = $stmt->get_result();

        if ($result->num_rows == 1) {
            $admin = $result->fetch_assoc();
            if (password_verify($password, $admin['password'])) {
                $_SESSION['admin_logged_in'] = true;
                $_SESSION['admin_id'] = $admin['admin_id']; // FONTOS: admin_id helyesen!
                $_SESSION['admin_username'] = $admin['username'];
                header("Location: ../admin/dashboard.php");
                exit;
            } else {
                $error = "Hibás jelszó!";
            }
        } else {
            $error = "Nincs ilyen felhasználó!";
        }
    } else {
        $error = "Minden mezőt ki kell tölteni!";
    }
}
```

Ez a kódrészlet ellenőrzi, hogy a felhasználó elküldte-e az űrlapot, “trim()” segítségével eltávolítja a felesleges szóközöket. Adatbázisból lekéri az admin adatait. Ez a kódrészlet megvizsgálja, hogy a bevitt jelszó meg e egyezik az adatbázison belüli hash-sel. Sikeres bejelentkezés esetén továbbít az vezérlőpultra, hiba esetén üzenetet küld, a felhasználónak.

A továbbiakban csatoljuk a HTML-hez a CSS stílus fájlokat, a törzsben létrehozuk az input mezőket, ahová az admin megadhatja a felhasználónevet, illetve a jelszót.

2.2.10.3. dashboard.php

Ez egy védett adminisztrációs portál, ahol csak hitelesített adminok kezelhetik a termékeket, és amely biztonságosan kezeli a felhasználói munkameneteket.

```
session_start();
```

Ellenőrzi, hogy az admin be van-e jelentkezve. Ha nincs bejelentkezve, átirányít a bejelentkező oldalra. Ha be van jelentkezve, betölti az admin felületet.

```
$admin_user = isset($_SESSION['user']) ? $_SESSION['user'] : "Admin";  
?>
```

Ez a kódrészlet egy feltételes értékadást valósít meg, amely ellenőrzi, hogy létezik-e “user” kulcs a session-ben. A kód további része az admin felület alapstruktúrája, Üdvözlő üzenet: Dinamikusan illeszti be a bejelentkezett admin nevét (\$admin_user), htmlspecialchars() biztosítja az XSS védelemt. Terméklista: Megjeleníti az összes terméket (product_list.php) Új termék: Űrlapot nyit meg új termék hozzáadásához (product_form.php).

```
<footer>  
    <p>Admin rendszer &copy; <?php echo date("Y"); ?></p>  
</footer>
```

Automatikus évjárat: A date("Y") kiírja az aktuális évet.

2.2.10.4. product_form.php

Ez a PHP file egy új termék felvételére szolgáló űrlapot valósít meg az admin felületen, AngularJS keretrendszer segítségével.

Ez a kódrészlet, ellenőrzi, hogy az admin be van-e jelentkezve A ProductController kezeli az űrlap működését: \$scope.product objektumba gyűjti az adatokat (ng-model direktívákkal kötve) addProduct() függvény küldi el a szerverre POST kéréssel Sikeres válasz esetén: Felugró üzenet (alert). Hiba esetén hibanaplózás és felhasználói értesítés Elküldi az űrlapot (type="submit")

Visszavisz a vezérlőpultra (dashboard.php) JavaScript redirecttel. Biztonság és Validáció
Kliensoldali validáció: AngularJS adatkötés és HTML5 required/min attribútumok
Szerveroldali kommunikáció: Adatok JSON formátumban kerülnek küldésre. Az admin kitölti az űrlapot. A "Hozzáadás" gombra kattintva az adatok elküldődnek a szerverre. A szerver válasza alapján siker/error üzenet jelenik meg. Sikeres mentés után automatikus átirányítás a terméklistára.

3. Az adatbázis célja

Ez az adatbázis egy étterem weboldalának működését támogatja, amely lehetővé teszi a felhasználók számára, hogy online rendeléseket adjanak le, termékeket böngésszenek, kosárba tegyék azokat, értékeléseket írjanak, és fizetési módot választhatnak ki a rendelés elvégzéséhez.

Ez az adatbázis lehetővé teszi az étterem számára, hogy hatékonyan kezelje az online rendeléseket, nyomon kövesse a rendelési folyamatokat, és egyszerűsítse a fizetési rendszert. Az értékelések és visszajelzések segíthetnek a szolgáltatás fejlesztésében, míg az adminisztrációs funkciók biztosítják a rendszer karbantarthatóságát.

3.1. Tervezési lépések

Kitaláltuk a témánkat, egy **foodorához** hasonló weboldal, ahol lehet ételeket rendelni, leginkább hamburgereket, de lehet egyéb más is (italok, menük, tálak). A **Farm Burger** nevű Váci üzletet tekintettük példaként, nekik van egy weboldaluk, amiről rendelni is lehetséges. Ezt használtuk fel példaként az adatbázisban, illetve majd a weboldalon. Kiterveltük, hogy milyen táblákat fogunk létrehozni. Először a táblákat határoztuk meg, miközben folyamatosan alakítottuk a struktúrát, mivel gyakran előfordult, hogy egy tábla egy másikra is hivatkozást igényelt. Ezt követően a kapcsolatokat oldottuk meg, mielőtt feltöltöttük volna a táblákat értékekkel. Elmélkedések során arra jutottunk, hogy a "Basket" tábla lehetséges, hogy feleslegesen van benne az adatbázisban, mert az "Orders" ugyan ezt a szerepet játszhatja. Rá eszméltünk, hogy a felhasználók és az ételek, italok közé szükséges egy kapcsolótábla, hiszen így N:M kapcsolat lép fel. A rendelés státuszát rögzíteni sem lenne rossz ötlet, későbbi fejlesztésekben véghez visszük. Az „Order_Details” tábla törlésével az „Orders” és „Products” táblák közvetlen összekapcsolására lesz szükség. A rendelések dátuma egy lekérdezés eredménye lesz. Kategóriákon belül felmerült, hogy alkategóriák is tartoznak, tehát az ételen belül van pl.:(leves, desszert stb.). Megoldásul, egy hierarchiát lehetne felépíteni. Kitaláltuk, hogy vizuálisan kategorizáljuk a menüt, és így egyszerűbben, sikerebben tudjuk majd ezt importálni az adatbázisunkba. Admin személy, ő egy új terméket tud hozzá adni az adatbázishoz értékekkel, illetve törölhet, módosíthat. Létrehoztunk egy "Order_Statuses" táblát, ahol a rendelés státuszát lehet megsejteni. A kuponoknál felmerült egy kérdés, hogy a termékre, vagy a rendelésre vonatkozhat-e a kupon, illetve mindkettőre, de eldöntöttük, hogy csak a rendelésre vonatkozik a kupon. Végző döntés az lett, hogy nem kezelünk kuponokat, ez tovább fejlesztés. A Kategóriákhoz létrehoztunk egy Group (csoport) táblát, amely az ételek és italok tárolására szolgál, és egy N:M kapcsolat felbontását segíti elő. A

felhasználó táblát összekapcsoltuk egy újonnan létrehozott táblával a “Settlements” (települések) táblával, ahol el tároljuk a település nevét, illetve az irányítószámot. Ezt a települések táblát további “Counties” táblával kapcsoljuk össze, ami a megye tárolására szolgál. Továbbá a felhasználóhoz tartozó táblát hoztunk létre, “Reviews” (értékelés) néven, ami a lehetővé teszi, hogy a felhasználó értékeléseket írjon a weboldalon. Ezt követően a “Products” táblához hozzá csatoltunk egy “Quantity_Units” táblát, amiben eltároljuk, hogy például a terméket literben, illetve adagban számoljuk. “Payments” táblában a fizetési lehetőségeket kezeljük, hogy a felhasználó rendelése után hogyan kívánkozik fizetni.

