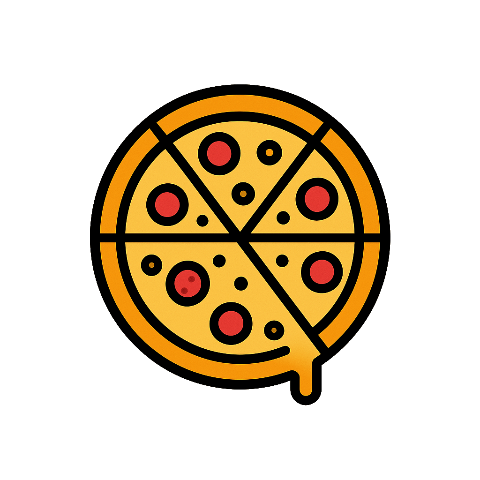
Hódmezővásárhelyi SZC Szentesi Pollák Antal Technikum

**Vizsgaremek - PollakPizza**



Készítette: Huszár Imre, Héja Gábor, Sztojka Milán

Tartalomjegyzék

[Bevezetés 4](#_Toc195520124)

[Téma ismertetése 4](#_Toc195520125)

[Téma választási indoklás 4](#_Toc195520126)

[Fejlesztendő program funkciói 4](#_Toc195520127)

[Célközönség 4](#_Toc195520128)

[Fejlesztői útmutató 5](#_Toc195520129)

[Fejlesztőkörnyezet és használt technológiák 5](#_Toc195520130)

[Hardveres környezet 5](#_Toc195520131)

[Fejlesztés során használt szoftverek 5](#_Toc195520132)

[Programnyelv és fejlesztőkörnyezet választásának indoklása 5](#_Toc195520133)

[A rendszer telepítése és futtatása 5](#_Toc195520134)

[Frontend indítása 5](#_Toc195520135)

[Backend indítása 6](#_Toc195520136)

[Projekt struktúra 7](#_Toc195520137)

[Adatbázis 7](#_Toc195520138)

[pizzas tábla: 8](#_Toc195520139)

[size tábla: 9](#_Toc195520140)

[toppings tábla: 10](#_Toc195520141)

[users tábla: 11](#_Toc195520142)

[orders tábla: 12](#_Toc195520143)

[orderTops tábla: 13](#_Toc195520144)

[pizzaToppings tábla: 14](#_Toc195520145)

[Kapcsolatok 15](#_Toc195520146)

[Alapértelmezett Adatok: 16](#_Toc195520147)

[Frontend 20](#_Toc195520148)

[HomePage.vue: 21](#_Toc195520149)

[MenuPage.vue: 22](#_Toc195520150)

[AdminPage.vue: 23](#_Toc195520151)

[Footer.vue: 24](#_Toc195520152)

[ForgotPasswordPage.vue: 25](#_Toc195520153)

[LoginPage.vue: 26](#_Toc195520154)

[Navbar.vue: 28](#_Toc195520155)

[OrderPage.vue: 29](#_Toc195520156)

[OrdersPage.vue: 30](#_Toc195520157)

[ProfilePage.vue: 31](#_Toc195520158)

[ResetPasswordPage.vue: 32](#_Toc195520159)

[Backend 33](#_Toc195520160)

[app.js: 33](#_Toc195520161)

[routes: 34](#_Toc195520162)

[controllers: 42](#_Toc195520163)

[services: 46](#_Toc195520164)

[middleware: 49](#_Toc195520165)

[models: 51](#_Toc195520166)

[Tesztelés 52](#_Toc195520167)

[Tesztfájlok 52](#_Toc195520168)

[Tesztek futtatása 52](#_Toc195520169)

[Példák 53](#_Toc195520170)

[Bejelentkezés 53](#_Toc195520171)

[Regisztráció 54](#_Toc195520172)

[Profil lekérés 55](#_Toc195520173)

[Profil frissítése 56](#_Toc195520174)

[Pizzák lekérése 57](#_Toc195520175)

[Hibakeresés 58](#_Toc195520176)

[Felhasználói dokumentáció 60](#_Toc195520177)

[Regisztráció 60](#_Toc195520178)

[Bejelentkezés 62](#_Toc195520179)

[Főoldal 64](#_Toc195520180)

[Admin Felület 65](#_Toc195520181)

[Rendelés kezelés oldal 67](#_Toc195520182)

[Menü oldal 68](#_Toc195520183)

[Rendelés felület 70](#_Toc195520184)

[Összefoglalás 72](#_Toc195520185)

[Irodalomjegyzék 73](#_Toc195520186)

# Bevezetés

## Téma ismertetése

A dolgozat egy olyan webalapú alkalmazás fejlesztésével foglalkozik, amely lehetővé teszi a felhasználók számára, hogy gyorsan és egyszerűen pizzát rendeljenek egy étterem weboldalán keresztül. A projekt célja egy felhasználóbarát és hatékony rendszer kialakítása, amely segíti a vendéglátóhelyeket online rendelési szolgáltatásuk optimalizálásában.

## Téma választási indoklás

A téma kiválasztása egy személyes tapasztalaton alapult: az egyik csapattag családtagja a vendéglátóiparban dolgozik, és a rendelési folyamat során felmerülő problémák inspiráltak minket a rendszer fejlesztésére. Úgy gondoltuk, hogy egy modern, letisztult és könnyen kezelhető rendelési platform létrehozásával hozzájárulhatunk a vendéglátás hatékonyságának növeléséhez.

## Fejlesztendő program funkciói

A rendszer fő funkciói közé tartozik:

* A pizzák böngészése és kiválasztása
* Kosárba helyezés és rendelés leadása
* Rendelés állapotának nyomon követése
* Felhasználói fiók létrehozása és kezelése

## Célközönség

A program elsődlegesen vendéglátóhelyek és pizzériák számára készült, akik online rendelési lehetőséget szeretnének biztosítani ügyfeleiknek. Emellett azokat a felhasználókat célozza meg, akik szeretnének gyorsan, kényelmesen és egyszerűen pizzát rendelni online.

# Fejlesztői útmutató

## Fejlesztőkörnyezet és használt technológiák

### Hardveres környezet

A fejlesztéshez használt számítógépek főként Intel Core i3-6100 processzorral és 8 GB RAM-mal rendelkeztek. Ez elegendő teljesítményt biztosított a fejlesztési és tesztelési folyamatokhoz.

### Fejlesztés során használt szoftverek

A projekt megvalósításához az alábbi szoftvereket és eszközöket használtuk:

* Visual Studio Code (VSC) – Kódszerkesztő és fejlesztői környezet
* XAMPP – Lokális szerverkörnyezet PHP és MySQL futtatásához
* MySQL – Az adatbázis-kezeléshez
* Google Chrome – A webalkalmazás teszteléséhez
* Node.js (20.17.0 verzió) – A backend fejlesztéséhez
* GitHub – Verziókezeléshez és a kód tárolására
* Trello – Feladatmenedzsment és csapatmunka szervezésére
* Gmail chat – A mindennapi kommunikáció biztosítására
* A Word kizárólag a dokumentáció elkészítéséhez volt szükséges.

## Programnyelv és fejlesztőkörnyezet választásának indoklása

A projekt fejlesztése során a JavaScript nyelvet használtuk mind a frontend, mind a backend oldalon, mivel széles körben támogatott, dinamikus és hatékony megoldást kínál a webalkalmazások fejlesztésére. A Node.js választása azért történt, mert lehetővé teszi a gyors és aszinkron szerveroldali műveletek végrehajtását, ami elengedhetetlen egy valós idejű rendelési rendszer esetében.

A fejlesztőkörnyezetként a Visual Studio Code-ot választottuk, mert könnyen testreszabható, számos bővítménnyel támogatott, és hatékonyan kezeli a JavaScript-alapú projekteket.

## A rendszer telepítése és futtatása

### Frontend indítása

* Navigálj a projekt mappájába:

cd PollakPizza

* Telepítsd a szükséges csomagokat:

npm install

* Indítsd el a weboldalt:

npm run dev

### Backend indítása

* Navigálj a backend mappába:

cd PollakPizza/backend

* Telepítsd a szükséges csomagokat:

npm install

* Töltsd fel az adatbázist:

Importáld a docs mappában található PZdatabase.sql fájlt a MySQL-be.