3.2. Egyedek meghatározása

3.2.1. Users (Felhasználók)

A felhasználók bejelentkezésekor, vagy regisztrálásakor megadott adatainak rögzítését kezeli. Ide tartozik a felhasználó teljes neve, e-mail címe, jelszója, telefonszáma, és a regisztrációjának dátuma.

3.2.2. Settlements (Települések)

Egy felhasználó megadott lakcíméhez tartozó település nevét, valamint irányítószámát tárolja

3.2.3. Counties (Megyék)

A felhasználóhoz tartozó megye nevét tárolja.

3.2.4. Baskets (Kosarak)

A felhasználó által adott kosárba betett termékeket és azok mennyiségét tárolja.

3.2.5. Payments (Fizetési módok)

A fizetési módokat tárolja.

3.2.6. Quantity_units(Mértékegységek)

A termékek mértékegységét tárolja.

3.2.7. Order_Statutes (Rendelés Státusza)

A rendelés státuszát tárolja, ide tartozik az az állapot, ahol éppen a csomag tart.

3.2.8. Orders (Rendelések)

A kosarak tartalmát tárolja, amibe bele tartoznak az alábbiak: a rendelés felvételének ideje, a kiszállítás ideje.

3.2.9. Products (Termékek)

Az oldalon látható és megrendelhető termékek nevét, magyar forintban mért értékét, hozzávalóinak részletes leírását, valamint a hozzá tartozó kép nevét jpg/png formátumban.

3.2.10. Products_Categories (Termékek kategóriái)

A termékekhez kapcsolódó kategóriák nevét tárolja.

3.2.11. Products_Groups (Termékek csoportjai)

A termékek kategóriáihoz tartozó csoportok nevét tartalmazza.

3.2.12. Admin

Az oldalt szerkesztő alkalmazott profiljának a felhasználónevét, és jelszavát tárolja.

3.2.13. Reviews (Értékelés)

A felhasználó által megosztott értékeléseket tárolja, ami tartalmaz leírást, számszerű értékelést és az értékelés időpontját.

3.3. Kapcsolatok meghatározása

3.3.1. 1:N kapcsolatok meghatározása

- A “Products ”1:N-es kapcsolatban áll a “Baskets” táblával, az “Baskets” tábla egy egyed előfordulásához a “Products” tábla egy egyed előfordulása tartozik, és a “Products” tábla egy sorához az “Baskets” tábla több sora tartozhat.
- A “Orders ”1:N-es kapcsolatban áll a “Baskets” táblával, az “Baskets” tábla egy egyed előfordulásához a “Orders” tábla egy egyed előfordulása tartozik, és a “Orders” tábla egy sorához az “Baskets” tábla több sora tartozhat.
- Az “Order_Statues ”1:N-es kapcsolatban áll a “Orders” táblával, a “Orders” tábla egy egyed előfordulásához az “Order_Statues” tábla egy egyed előfordulása tartozik, és a “Order_Statues” tábla egy sorához a “Orders” tábla több sora tartozhat.
- A “Quantity_units” 1:N-es kapcsolatban áll a “Orders” táblával, a “Orders” tábla egy egyed előfordulásához a “Quantity_units” tábla egy egyed előfordulása tartozik, és a “Quantity_units” tábla egy sorához a “Orders” tábla több sora tartozhat.
- A “Payments” 1:N-es kapcsolatban áll a “Orders” táblával, a “Orders” tábla egy egyed előfordulásához a “Payments” tábla egy egyed előfordulása tartozik, és a “Payments” tábla egy sorához a “Orders” tábla több sora tartozhat.
- A “Product_Categories” 1:N-es kapcsolatban áll a “Products” táblával, a “Products” tábla egy egyed előfordulásához a “Product_Categories” tábla egy egyed előfordulása tartozik, és a “Product_Categories” tábla egy sorához a “Products” tábla több sora tartozhat.
- A “Product_Groups” 1:N-es kapcsolatban áll a “Product_Categories” táblával, a “Product_Categories” tábla egy egyed előfordulásához a “Product_Groups” tábla egy

egyed előfordulása tartozik, és a “Product_Groups” tábla egy sorához a “Product_Categories” tábla több sora tartozhat.

- Az “Users” 1:N-es kapcsolatban áll a “Baskets” táblával, a “Baskets” tábla egy egyed előfordulásához az “Users” tábla egy egyed előfordulása tartozik, és az “Users” tábla egy sorához a “Baskets” tábla több sora tartozhat.