* Indítsd el a backend szervert:

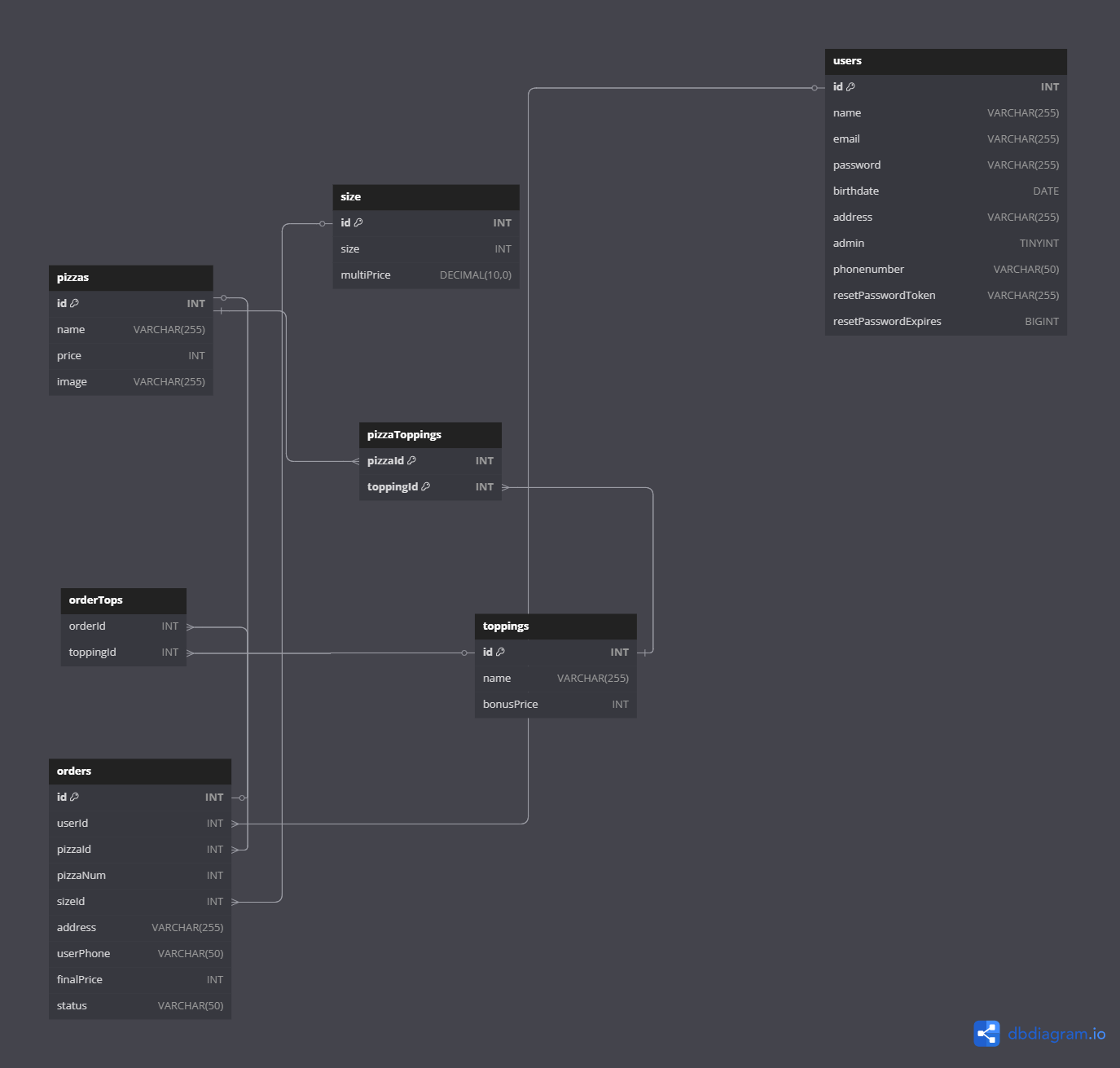
node .

Ez a fejlesztési környezet biztosítja, hogy a rendszer megfelelően működjön, és gördülékeny online pizza rendelési élményt nyújtson a felhasználók számára.

# Projekt struktúra

## Adatbázis

Az adatbázis felépítése:

Az alábbi adatbázis struktúra a pizzarendelő webalkalmazás adatait tárolja, és különböző táblák segítségével kezeli a felhasználók, rendeléseik, pizzák, méretek és feltétek közötti kapcsolatokat. Az adatbázis jól strukturált és kapcsolatokat teremt az egyes táblák között, hogy biztosítsa a hatékony és rugalmas adatkezelést.

### pizzas tábla:

A pizzas tábla a webalkalmazásban elérhető pizzák adatait tárolja. Minden rekord egy-egy pizza típusnak felel meg, amely tartalmazza a nevét, árát és a hozzá tartozó képet.

|  |  |  |
| --- | --- | --- |
| Oszlop | Típus | Leírás |
| id | INT | Egyedi pizza azonosító |
| name | VARCHAR(255) | A pizza neve (pl. Margherita) |
| price | INT | A pizza alapára. |
| image | VARCHAR(255) | URL útvonal a pizza képéhez. |

### size tábla:

A size tábla a különböző pizzaméreteket és azokhoz tartozó árnövekményeket tárolja. Minden rekord egy pizzaméretet jelöl, amelyhez egy hozzáadott ár tartozik.

|  |  |  |
| --- | --- | --- |
| Oszlop | Típus | Leírás |
| id | INT | Egyedi méret azonosító |
| size | INT | A pizza mérete (pl. 32). |
| multiPrice | Decimal(10,0) | Méret alapján árnövekmény. |

### toppings tábla:

A toppings tábla az elérhető pizzafeltéteket és azok többletköltségeit tárolja. Minden rekord egy-egy feltétet tartalmaz, amelyet a felhasználók hozzáadhatnak a pizzáikhoz, és amelyhez egy adott ár tartozik.

|  |  |  |
| --- | --- | --- |
| Oszlop | Típus | Leírás |
| id | INT | Egyedi feltét azonosító |
| name | VARCHAR(255) | A feltét neve (pl. Sajt). |
| bonusPrice | INT | A feltétért járó többletköltség |

### users tábla:

A users tábla a felhasználók adatait tárolja, akik regisztrálnak a rendszerben. Minden rekord egy-egy felhasználó adatait tartalmazza, mint például a nevét, e-mail címét, és a jelszavát.

|  |  |  |
| --- | --- | --- |
| Oszlop | Típus | Leírás |
| id | INT | Egyedi felhasználói azonosító |
| name | VARCHAR(255) | A felhasználó teljes neve |
| email | VARCHAR(255) | A felhasználó email címe |
| password | VARCHAR(255) | A felhasználó jelszava (titkosított) |
| birthdate | DATE | A felhasználó születési dátum |
| address | VARCHAR(255) | Kiszállítási cím. |
| admin | TINYINT | Admin státuszt jelöl (0/1). |
| phonenumber | VARCHAR(50) | Telefonszám |
| resetPasswordToken | VARCHAR(255) | Jelszó visszaállítási token |
| resetPasswordExpires | BIGINT | A visszaállító token lejárati ideje |

### orders tábla:

Az orders tábla a leadott pizzarendeléseket kezeli, és tárolja az egyes rendelésekkel kapcsolatos adatokat, például a rendelés dátumát, a felhasználót, aki leadta, és a rendelés állapotát.

|  |  |  |
| --- | --- | --- |
| Oszlop | Típus | Leírás |
| id | INT | Egyedi rendelési azonosító |
| userId | INT | Idegen kulcs a users táblára |
| pizzaId | INT | Idegen kulcs a pizzas táblára |
| pizzaNum | INT(11) | A rendelési pizza darabszáma |
| sizeId | INT | Idegen kulcs a size táblára. |
| address | VARCHAR(255) | A rendelés kiszállítási címe |
| userPhone | VARCHAR(50) | A rendelő telefonszáma |
| finalPrice | INT | A végleges ár |
| status | VARCHAR(50) | A rendelés státusza (pl. Kész). |

### orderTops tábla:

Az orderTops tábla a rendeléshez tartozó feltéteket tárolja. Mivel a rendeléshez több feltétel is hozzárendelhető, és egy feltét több rendeléshez is tartozhat, ez a tábla kezeli a több a többhöz kapcsolatot az orders és toppings táblák között.

|  |  |  |
| --- | --- | --- |
| Oszlop | Típus | Leírás |
| orderId | INT | Idegen kulcs az orders táblára |
| toppingId | INT | Idegen kulcs a toppings táblára. |

### pizzaToppings tábla:

A pizzaToppings tábla a pizzákhoz tartozó feltéteket tárolja. Mivel egy pizza több feltétet is tartalmazhat, és egy feltét több pizzán is szerepelhet, ez a tábla kezeli a több a többhöz kapcsolatot a pizzas és toppings táblák között.

|  |  |  |
| --- | --- | --- |
| Oszlop | Típus | Leírás |
| pizzaId | INT | Idegen kulcs a pizzas táblára |
| toppingId | INT | Idegen kulcs a toppings táblára. |

### Kapcsolatok

#### Egy-Többhöz kapcsolatok:

orders → users

orders → size

orders → pizzas

#### Több-Többhöz kapcsolatok:

pizzas ↔ toppings a pizzaToppings táblán keresztül.

orders ↔ toppings az orderTops táblán keresztül.

### Alapértelmezett Adatok:

#### Előre definiált pizzák:

|  |  |  |
| --- | --- | --- |
| Név | Ár(FT) | Kép URL |
| Margherita | 1000 | src/assets/image/pizzas.jpg |
| Pepperoni | 1200 | src/assets/image/pizzas.jpg |
| Hawaii | 1300 | src/assets/image/pizzas.jpg |
| Kukoricás | 1300 | src/assets/image/pizzas.jpg |
| Sonkás | 1300 | src/assets/image/pizzas.jpg |
| Kolbászos | 1300 | src/assets/image/pizzas.jpg |
| Baconos | 1300 | src/assets/image/pizzas.jpg |
| Sonkás-Gombás | 1300 | src/assets/image/pizzas.jpg |

#### Előre definiált feltétek:

|  |  |
| --- | --- |
| Név | Ár(FT) |
| Sajt | 200 |
| Pepperoni | 300 |
| Ananász | 250 |
| Sonka | 300 |
| Kukorica | 250 |
| Kolbász | 250 |
| Bacon | 250 |
| Gomba | 250 |
| Mozzarella | 250 |

#### Előre definiált méretek:

|  |  |
| --- | --- |
| Méret (cm) | Ár szorzó |
| 32 | 1 |
| 36 | 1,5 |
| 42 | 2 |
| 60 | 2,5 |

#### Pizza-Feltét Kapcsolatok:

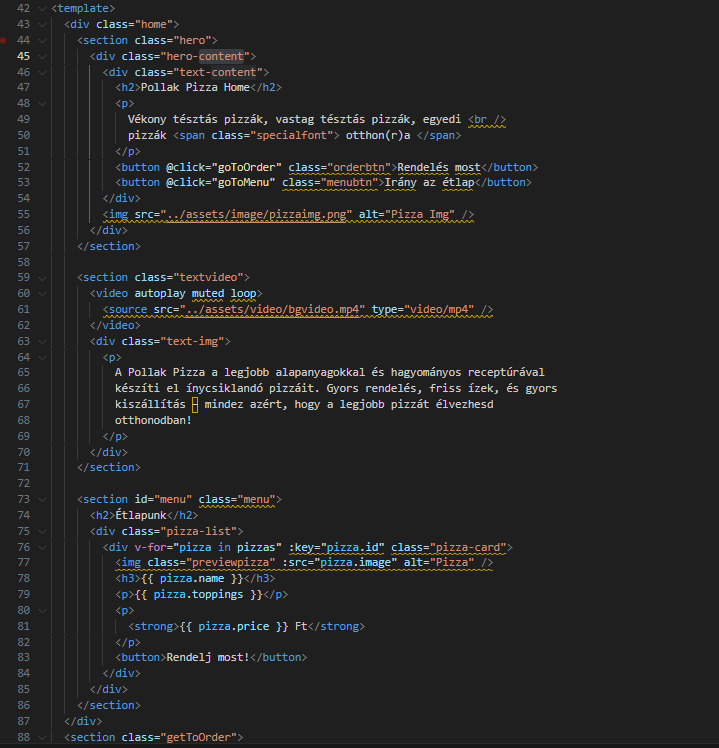
|  |  |
| --- | --- |
| Név | Ár(FT) |
| Margherita | Sajt |
| Pepperoni | Sajt, Pepperoni |
| Hawaii | Sajt, Ananász, Sonka. |
| Kukoricás | Sajt, Kukorica |
| Sonkás | Sajt, Sonka |
| Kolbászos | Sajt, Kolbász |
| Baconos | Sajt, Bacon |
| Sonkás-Gombás | Sajt, Sonka, Gomba |

## Frontend

A frontend a Vue.js keretrendszerre épül, amely lehetővé teszi a komponensalapú fejlesztést és a dinamikus felhasználói felület kialakítását. A rendszer több főbb komponensből áll, melyek együtt biztosítják a webalkalmazás működését.

Megjegyzés: Az alábbi képeken szereplő kódrészletek illusztrációként szolgálnak – nem minden esetben tartalmazzák a teljes forráskódot.

### HomePage.vue:

A HomePage.vue komponens a webalkalmazás főoldala, ahol a felhasználók megtekinthetik az étlapot, böngészhetnek az elérhető pizzák között, és rendelhetnek pizzát. A komponens tartalmazza az étlapot, amely az összes elérhető pizzát listázza, valamint lehetőséget biztosít a felhasználóknak, hogy kiválasszák a kívánt pizzákat, és hozzáadják őket a kosárhoz.

### MenuPage.vue:

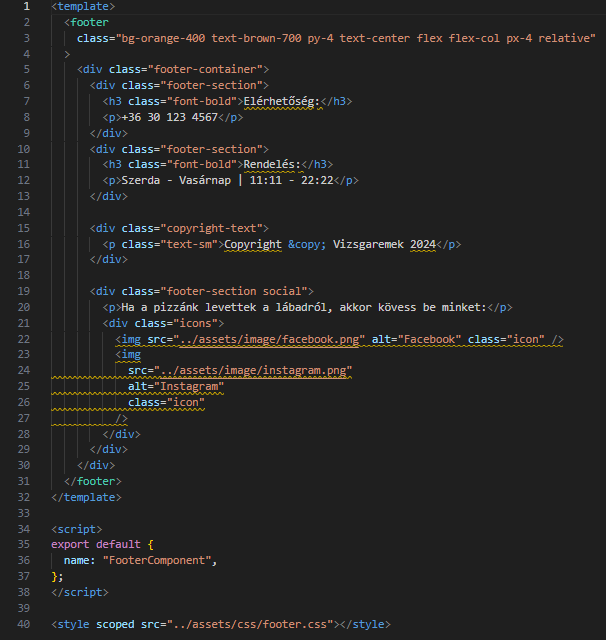
A MenuPage.vue komponens az étlap oldal, ahol a felhasználók megtekinthetik az elérhető pizzák listáját, böngészhetnek azok között, és rendelhetnek pizzát. Az oldal dinamikusan jeleníti meg a pizzák neveit, áraikat, valamint a rendeléshez szükséges egyéb információkat.



### AdminPage.vue:

Az AdminPage.vue komponens az adminisztrátori felületet biztosít, ahol az adminisztrátorok kezelhetik a pizzákat, felhasználókat, és egyéb adminisztratív funkciókat végezhetnek. Ez az oldal lehetővé teszi a pizzák hozzáadását, módosítását vagy törlését, valamint a felhasználók kezelését is.

### Footer.vue:

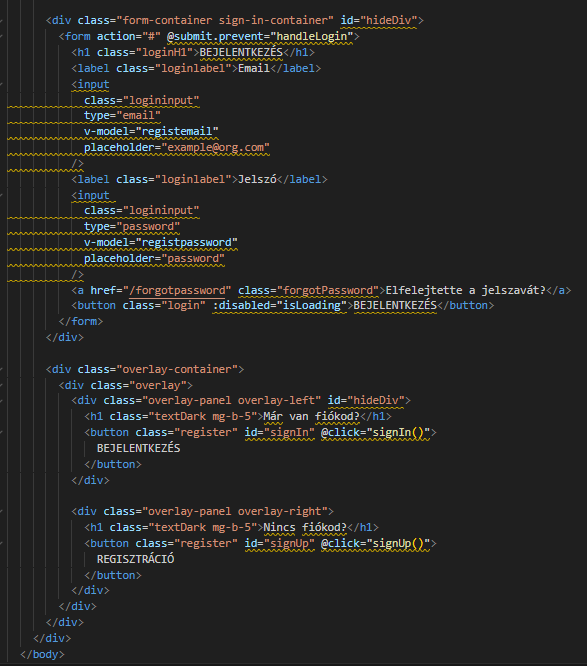
A Footer.vue komponens az oldalak alján elhelyezkedő láblécet biztosít, amely minden oldalra megjelenik, és tartalmazza a weboldal alapvető információit, mint például a kapcsolatokat, szerzői jogi adatokat, vagy a közösségi média hivatkozásokat.

### ForgotPasswordPage.vue:

A ForgotPasswordPage.vue komponens egy olyan oldal, amelyen a felhasználók megadhatják a regisztrált e-mail címüket, hogy jelszóhelyreállító linket kapjanak. Ez segít a felhasználóknak abban, hogy új jelszót állíthassanak be, ha elfelejtették a korábbit.

### LoginPage.vue:

A LoginPage.vue komponens az első oldal, amelyet a felhasználók látnak, amikor belépnek a weboldalra. Ezen az oldalon a felhasználók egyszerre bejelentkezhetnek, ha már regisztráltak, vagy regisztrálhatnak egy új fiókot, ha még nem rendelkeznek felhasználói fiókkal.



### Navbar.vue:

A Navbar.vue komponens a weboldal tetején található navigációs sávot biztosítja, amely lehetővé teszi a felhasználók számára, hogy könnyedén navigáljanak az oldalaink között. A navigációs sáv segít a felhasználóknak gyorsan elérni a különböző főbb oldalakat, mint a főoldal, étlap, kosár, admin felület, és más fontos részek.

### OrderPage.vue:

A CheckoutPage.vue komponens a rendelési oldal, ahol a felhasználók leadhatják rendeléseiket a weboldalon. Az oldal lehetőséget biztosít arra, hogy a felhasználók végigvigyék a rendelési folyamatot, válasszák ki a kívánt pizzákat, töltse ki a szükséges adatokat, és véglegesítsék a rendelést.

### OrdersPage.vue:

A OrdersPage.vue komponens az adminisztrátorok és dolgozók számára készült oldal, ahol nyomon követhetik a bejövő rendeléseket, és módosíthatják azok státuszát. Ez az oldal lehetővé teszi a rendelési adatok kezelését és frissítését a munkahelyi hatékonyság növelése érdekében.

### ProfilePage.vue:

A ProfilePage.vue komponens egy olyan oldal, ahol a felhasználók megtekinthetik és módosíthatják a rendszerben tárolt személyes adataikat. Ez az oldal lehetőséget biztosít a felhasználók számára, hogy frissítsék a profiljukat, így biztosítva a gyorsabb bejelentkezést és rendelést.



### ResetPasswordPage.vue:

A ResetPasswordPage.vue komponens azon oldal, ahol a felhasználók akkor tudják megváltoztatni elfelejtett jelszavukat, ha a regisztrált e-mail címükre küldött linkre kattintanak. Az oldal biztosítja a jelszó biztonságos visszaállításának folyamatát, amely segít a felhasználóknak gyorsan visszanyerni a hozzáférésüket a fiókjukhoz.