- Az “Users” 1:N-es kapcsolatban áll a “Reviews” táblával, a “Reviews” tábla egy egyed előfordulásához az “Users” tábla egy egyed előfordulása tartozik, és az “Users” tábla egy sorához a “Reviews” tábla több sora tartozhat.
- A “Counties” 1:N-es kapcsolatban áll a “Settlements” táblával, a “Settlements” tábla egy egyed előfordulásához a “Counties” tábla egy egyed előfordulása tartozik, és a “Counties” tábla egy sorához a “Settlements” tábla több sora tartozhat.
- A “Settlements” 1:N-es kapcsolatban áll az “Users” táblával, az “Users” tábla egy egyed előfordulásához a “Settlements” tábla egy egyed előfordulása tartozik, és a “Settlements” tábla egy sorához az “Users” tábla több sora tartozhat.

3.3.2 N:M kapcsolatok felbontása

- Mivel a `Products` N:M-es kapcsolatban áll az `Baskets` táblával, ezért felbontjuk kettő darab 1:N-es kapcsolatra, így létrejön az `Orders` tábla
 - o és a `Products` 1:N-es kapcsolatban áll az `Orders` táblával, a `Orders` tábla egy egyed előfordulásához az `Products` tábla egy egyed-előfordulása tartozik, és a “Products” tábla egy sorához az “Orders” tábla több sora tartozhat,
 - o és a `Baskets` 1:N-es kapcsolatban áll az `Orders` táblával, a `Orders` tábla egy egyed előfordulásához az `Baskets` tábla egy egyed-előfordulása tartozik, és a “Baskets” tábla egy sorához az “Orders” tábla több sora tartozhat.

3.4 Táblák

3.4.1. Users (Felhasználók)

user_id	settlement_id	full_name	email	password	phone_number	registration_date	address
2	18	TesztFelhasznalo2cserelt	test2@gmail.hu	\$2y\$10\$He0J4OCZ92UWQ/VhSWVNo.ZmwlwengkJTCEjtbEpnk...	36205644333	2025-04-01	Teszt utca 32.
3	3	Kertész Ruben	rubenkertesz@gmail.com	\$2y\$10\$xtZVwzdDoWdt.KRvKW/QTumAmQnp32bkXcyTZzp5jOp...	36205462311	2025-04-10	Dózsa György út 14.

mező neve	mező típusa	mező leírása	megjegyzés	Minta adatok
user_id	int	Felhasználó azonosítója	Elsődleges kulcs	1
settlement_id	int(11)	Település idegen kulcsa		1
address_id	int	Település idegen kulcsa		1
full_name	varchar(55)	Felhasználó teljes neve		Minta Péter
email	varchar(50)	Felhasználó email címe		mintapeter@gmail.com
password	varchar(30)	Felhasználó jelszója		erosJelszo
phone_number	varchar(15)	Felhasználó telefonszáma		36304326168
registration_date	date	Felhasználó regisztrációjának dátuma		2025.03.02.
address	varchar	Lakcím megnevezése		Rákóczi fejedelem út 25.

3.4.2. Settlements (Települések)

settlement_id	county_id	settlement_name	postcode
1	1	Alsópetény	16425
2	1	Alsótold	7621
3	1	Balassagyarmat	13657

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
settlement_id	int(11)	Település azonosítója	Elsődleges kulcs	1
county_id	int(11)	Megye idegen kulcsa		1
settlement_name	text	Település neve		Balassagyarmat
postcode	int(4)	Település irányítószáma		2660

3.4.3. Counties (Megyék)

county_id	county_name
1	Nógrád
2	Heves
3	Pest
4	Komárom-Esztergom

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
county_id	int(11)	Megye azonosítója	Elsődleges kulcs	1
county_name	text	Megye neve		Nógrád

3.4.4. Orders (Rendelések)

order_id	user_id	payment_id	order_status_id	order_date	delivery_date
3	2	3	1	2025-04-07 14:05:23	2025-04-10
13	2	3	1	2025-04-08 08:44:53	2025-04-08
14	2	1	1	2025-04-09 13:09:05	2025-04-09

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
order_id	int(11)	Rendelés azonosítója	Elsődleges kulcs	1
user_id	int(11)	Felhasználók idegen kulcsa		1
payment_id	int(11)	Fizetési módok idegen kulcsa		1
order_status_id	int(11)	Megrendelés státuszok idegen kulcsa		1
order_date	datetime	Rendelés dátuma és ideje		2025.03.05. 12:32
delivery_date	date	Kiszállítás dátuma		2025.03.05.