## Backend

A backend a Node.js környezetre és az Express.js keretrendszerre épül, amely lehetővé teszi egy stabil, bővíthető és jól strukturált szerveroldali alkalmazás létrehozását. A rendszer több főbb modulra tagolódik, amelyek külön-külön felelősek az egyes funkciók kezeléséért.

### app.js:

Az app.js a Node.js és Express alapú alkalmazásunk fő inicializáló fájlja, amely a szerver indításáért és a fő útvonalak beállításáért felelős. Az Express keretrendszert használva az alkalmazás biztosítja, hogy a felhasználók számára elérhető legyen minden szükséges API végpont, amelyek különböző funkciókat látnak el, mint például rendelésfelvétel, felhasználókezelés és egyéb szolgáltatások.



### routes:

A **routes** mappa és fájlok az alkalmazás különböző útvonalait (vagy endpointjait) kezelik. Az Express keretrendszerben az útvonalak határozzák meg, hogyan reagáljon az alkalmazás a bejövő HTTP kérésekre, például GET, POST, PUT, DELETE típusú kérésekre. Mivel a **routes** mappa általában az alkalmazás logikai struktúrájának központját képezi, itt történik a különböző erőforrások (például felhasználók, pizzák, rendelés) kezelésének és működésének beállítása.addPizzas.js: Pizza hozzáadására, és feltétek hozzáadására szolgál



#### admin.js:

Az admin.js fájl az adminisztrátori felülethez kapcsolódó útvonalak kezeléséért felelős. Ezen keresztül az adminisztrátorok képesek kezelni a felhasználói adatokat, valamint figyelemmel kísérhetik és módosíthatják a rendeléseket. Az adminisztrátori funkciók általában magasabb szintű jogosultságokkal rendelkező felhasználók számára elérhetők, és az adminisztrátorok számára biztosítanak lehetőséget a rendszer kezelésére.

#### allpizzas.js:

Az allpizzas.js fájl egy egyszerű Express útvonalat tartalmaz, amely lehetővé teszi a felhasználók számára, hogy véletlenszerűen lekérdezzenek egy adott számú pizzát az adatbázisból. Ez a funkció hasznos lehet olyan esetekben, amikor szeretnénk egy változatos pizzaválasztékot kínálni a felhasználóknak, például a kezdőlap vagy étlap megjelenítésekor.

#### allsizes.js:

Az allsizes.js fájl egy Express útvonalat tartalmaz, amely lehetővé teszi, hogy lekérjük az összes pizzaméretet az adatbázisból. Ezt az útvonalat általában akkor használják, amikor a felhasználó szeretné megtekinteni az elérhető pizzaméreteket, például egy rendelés leadása előtt, hogy kiválaszthassa a kívánt pizzaméretet.

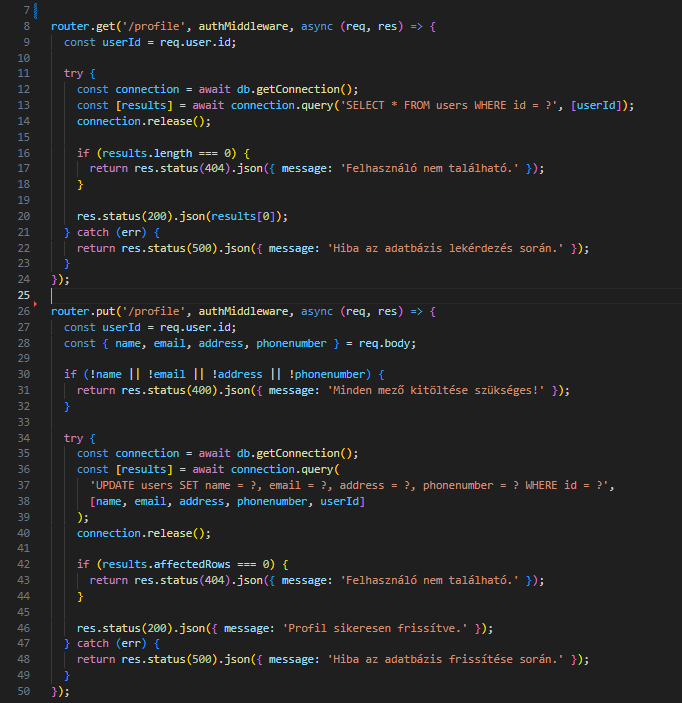
#### orders.js:

Az orders.js fájl egy Express útvonalat tartalmaz, amely a rendelésekkel kapcsolatos különböző műveleteket kezeli. Ez magában foglalja a rendelés lekérdezését, frissítését, a kész rendelés törlését, valamint az új rendelés feladását. Az útvonalak hasznosak lehetnek a rendelésekkel kapcsolatos adminisztrációs és felhasználói műveletek során.

#### auth.js:

Az auth.js fájl az Express alkalmazásban található, és a felhasználók bejelentkezési és regisztrációs folyamatait kezeli. Az útvonalak segítségével a felhasználók regisztrálhatnak, bejelentkezhetnek, és kezelhetik a jelszavakat. Ezen kívül tartalmazhat token alapú autentikációt, mint például JWT (JSON Web Token), amely biztosítja a felhasználói munkamenetek biztonságos kezelését.

#### profile.js:

A profile.js fájl az Express alkalmazásban található, és a felhasználói profil kezeléséért felelős útvonalakat tartalmazza. A felhasználók ezen az útvonalon keresztül módosíthatják a saját adataikat, például nevüket, címüket, telefonszámukat vagy akár jelszavakat is. A profil adatainak frissítése biztonságos módon történik, és az adatok a backend adatbázisban frissülnek.

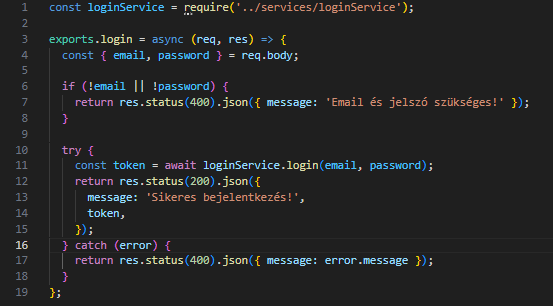
#### pizzas.js:

A pizzas.js fájl az Express alkalmazásban található, és a pizzák adatainak lekérdezéséért felelős útvonalat tartalmazza. Ennek az útvonalnak a feladata, hogy visszaadja a webalkalmazásban elérhető pizzák listáját, amely tartalmazza az egyes pizzák nevét, árát, és képet. Ezt az adatot az alkalmazás frontendjén használják a pizzák megjelenítésére, például az étlap oldalon.

### controllers:

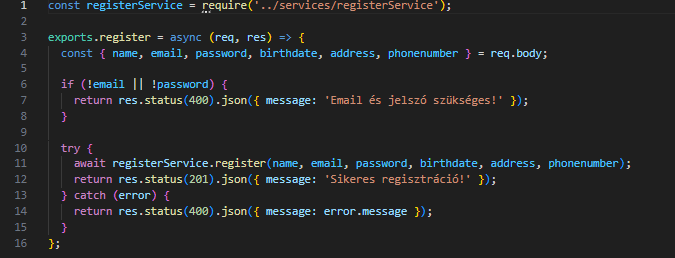
A controllers mappa az Express alkalmazásban a különböző útvonalakhoz (routes) tartozó vezérlőket tartalmazza. A vezérlők felelnek az alkalmazás logikájáért, amely az útvonalak kéréseit kezeli, adatokat kér le az adatbázisból, és válaszokat küld a felhasználónak. A vezérlők segítenek elválasztani az alkalmazás üzleti logikáját az útvonalak kezelésétől, így tisztább és karbantarthatóbb kódot eredményeznek.

#### loginController.js:

A loginController.js felelős a felhasználók bejelentkezési folyamatának kezeléséért az Express alkalmazásban. Az alapvető logika a felhasználó hitelesítésére és érvényesítésére épül, hogy biztosítsuk, hogy csak azok férjenek hozzá a rendszerhez, akik érvényes bejelentkezési adatokat adtak meg.

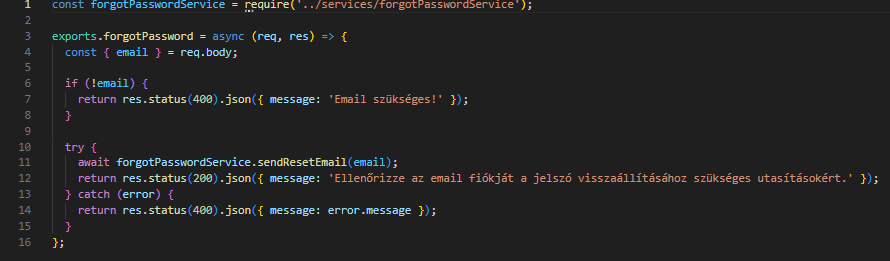
#### registerController.js:

A registerController.js a felhasználói regisztrációs folyamatot kezeli az Express alkalmazásban. Ez a vezérlő lehetővé teszi az új felhasználók számára, hogy regisztráljanak a rendszerben, ellenőrzi a beírt adatokat, titkosítja a jelszót, és létrehozza az új felhasználói fiókot.



#### forgotPasswordController.js:

A forgotPasswordController.js vezérlő feladata, hogy kezelje a felhasználók jelszó visszaállítási kérelmeit. Ez az útvonal lehetővé teszi a felhasználók számára, hogy a regisztrált e-mail címük megadásával kérjenek egy jelszó helyreállító linket. Az e-mailben küldött link egy tokennel rendelkezik, amely a felhasználót a jelszó módosító oldalra irányítja.



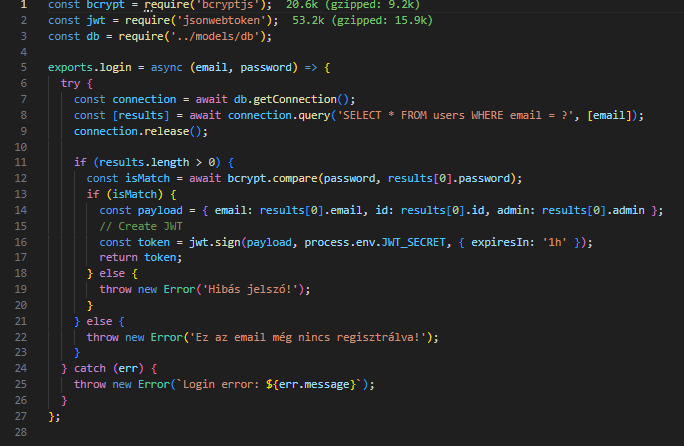
#### resetPasswordController.js:

A forgotPasswordController.js a felhasználók elfelejtett jelszavának kezelését végzi. A felhasználó megadja az e-mail címét, és a rendszer egy jelszó-visszaállító linket küld a megadott e-mail címre. Ez a vezérlő biztosítja, hogy az e-mail cím érvényes legyen, és ha létezik a rendszerben, elküldi a jelszó visszaállító linket, amely segítségével a felhasználó új jelszót állíthat be.

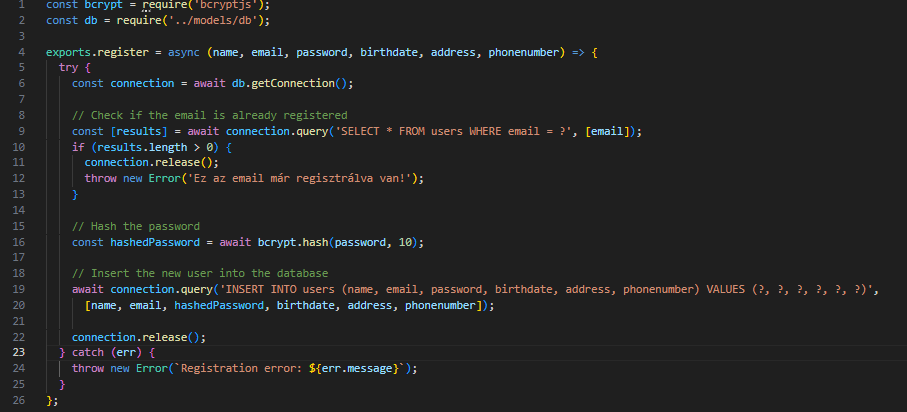
### services:

A services mappa az alkalmazás üzleti logikájáért felelős kódot tartalmazza. A vezérlők által hívott szolgáltatások kezelik az adatbázissal kapcsolatos műveleteket, valamint az alkalmazás üzleti logikáját, például a felhasználói adatok kezelését, az árak számítását, vagy egyéb összetett műveleteket. Az adatbázis műveletek elvégzése és a logika implementálása külön szolgáltatásokban történik, hogy a kód jól strukturált és karbantartható legyen.

#### loginService.js:

A loginService.js a bejelentkezési logikát kezeli. Feladata a felhasználó hitelesítése, amely a megadott e-mail cím és jelszó alapján történik. A bejelentkezés során az alkalmazás ellenőrzi, hogy a felhasználó létezik-e, és ha igen, akkor összehasonlítja a megadott jelszót a tárolt titkosított jelszóval. Ha a hitelesítés sikeres, egy JSON Web Token (JWT) kerül generálásra és visszaküldésre, amely a továbbiakban a felhasználó azonosítására szolgálhat.

#### registerService.js:

A registerService.js felelős a felhasználók regisztrálásáért. A regisztrációs folyamat során a felhasználó megadja a szükséges adatokat, mint például a nevét, e-mail címét, jelszavát és egyéb elérhetőségi adatokat. A jelszót titkosítani kell, mielőtt tárolnánk az adatbázisban. Ha minden adat megfelelő, a felhasználó regisztrációja sikeres, és létrejön egy új rekord az adatbázisban.

#### forgotPasswordService.js:

A forgotPasswordService.js a felhasználók jelszó visszaállítási folyamatát kezeli. A szolgáltatás ellenőrzi, hogy a felhasználó által megadott e-mail cím létezik-e a rendszerben. Ha létezik, akkor a rendszer generál egy egyedi jelszó-visszaállító tokent, amelyet e-mailben küld el a felhasználónak. Az e-mail tartalmazza a jelszó-visszaállító linket, amelyen keresztül a felhasználó új jelszót állíthat be.

### middleware:

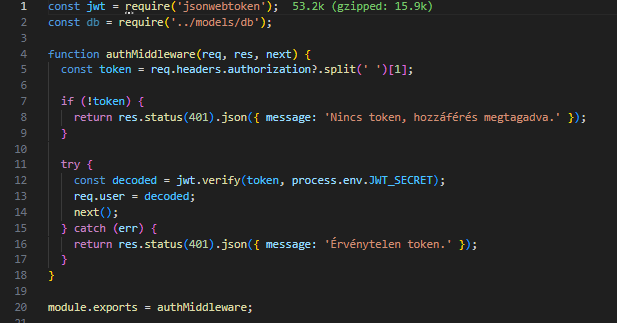
A middleware mappa az alkalmazás köztes rétegeit tartalmazza. A köztes rétegek olyan funkciók, amelyek az egyes HTTP kérések feldolgozása előtt, alatt vagy után futnak. Ezeket gyakran használják autentikáció, jogosultság-ellenőrzés, naplózás, hibakezelés, és egyéb kiegészítő logikai műveletek végrehajtására.

A köztes rétegek lehetővé teszik, hogy az alkalmazás különböző részeihez hozzáférjünk, feldolgozzuk a kéréseket, majd továbbítsuk azokat a megfelelő útvonalakra vagy vezérlőkhöz.

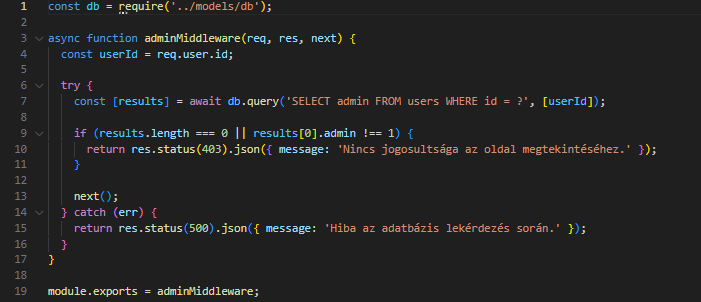
#### auth.js:

Az **auth.js** köztes réteg a **JSON Web Token (JWT)** alapú hitelesítést kezeli. Ez a köztes réteg megvizsgálja a bejövő HTTP kérésekhez csatolt JWT tokent, és ellenőrzi annak érvényességét. Ha a token érvényes, akkor lehetővé teszi a kérés további feldolgozását, míg, ha a token érvénytelen vagy hiányzik, a kérés elutasításra kerül.

Ez a mechanizmus biztosítja, hogy a felhasználók csak akkor férhessenek hozzá védett erőforrásokhoz, ha sikeresen bejelentkeztek, és rendelkeznek érvényes JWT tokennel.



#### admin.js:

Az admin.js egy köztes réteg, amely ellenőrzi, hogy a felhasználó rendelkezik-e admin jogosultságokkal. Ez fontos lépés a rendszer védelme érdekében, hogy csak a megfelelő jogosultsággal rendelkező felhasználók férhessenek hozzá adminisztratív feladatokhoz, például felhasználók kezelése, rendeléskezelés, vagy bármilyen érzékeny információhoz.

### models:

A models mappa az alkalmazás adatbázis modelljeit tartalmazza. Az adatbázis modellek definiálják az adatbázis táblákat, azok mezőit, típusait és a mezők közötti kapcsolatokat. Az alkalmazás üzleti logikája a modelleken keresztül éri el az adatokat, és a modellek felelősek az adatok érvényesítéséért, manipulálásáért és lekérdezéséért.

#### db.js:

A db.js felelős az adatbázis kapcsolódás beállításáért és kezeléséért. Ebben a fájlban konfiguráljuk a MySQL adatbázis kapcsolatot, hogy az alkalmazás adatokat tudjon lekérdezni, módosítani és tárolni az adatbázisban. A kapcsolatot egyszer beállítjuk, és ezt a beállítást más fájlokban (például a modellekben vagy a szolgáltatásokban) használhatjuk a MySQL műveletekhez.

# Tesztelés

A Jest és a Supertest kombinációja ideális a Node.js alkalmazások teszteléséhez. A Jest egy tesztelő keretrendszer, amely segít az egység- és integrációs tesztek megírásában, míg a Supertest lehetővé teszi az HTTP kérések egyszerű tesztelését, például API végpontok válaszainak ellenőrzését.

## Tesztfájlok

A tests mappában tárolhatók a különböző tesztfájlok, amelyek a projekt egyes funkcióit tesztelik. Mivel a projekt különböző útvonalakat, például bejelentkezési, regisztrációs, profil és pizzák kezelését valósít meg, külön tesztfájlok készíthetők minden egyes funkcióhoz. Az alábbiakban bemutatom a legfontosabb tesztfájlokat és azok alapvető tesztjeit.