3.4.5. Baskets (Kosarak)

basket_id	product_id	order_id	quantity
5	47	13	1
6	46	13	1
7	48	13	1

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
basket_id	int(11)	Kosarak azonosítója	Elsődleges kulcs	1
product_id	int(11)	Termékek idegen kulcsa		1
order_id	int(11)	Rendelések idegen kulcsa		1
quantity	int(11)	Termék darabszáma		3

3.4.6. Payments (Fizetési módok)

payment_id	payment_method
1	kézpénz
2	bankkártya
3	SZÉP-kártya

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
payment_id	int(11)	Fizetési mód azonosítója	Elsődleges kulcs	1
payment	varchar	Fizetési mód megnevezése		bankkártya

3.4.7. Quantity_units(Mértékegységek)

quantity_unit_id	quantity_unit_value
1 db	
2 adag	
3 l	

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
quantity_unit_id	int(11)	Mértékegység azonosítója	Elsődleges kulcs	1
quantity_unit_value	varchar	Mértékegység értéke		l

3.4.8. Order_Statuses (Rendelés Státusza)

order_status_id	status
1 felvéve	
2 készítés alatt	
3 szállítás alatt	
4 kiszállítva	

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
order_status_id	int(11)	Rendelés státuszának azonosítója	Elsődleges kulcs	1
status	varchar	Rendelés státusza		szállítás alatt

3.4.9. Products (Termékek)

product_id	product_category_id	quantity_unit_id	product_name	price	description	picture
39	1	2	Cézár saláta	2500	Grillezett csirkemell, római saláta, parmezán, kru...	cezar_salata.jpg
40	1	2	Görög saláta	2300	Paradicsom, uborka, feta sajt, olívbogyó, lilahag...	gorog_salata.jpg
41	1	2	Tonhal saláta	2600	Tonhal, saláta, kukorica, lilahagyma, paradicsom, ...	tonhal_salata.jpg

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
product_id	int(11)	Termékek azonosítója	Elsődleges kulcs	1
product_category_id	int(11)	Termékek kategóriájának idegen kulcsa		1
quantity_unit_id	int(11)	Mértékegységek idegen kulcsa		1
product_name	varchar(255)	Termék neve		Dupla sajtburger
price	int(11)	Termék ára, Forintban		2500
description	varchar	Termék leírása	szabad szöveges mező mely tartalmazza a nyersanyagokat, és/vagy további feldolgozott összetevőket	Házi buci, uborka, sajt, marhahús
picture	varchar(255)	Kép a termékről		hamburger.jpg

3.4.10. Products_Categories (Termékek kategóriái)

product_category_id	product_group_id	product_category_name
1	1	Saláták
2	1	Levesek
3	2	Hamburgerek

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
product_category_id	int(11)	Termékek kategóriájának azonosítója	Elsődleges kulcs	1
product_group_id	int(11)	Termékek csoportjainak idegen kulcsa		1
product_category_name	varchar	Termékek kategóriáinak neve		Hamburgerek

3.4.11. Products_Groups (Termékek csoportjai)

product_group_id	product_group_name
1	Előételek
2	Főételek
3	Desszertek
4	Italok

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
product_group_id	int(11)	Termékek csoportjainak azonosítója	Elsődleges kulcs	1
product_group_name	varchar	Termékek kategóriáinak neve		Előétel

3.4.12. Reviews(Értékelések)

review_id	user_id	review_value	description	review_date
1	2	3	Finom ételek!	2025-04-14 09:41:38
2	3	5	Gyors kiszállítás, kiváló ételek!	2025-04-14 09:43:18

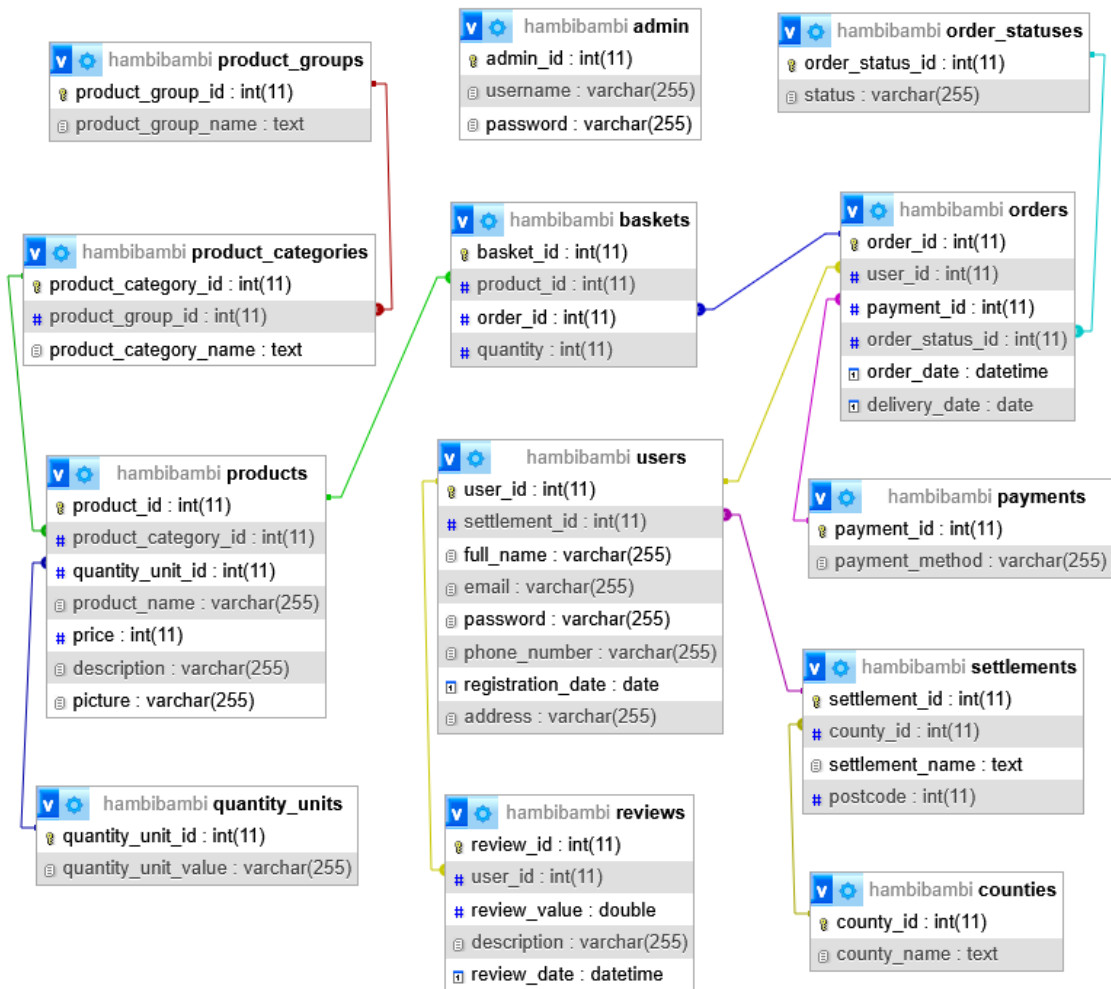
Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
review_id	int(11)	Értékelés azonosítója	Elsődleges kulcs	1
user_id	int(11)	Felhasználó idegen kulcsa		1
product_id	int(11)	Termék idegen kulcsa		1
review_value	double	Ertékelés értéke		4.5
description	varchar	Értékelés leírása		Finom volt!
review_date	datetime	Értékelés időpontja		2025.03.03.

3.4.13. Admin

admin_id	username	password
1	admin	\$2y\$10\$ORMO8mzEK/YTAYIWuKS82efWHKq5gfHzNgUeHA8tpWX...