* authRoutes.test.js: Bejelentkezési és regisztrációs útvonalak tesztelése
* profileRoutes.test.js: Felhasználói profil útvonalak tesztelése
* pizzaRoutes.test.js: Pizzák listájának lekérdezése

## Tesztek futtatása

A tesztek futtatásához egyszerűen használd a következő parancsot a terminálban:

npm test

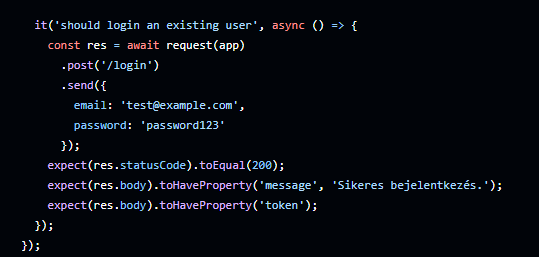
## Példák

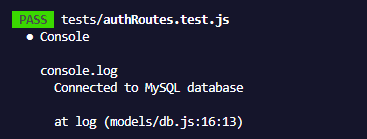
### Bejelentkezés

A bejelentkezési útvonal a /login végponton érhető el. A felhasználónak a következő adatokat kell megadnia a kérésben:

email: A regisztrált felhasználó email címe.

password: A felhasználó titkosított jelszava.

Példa kérés: 

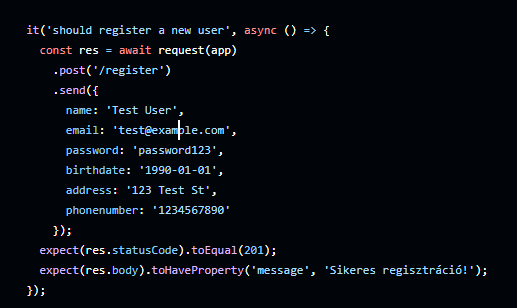


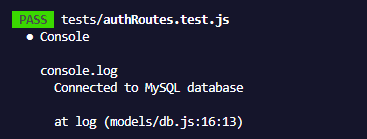
### Regisztráció

A regisztrációs útvonal a /register végponton érhető el. A felhasználónak a következő adatokat kell megadnia a kérésben:

* name: A felhasználó teljes neve.
* email: A felhasználó email címe, amely a regisztráció során egyedinek kell lennie.
* password: A felhasználó titkosított jelszava.
* birthdate: A felhasználó születési dátuma.
* address: A felhasználó kiszállítási címe.
* phonenumber: A felhasználó telefonszáma.

Példa kérés:

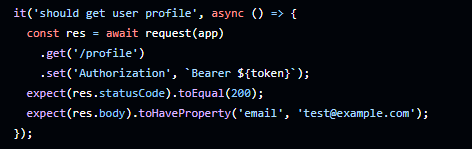


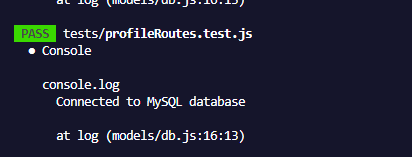


### Profil lekérés

A felhasználói profil lekérdezése a /profile végponton érhető el. A kérésnek tartalmaznia kell a felhasználói azonosítót vagy a JWT (JSON Web Token) hitelesítést a kérésben. A válasz tartalmazza a felhasználó adatait, mint például a nevét, email címét.

Példa kérés:

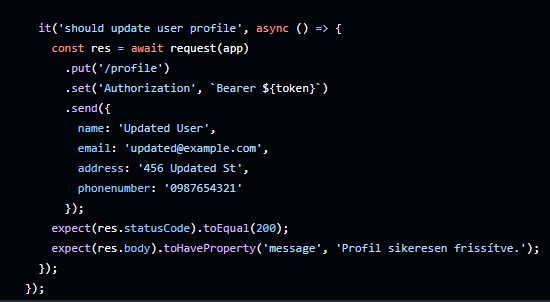


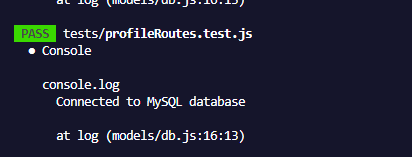


### Profil frissítése

A felhasználói profil frissítése a /profile végponton érhető el. A kéréshez érvényes JWT token szükséges az Authorization fejlécben, valamint azokat az adatokat, amelyeket frissíteni szeretne a felhasználó. Az adatokat JSON formátumban kell megadni, és ezek tartalmazhatják a felhasználó nevét, email címét, címét vagy telefonszámát.

Példa kérés:

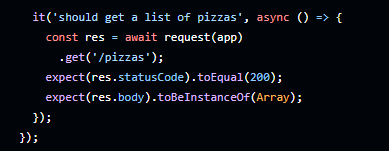


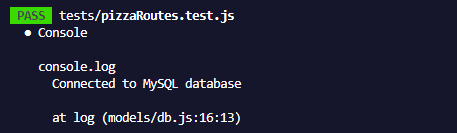


### Pizzák lekérése

A pizzák listájának lekérdezése a /pizzas végponton érhető el. Ez az útvonal lehetővé teszi, hogy a felhasználók lekérjék az összes elérhető pizzát az étlapon. A válasz tartalmazza az egyes pizzák adatait, mint például a pizza nevét, árát és képét.

Példa kérés:





## Hibakeresés

A hibakeresés során, ha problémák merülnek fel a tesztek futtatása közben, az alábbi lépéseket érdemes követni:

* Adatbázis elérhetősége és beállításai

Adatbázis kapcsolat: Ellenőrizd, hogy a projekt megfelelően csatlakozik az adatbázishoz. Nyisd meg a db.js fájlt, és ellenőrizd, hogy a helyes adatbázis konfiguráció van megadva (pl. felhasználónév, jelszó, host, port).

Tesztadatok: Győződj meg róla, hogy a tesztadatok (pl. felhasználók, pizzák, rendelések) elérhetőek az adatbázisban. Ha szükséges, manuálisan töltsd fel a tesztadatokat.

Adatbázis kapcsolat: Ha az adatbázis nem válaszol, próbáld újraindítani a MySQL szolgáltatást, és ellenőrizd, hogy a megfelelő porton fut.

* A szükséges csomagok telepítése

Telepített csomagok ellenőrzése: Győződj meg róla, hogy minden szükséges csomag telepítve van a projektben. Futtasd a következő parancsokat a projekt könyvtárában:

npm install

Követelmények: Ellenőrizd, hogy a package.json fájlban szereplő összes függőség telepítve van, különösen a teszteléshez szükséges könyvtárak (pl. jest, supertest).

Ellenőrizd a verziókat: Győződj meg róla, hogy az alkalmazott csomagok verziója kompatibilis a projekttel. Ha szükséges, frissítsd a csomagokat.

* Tesztek futtatása előtt

Tesztesetek előkészítése: Ellenőrizd, hogy a tesztadatok megfelelően vannak előkészítve, és hogy a szükséges rekordok már jelen vannak az adatbázisban. A tesztek nem fognak működni, ha a szükséges adatok hiányoznak.

Ha például egy felhasználói bejelentkezést tesztelsz, győződj meg róla, hogy van érvényes felhasználó az adatbázisban.

Tisztítás: A tesztek futtatása előtt tisztítsd meg az adatbázist a nem szükséges rekordoktól. Egyes tesztek nem működnek megfelelően, ha az adatbázisban felhalmozódnak a régi adatok. Egy adatbázis visszaállítási vagy törlési folyamat is segíthet az ilyen helyzetek elkerülésében.

Tesztek izolálása: Gondoskodj róla, hogy a tesztek nem függnek más tesztektől. Ha egy teszt sikertelen, ne befolyásolja a következő teszteket. Ha szükséges, használd az adatbázis tranzakciókat, hogy minden teszt után visszaállíthasd az állapotot.

* Tesztelési naplózás és hibák

Részletes naplók: A tesztek futtatásakor mindig nézd meg a részletes naplókat. Az esetlegesen felmerülő hibák leírásai segíthetnek azonosítani a problémát.

Hibák elemzése: Ha egy teszt hibát ad, vizsgáld meg, hogy a hiba az adatbázishoz kapcsolódik-e, vagy esetleg az alkalmazás logikájában keresendő.

* Tesztelési környezet

Tesztkörnyezet: Ha lehetséges, külön tesztkörnyezetet használj a fejlesztési környezettől, hogy ne zavarják egymást a tesztelések és a valós működés. Ez különösen fontos adatbázis-tesztelésnél.

Környezet változók: Ellenőrizd, hogy a szükséges környezeti változók (pl. DATABASE\_URL, JWT\_SECRET) helyesen vannak beállítva a fejlesztési, tesztelési és éles környezetben.

* Tesztelési eszközök frissítése

Jest és Supertest frissítése: Ha a tesztek nem működnek megfelelően, próbáld frissíteni a használt tesztelési könyvtárakat, például:

npm update jest supertest

Jest konfiguráció: Győződj meg róla, hogy a jest.config.js fájl megfelelően van beállítva. A konfigurációs fájl segít a tesztelési környezet megfelelő beállításában.

Ha a fenti lépések követése után továbbra is problémák merülnek fel, érdemes részletesebben átnézni a kódban lévő logikát, illetve konzultálni a csapattal vagy a fejlesztői közösséggel.