Mező neve	Mező típusa	Mező leírása	megjegyzés	Minta adatok
admin_id	int(11)	Admin azonosítója	Elsődleges kulcs	1
username	int(11)	Admin felhasználóneve		TestAdmin
password	varchar	Admin jelszava		AdminPass

3.5. Adatbázis diagram





3.6. Adatbázis továbbfejlesztési lehetőségek

Egy új 'Coupons' tábla, amely tárolja egy kupon értékét, valamint azt, hogy mikor jár le a felhasználási ideje. Rendelésenként a felhasználó szerezhethet pontokat, amiket kuponra betudna váltani.

A termékekhez tartozó összetevők, nyersanyagok tárolása és a termékekhez való rendelése. A felhasználó képes legyen egy rendelt termékből törölni, például a hamburgerből az uborkát, vagy esetleg még hozzá ad valamit, mondjuk extra sajtot.

Családok tárolása, melyik felhasználó melyik családhoz tartozik.

4. API tesztelése

Az ECHOAPI for VS Code egy hatékony és könnyen használható VS Code kiterjesztés, amely lehetővé teszi az API-k tesztelését közvetlenül a fejlesztőkörnyezetben. Az eszköz támogatja a

GET, POST, PUT, DELETE és más HTTP metódusokat, valamint lehetőséget biztosít JSON, form-data, URL-encoded vagy raw adatok küldésére.

A kiterjesztés különösen előnyös azok számára, akik nem szeretnének külön alkalmazást, például Postmant vagy Insomniát használni, hanem egy könnyű és beépített megoldásra van szükségük. Az ECHOAPI vizuális visszajelzést ad a kérések eredményéről, lehetőséget biztosít minták (samples) mentésére, és támogatja a különböző környezetek (environment) kezelését, így a fejlesztők könnyedén válhatnak dev, staging és production API végpontok között.

Az ECHOAPI a VS Code beépített funkcióival zökkenőmentesen működik, kevesebb erőforrást használ, mint a Postman, és gyors megoldást nyújt API végpontok ellenőrzésére és hibakeresésére.

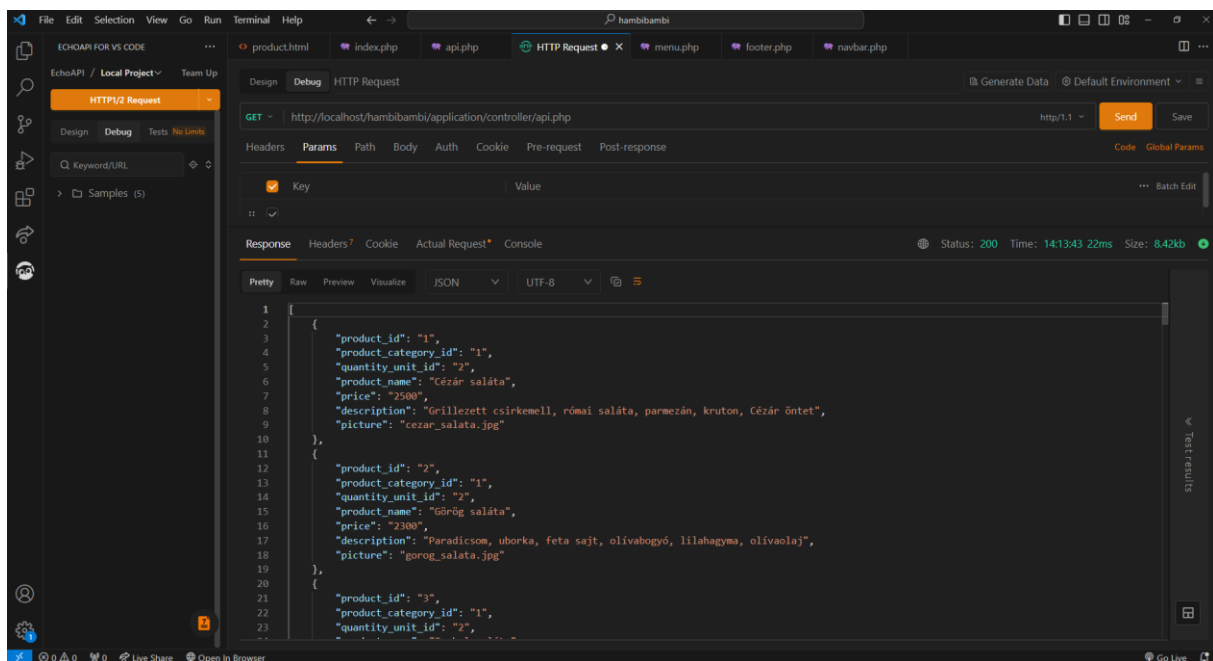
4.1. GET kérések tesztelése

Az ECHOAPI for VS Code sikeresen teljesítette a GET kérés tesztelését, amely megerősíti, hogy az API megfelelően működik és a várt válaszokat adja vissza. A kiterjesztés lehetővé tette az egyszerű és gyors tesztelést közvetlenül a VS Code környezetben, így nincs szükség külső API tesztelő eszközre.

A GET kérés során a rendszer visszaadta a kívánt adatokat, a válasz státusza 200 OK, ami azt jelenti, hogy az API megfelelően kommunikál és feldolgozza a kéréseket. Az eszköz vizuálisan is jól strukturált módon jeleníti meg az eredményeket, így a fejlesztők könnyedén ellenőrizhetik az API válaszait.

A képernyőképen jól látható a kérés URL-je, a válasz JSON formátumban, valamint az időbélyeg, amely mutatja a válaszidőt. Ez egyértelmű visszajelzés arról, hogy az API stabil és gyors.

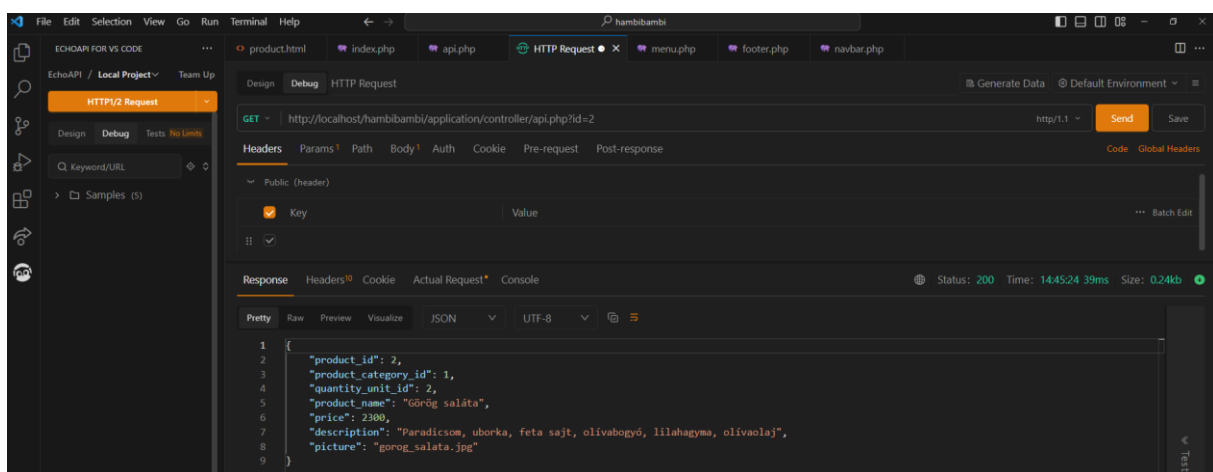
A kép az alábbiakban megtekinthető:



4.2. GET kérés tesztelése id alapján

A GET kérés tesztelése sikeres volt, mivel az API megfelelően reagált az id alapján történő lekérdezésre. A 2-es azonosítójú terméket kértem le, és a válasz tartalmazta annak minden releváns adatát, beleértve a kategóriáját, nevét, árát, leírását és a hozzá tartozó képfájlt. Ez azt jelenti, hogy az adatbázis-lekérdezés pontosan működik, és az API képes megfelelően kiszolgálni az egyedi termékekre vonatkozó lekérdezéseket.

Az API válasza JSON formátumban érkezett, ami biztosítja az adatok könnyű feldolgozását és továbbítását más rendszerek számára. A teszt alapján megállapítható, hogy a GET kérés funkcionáltsága megfelelő, az API hatékonyan és pontosan adja vissza a kért adatokat. Ezzel garantált, hogy a rendszer a kliensoldali alkalmazásokkal is zökkenőmentesen együttműködik.



4.3. POST kérések tesztelése

Az ECHOAPI for VS Code segítségével sikeresen végrehajtottuk a POST tesztelést, amely igazolja, hogy az API megfelelően kezeli az adatok beszúrását az adatbázisba.

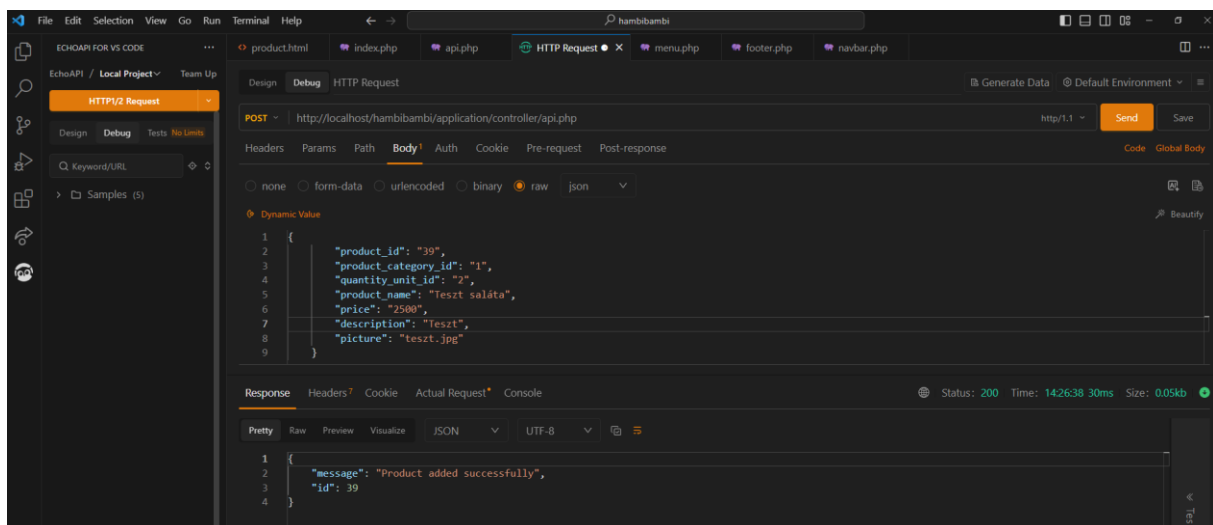
A POST kérés elküldése után az API egy JSON választ adott vissza, amely tartalmazta a következő üzenetet:

```
{ "message": "Product added successfully", "id": 39 }
```

Ez azt jelenti, hogy a termék sikeresen bekerült az adatbázisba, és az új rekord azonosítója 39 lett. A szerver megfelelően feldolgozta a beérkező adatokat, és azokat a megfelelő táblákba illesztette be.

A sikeres válasz 200-as státuszkóddal érkezett, amely azt jelzi, hogy az API helyesen működik, és minden szükséges adatot fogadott és feldolgozott.

Ez a teszt megerősíti, hogy az API megbízhatóan végzi az új adatok rögzítését, és a fejlesztők biztosak lehetnek benne, hogy a POST endpoint működése megfelelő. A részletek az alábbi képernyőképen láthatók.



4.4. PUT kérések tesztelése

A tesztelése során kiderült, hogy a PUT kérés nem működik, mivel megfeletkeztünk a CRUD metódusok engedélyezéséről.

A CORS (Cross-Origin Resource Sharing) és a megfelelő HTTP metódusok engedélyezése szükséges az API megfelelő működéséhez, különösen a PUT, DELETE és más nem-

alapértelmezett metódusok esetében. Ha az API jelenleg nem kezeli megfelelően a PUT és DELETE kéréseket, akkor szükséges kiegészíteni a megfelelő HTTP-header beállításokat.

```
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");
header("Access-Control-Allow-Headers: Content-Type, Authorization");
if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
    http_response_code(200);
    exit();
}
```

Miért szükségesek ezek a fejlécek?

Access-Control-Allow-Origin: *

- Engedélyezi a más domainről érkező kéréseket. Ha csak egy adott domaint szeretnél engedélyezni, akkor pl. így módosíthatod:

```
header("Access-Control-Allow-Origin: http://localhost");
```

Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS

- Felsorolja azokat az engedélyezett HTTP metódusokat, amelyeket az API kezelni tud.