# Felhasználói dokumentáció

Ez a dokumentum bemutatja, hogyan használhatja a weboldalt a felhasználó, hogy a legjobb élményt nyújtsa a pizzarendelés során. Az alábbiakban ismertetjük az oldal főbb funkcióit, és lépésről lépésre bemutatjuk, hogyan tudja kihasználni az összes lehetőséget.

## Regisztráció

* Bejelentkezés oldal megnyitása

Amikor először lép be az oldalra, a Bejelentkezési oldal fogja fogadni. Itt lehetősége van bejelentkezni, de ha még nincs fiókja, a "REGISZTRÁCIÓ" gombbal regisztrálhat.

* Regisztráció

A "REGISZTRÁCIÓ" gombra kattintva átirányítjuk a Regisztrációs oldalra.

Itt meg kell adnia az alábbi adatokat:

Név: Adja meg a teljes nevét.

Születési dátum: Kattintson a naptár ikonra, és válassza ki a születése dátumát, vagy írja be manuálisan.

Jelszó: Adjon meg egy jelszót, amit később a bejelentkezéshez használni fog. Fontos, hogy biztonságos jelszót válasszon.

Telefonszám: Adja meg a telefonszámát.

Email cím: Az email címét adja meg, amelyet később a bejelentkezéshez fog használni.

Cím: Végül adja meg a kiszállítási címét, ahol a pizzáját szeretné megkapni.

* Regisztráció befejezése

Miután minden adatot kitöltött, kattintson a "REGISZTRÁCIÓ" gombra.

A rendszer ellenőrzi az adatokat, és ha minden rendben van, egy visszajelzést ad, amely megerősíti a sikeres regisztrációt.

* További lépések

Sikeres regisztráció után automatikusan beléphet az oldalra, és hozzáférhet a felhasználói fiókjához, ahol később módosíthatja adatait vagy rendeléseket adhat le.

## Bejelentkezés

Miután sikeresen regisztrált (vagy már rendelkezik egy felhasználói fiókkal), a bejelentkezési folyamat rendkívül egyszerű.

Bejelentkezéshez szükséges adatok:

* Email cím: *Az email cím, amivel regisztrált a weboldalon.*
* Jelszó: *A regisztrációkor megadott jelszó.*

Bejelentkezés lépései:

* Nyissa meg a Bejelentkezési oldalt.
* Adja meg az email címét és a jelszavát.
* Kattintson a "Bejelentkezés" gombra.

Sikeres bejelentkezés után hozzáférhet a felhasználói fiókjához, ahol kezelheti rendeléseit, szerkesztheti profilját, és nyomon követheti a rendelés állapotát.

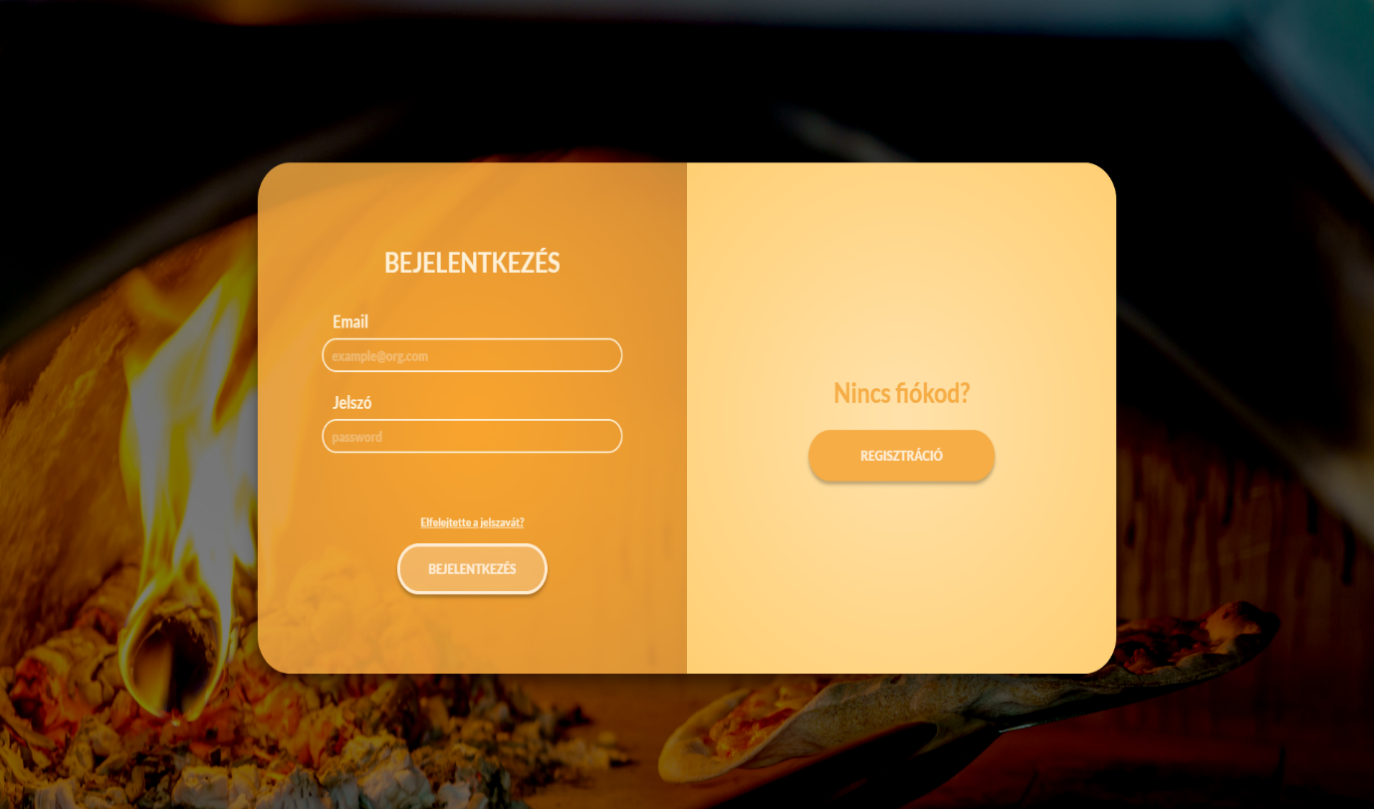
* Elfelejtett Jelszó
* Ha elfelejtette a jelszavát, az "Elfelejtette a jelszavát?" lehetőséget választhatja.

Jelszó visszaállítási lépések:

* Kattintson az "Elfelejtette a jelszavát?" linkre a bejelentkezési oldalon.
* Adja meg azt az email címet, amivel regisztrált.
* Ha helyes email címet adott meg, a rendszer küld egy jelszó-visszaállító linket az Ön által megadott email címre.
* Nyissa meg az emailt, és kattintson a benne található linkre.
* Ez átirányítja a jelszó-visszaállító oldalra, ahol új jelszót tud megadni.
* Miután megváltoztatta a jelszót, már ismét be tud lépni a fiókjába a frissített jelszóval.

*Bejelentkezés új jelszóval*

Miután sikeresen megváltoztatta a jelszavát, visszatérhet a Bejelentkezési oldalra, és bejelentkezhet az új jelszavával.



## Főoldal

A Főoldal az első oldal, amelyet a felhasználó lát, miután sikeresen bejelentkezett. Az oldalon könnyedén elérheti a szolgáltatásaink nagy részét és navigálhat a különböző funkciók között.

Főoldal jellemzői:

* Pizza ajánlatok megtekintése:

Az oldal tetején található szakaszban pizza ajánlatokat tekinthet meg. Itt különféle pizzák, akciók és ajánlatok várják.

Ha érdeklik a friss ajánlatok, egyszerűen görgessen lejjebb, és böngésszen a pizzák között.

* Rendelés indítása:

Rendelés gomb: Az oldal alján található "Rendelés" gombra kattintva kezdhet új rendelést. Ezen keresztül könnyedén elérhet egy új rendelési felületet.

Navigációs sáv: A navigációs sávon is rákattinthat a "Rendelés" menüpontra, amely szintén elviszi a rendelési oldalra, ahol hozzáadhatja kedvenc pizzáját a kosárhoz.

* Pizza lista megtekintése:

Ha kíváncsi az összes elérhető pizzára, navigálhat a Pizzák listájához, ahol az összes étlapunkat megtekintheti.

Itt lehetősége van arra, hogy azonnal rendeljen egy pizzát, vagy helyezze a választott pizzát a kosárba és folytassa a vásárlást a későbbiekben.

* Kényelmes navigáció:

A főoldalon könnyen hozzáférhet a legfontosabb funkciókhoz, és a felhasználóbarát kialakításnak köszönhetően gyorsan elérheti a kívánt szolgáltatásokat.

A navigációs sáv és a rendelés gombok segítségével nemcsak gyorsan rendelhet pizzát, hanem a pizza listát is könnyedén áttekintheti.

## Admin Felület

Az Admin Felület a webalkalmazás azon része, amely lehetővé teszi az adminisztrátorok számára, hogy kezeljék a felhasználókat és az étlapot, beleértve a pizzákat és azok jellemzőit. Az oldal két fő részre van osztva: Felhasználó Kezelés és Pizza Kezelés.

* Felhasználó Kezelés:

Az adminisztrátor képes kezelni a felhasználói adatokat.

A felhasználó adatainak módosítása és törlésük is lehetséges az adminisztrátor számára.

Felhasználói adatok módosítása: Az admin képes a felhasználó nevének, email címének, jelszavának, telefonszámának, címének és más fontos információk módosítására.