- A PUT és DELETE kérés sokszor le van tiltva szerveroldalon, ezért kell ezt explicit engedélyezni.

Access-Control-Allow-Headers: Content-Type, Authorization

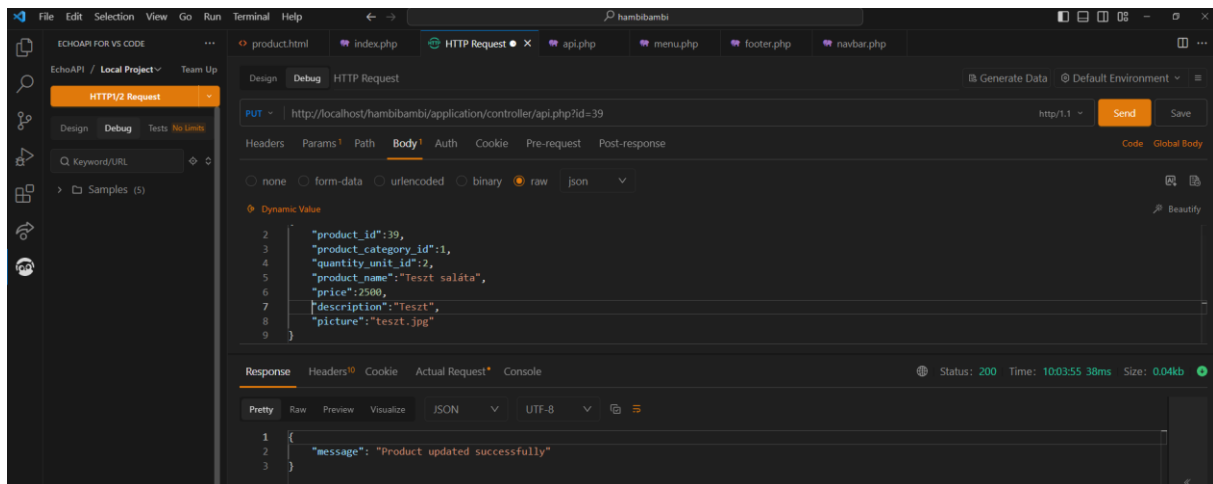
- Az API fogadhat Content-Type (JSON, form-data, stb.) és Authorization (tokenek, API kulcsok) fejléceket.

OPTIONS válasz kezelése

- A böngészők bizonyos esetekben előzetes OPTIONS kérést küldenek (preflight request), hogy ellenőrizzék a szerver támogatását.
- Ha ezt az OPTIONS kérést nem kezeli a szerver, akkor a PUT és DELETE kérések nem fognak működni.

Tesztelés utáni visszakapott érték

```
{ "product_id": "39",
  "product_name": "Frissített Saláta",
  "price": "2600",
  "description": "Salátaöntettel és extra zöldségekkel" }
```



4.4. DELETE kérések tesztelése

Az ECHOAPI for VS Code segítségével sikeresen végrehajtottuk a DELETE tesztelést, amely igazolja, hogy az API megfelelően kezeli az adatok törlését az adatbázisból.

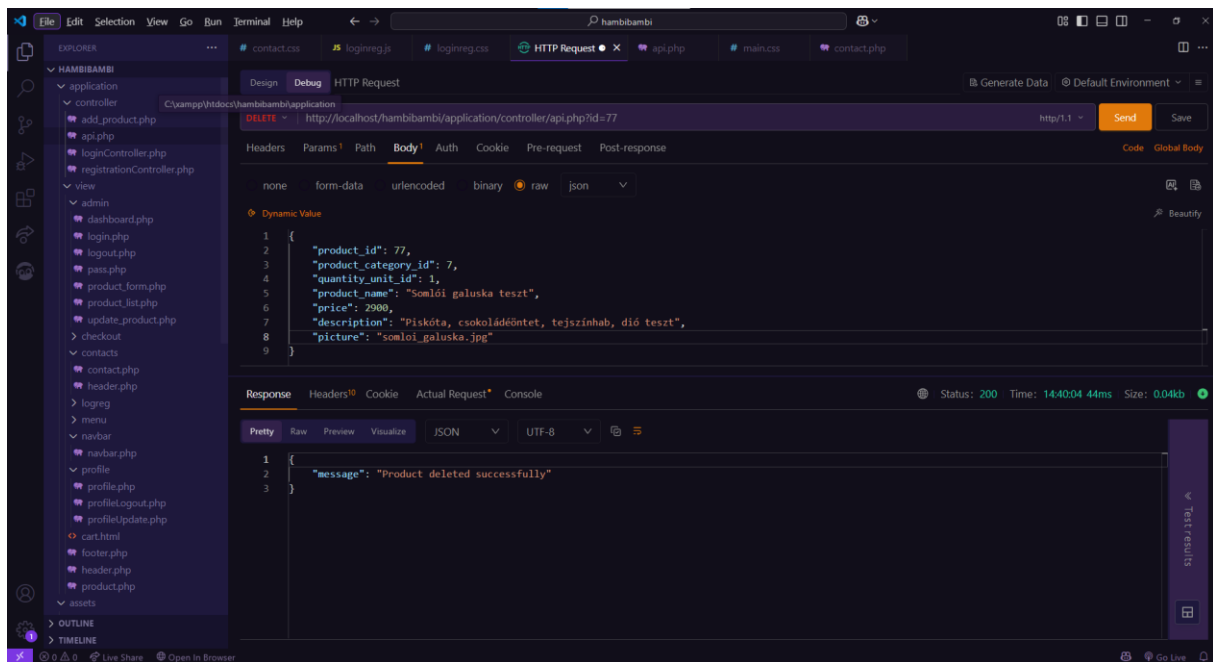
A DELETE kérés elküldése után az API egy JSON választ adott vissza, amely tartalmazta a következő üzenetet:

```
{ "message": "Product deleted successfully" }
```

Ez azt jelenti, hogy a termék sikeresen törlődött az adatbázisból. A szerver megfelelően feldolgozta a beérkező adatokat, és azokat kivette a megfelelő táblákból.

A sikeres válasz 200-as státusz-kóddal érkezett, amely azt jelzi, hogy az API helyesen működik, és minden szükséges adatot fogadott és feldolgozott, majd törölt.

Ez a teszt megerősíti, hogy az API megbízhatóan végzi az id alapján kiválasztott adatok törlését, és a fejlesztők biztosak lehetnek benne, hogy a DELETE endpoint működése megfelelő. A részletek az alábbi képernyőképen láthatók.



5. Összefoglalás

Összeségében a projektünk megkezdésekor eltervezett feladatok jelentős részét létre tudtuk hozni. Amelyek nem készültek el, azokat továbbfejlesztésnek írtuk be.

A fejlesztés közben a kéttagú csapatunk folyamatos együttműködése és kapcsolattartása folyamán sikerült párhuzamosan haladni a munkálatokkal.

5.1. Továbbfejlesztési lehetőségek

5.1.1. Nyelv választása:

A fejlécben az étterem logója mellett a kiválasztott nyelv zászlója jelenjen meg. Ezzel együtt a weboldal nyelvezete is megváltozna az adott nyelvre. Alapmértékben magyar az oldal, viszont több másik nyelvből is lehessen választani, mint például angol, német, vagy esetleg spanyol.

5.1.2. Sötét mód:

Szintén az fejlécben a logó mellett egy slider segítségével lehessen változtatni az oldal témáján. Ha sötét módba vált, akkor az oldal témája sötétbe váltson. Ezzel a funkcióval az oldal szemkímélőbbé válik, így a weboldal használata élvezhetőbbé válik.

5.1.3. Keresés funkció:

Szintén az fejlécben a logó mellett egy kereső funkció létrehozása, amellyel rá lehet keresni bizonyos ételre vagy italra.

5.1.4. Adatbázis szerveren történő működtetése:

Az adatbázis szerveren történő működtetése, annak érdekében, hogy a weboldal bárholnan elérhető legyen forrásállományok és alkalmazások telepítése nélkül. Ennek segítségével rendelés leadása után létrejövő rendelés státusza kezelhető lesz.

5.1.5. Adatok szerkesztése a rendelés véglegesítésénél:

A “Rendelés véglegesítése” oldalon szerkeszthetővé váljon a felhasználó adatai. Ennek segítségével rugalmasabb a weboldal, és nem kell visszalépni a profil adatok szerkesztéséhez.

5.1.6. Bug-ok javítása, optimalizálás, karbantartás:

További tesztelések elvégzése, jelenleg ismert, és a jövőben tudomásunkra jutó hibák javítása, egyéb optimalizációk elvégzése, folyamatos karbantartás.

5.1.7. Rendelés visszaigazolás, email küldése:

A rendelés leadásával a felhasználó email címére egy levelet küldjön, amely tartalmazza a visszaigazolást a rendelésről, és a várható kiszállítási időt.

5.1.8. Felhasználó adatainak erősebb védelme

A felhasználó által megadott adatok megvédése szempontjából erősebb védelem kialakítása.

5.1.9. Előző rendelések listája:

A profil felületen láthatóak legyenek a felhasználó eddig feladott rendelései.

5.2. Felmerült akadályok

Az adatbázis importálásánál akadály lépett fel, hogy az “Users” tábla már létezik, ezt úgy oldottuk meg, hogy DROP TABLE IS EXIST függvénnyel megoldottuk. Egy megyéhez több település, egy településhez egy megye ezt a kapcsolatot a tervező nézetben módosítani kellett. Fordítva lett bekötve a “Products” és a “Product_categories” tábla.

A termékeknek az alapértelmezett egységét be kellett állítani, például az téstáknál adag legyen. A “Basket” táblából a Designation mezőt ki kellett törölni. A “Basket” táblából törölni kellett “net_unit_price” mezőt, hiszen azt ki tudtuk számítani így nem volt fontos mezőt csinálni neki. A “Payments” és az “Orders” közötti kapcsolatot, az “Order_Statues” és az “Orders” kapcsolatot, valamint a “Coupon” és az “Order” kapcsolatot is meg kellett fordítani.

Az „Orders” és a „Product” táblákat a „Baskets” tábla köti össze, így a „Baskets” és az „Orders” 1:N-es, valamint a „Baskets” és a „Products” 1:N-es kapcsolat jön létre.

Az „Users” és az „Baskets” közötti kapcsolat nem kellett.

A rendelés véglegesítésekor a kosár adatait nem tárolta el egy kódolítás miatt, ami végül ki lett javítva.

5.3. Források:

1. Bootstrap: <https://getbootstrap.com>
2. Stack Overflow: <https://stackoverflow.com>
3. Uiverse: <https://uiverse.io>
4. Font Awesome: <https://fontawesome.com/>
5. Microsoft Learn: <https://learn.microsoft.com/>
6. W3SCHOOLS: <https://www.w3schools.com>
7. Gencraft: <https://gencraft.com>

5.4 Mellékletek

5.4.1. Forrásállományok

A teljes vizsgaremek publikus github linkje:

<https://github.com/KertACEnebuR/Hambibambi-vegleges>

5.4.2. Adatbázis

Az adatbázis tartozó sql file linkje:

<https://github.com/KertACEnebuR/Hambibambi-vegleges/blob/main/hambibambi/databases/hambibambi.sql>

Az üres, adatokat nem tartalmazó sql file:

https://github.com/KertACEnebuR/Hambibambi-vegleges/blob/main/hambibambi/databases/Hambibambi_%C3%BCres.sql

5.4.3. Xampp

Az adatbázissal való kapcsolathoz szükséges a xampp alkalmazás, mely az alábbi linkről tölthető le.

<https://www.apachefriends.org/download.html>

5.4.4. Dokumentáció

A dokumentációt az alábbi linkeken találja:

Dokumentáció pptx formátumban:

<https://github.com/KertACEnebuR/Hambibambi-vegleges/blob/main/hambibambi/documents/HambiBambi%20prezent%C3%A1ci%C3%B3.pptx>

Dokumentáció WORD-ben:

<https://github.com/KertACEnebuR/Hambibambi-vegleges/blob/main/hambibambi/documents/HambiBambi.docx>