Felhasználó törlés: Az admin lehetőséget kap arra, hogy törölje a felhasználókat a rendszerből, amennyiben szükséges.

* Pizza Kezelés:

Pizza lista megtekintése: Az admin megtekintheti az összes jelenlegi pizzát, azok árait, a hozzájuk tartozó feltéteket és a pizzák képeit.

Pizza adatok: Az admin számára az összes fontos adat elérhető, beleértve a pizzák nevét, árát, feltéteit és a hozzájuk rendelt képeket.

* Új pizza hozzáadása:

Az admin felület tartalmaz egy gombot, amelyet megnyomva megjelennek a beviteli mezők, amelyek segítségével az admin hozzáadhat egy új pizzát a menühöz.

Beviteli mezők: Az admin megadhatja az új pizza nevét, alapárát, képét és a választott feltéteket, amelyeket az új pizzához hozzárendelhet.

A beviteli mezők biztosítják, hogy az adminisztrátor könnyedén hozzáadhassa az új pizzát a rendszerhez a kívánt paraméterekkel.

* Admin Felület Jellemzői:

A felület felhasználóbarát és egyszerűen kezelhető, így az adminisztrátorok könnyedén végezhetik el a szükséges módosításokat.

Az adminisztrátorok számára elérhetőek a funkciók, amelyek biztosítják a felhasználók kezelését és a pizzák dinamikus módosítását.

A pizza hozzáadása egy egyszerű gombnyomással történik, biztosítva a gyors és hatékony munkafolyamatot.



## Rendelés kezelés oldal

A Rendelés Kezelés Oldal a webalkalmazás azon része, amely az adminisztrátorok és alkalmazottak számára lehetővé teszi a leadott rendelésekkel kapcsolatos adatok megtekintését és kezelését. Ez az oldal segíti a rendelésekkel kapcsolatos állapotok frissítését és a lezárult rendelések nyomon követését.

* Funkciók:

Rendelések megtekintése:

Az adminok és alkalmazottak képesek megtekinteni a jelenlegi rendeléseket, amelyek a rendszerben beérkeztek.

Az oldal bemutatja a rendelés adatait, beleértve a felhasználó nevét, rendelés dátumát, pizzák listáját, méretet, feltéteket, kiszállítási címet és rendelés státuszát.

* Státusz módosítása - "Kész" jelölés:

Az adminok és alkalmazottak képesek a rendeléseket „Kész” állapotba állítani, jelezve, hogy a rendelés elkészült és kiszállításra vagy átvételre kész.

Amint egy rendelést „Kész”-re állítanak, az automatikusan áthelyeződik a "Kész rendelések" listába, hogy könnyebben nyomon követhessék a befejezett rendeléseket.

* Kész rendelések:

A rendelés a „Kész” státusz beállítása után átkerül a „Kész rendelések” részbe, ahol az adminok és alkalmazottak könnyedén megtekinthetik a lezárt rendelések listáját.

Ez segít az adminisztrátoroknak és alkalmazottaknak abban, hogy nyomon kövessék a befejezett rendeléseket és biztosítsák a szállítást vagy átvételt.

* Rendelés törlés:

Amint egy rendelés befejeződött és már nem szükséges, az adminisztrátorok és alkalmazottak képesek törölni a rendeléseket a rendszerből.

Ez segít a tisztán tartani a rendelés listát és megszabadulni a már lezárt rendelésektől.

* Jellemzők:

Adminisztrátorok és alkalmazottak számára elérhető: Csak az adminok és alkalmazottak férhetnek hozzá a rendeléskezelési felülethez, így biztosítva, hogy a vásárlói adatok és rendelések kezelése biztonságos legyen.

Egyszerű státusz frissítés: A rendelés állapotának módosítása könnyen elvégezhető egy kattintással, amely segít gyorsítani a rendeléskezelési folyamatot.

Átláthatóság: A rendszer egyértelműen megkülönbözteti a „Kész” rendelések listáját a folyamatban lévő rendelésekétől, így az adminok és alkalmazottak könnyedén navigálhatnak.

## Menü oldal

A Menü Oldal célja, hogy a felhasználók könnyedén megtekinthessék az elérhető pizzákat, és egyszerűen megtalálják az általuk preferált pizzát a rendeléshez. Az oldal lehetőséget biztosít arra, hogy a felhasználók ne csak böngésszenek a pizzák között, hanem szűrjenek is, például név és feltét alapján, hogy gyorsabban rátaláljanak a kívánt ételre.

* Funkciók:

Pizza lista megtekintése:

Az oldal tartalmazza az összes pizzát, amit a rendszerben elérhetők. Minden egyes pizza információt, mint a neve, ára, képe és feltétek listája, könnyedén elérhetik a felhasználók.

* Keresési lehetőség:

Név szerinti keresés: A felhasználók a pizzák nevét megadva kereshetnek az oldalon. A rendszer megjeleníti azokat a pizzákat, amelyek nevében szerepel a keresett kifejezés.

Feltét szerinti keresés: A felhasználók különböző feltéteket adhatnak meg, például "sajt", "gomba", "szalámi", stb. A rendszer ez alapján keres majd pizzákat, amelyek tartalmazzák a keresett feltéteket. Ez segít azoknak, akik tudják, milyen feltéteket szeretnének, de nem biztosak benne, hogy melyik pizza kínálja azokat.

* Pizza kiválasztása:

Miután a felhasználó megtalálta a számára megfelelő pizzát, képes azonnal rendelni azt, vagy kosárba helyezni, hogy később folytassa a rendelést.

A pizza információi, mint a név, ár, kép és feltétek világosan megjelennek, így a felhasználó könnyedén meghozhatja döntését.

* Kosárba helyezés:

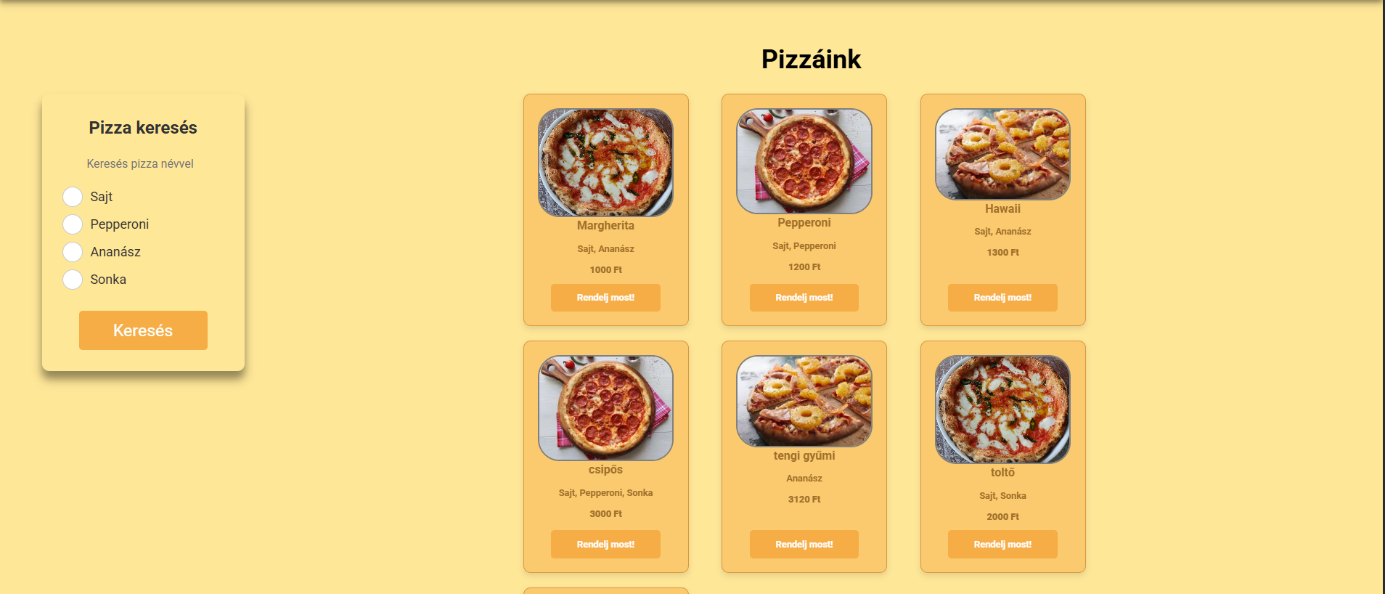
Az oldal lehetővé teszi, hogy a felhasználók egyszerűen kosárba helyezzék a pizzákat, így folytathatják a vásárlást, és a végén egy összegző oldalon láthatják a rendelésük összegét.

A kosárba tett pizzák később szerkeszthetők (pl. módosíthatják a választott feltéteket vagy pizzát), mielőtt a végleges rendelés leadásra kerül.

* Jellemzők:

Felhasználóbarát keresőfelület: A név és feltét szerinti keresés segít a felhasználóknak gyorsan és hatékonyan megtalálni a számukra ideális pizzát, minimalizálva a böngészési időt.

Átlátható információk: A felhasználók világosan láthatják a pizza részletes információit (név, ár, feltétek, kép), ami segít a döntéshozatalban.

Kosár és rendelés: Az oldal integrálja a kosarat, így a felhasználók kényelmesen vásárolhatnak és később befejezhetik a rendelést.

## Rendelés felület

A Rendelés Felület a felhasználók számára biztosítja a lehetőséget, hogy könnyedén rendeljenek pizzát, módosítsák a választott pizzáik számát, méretét, és akár egyedi pizzákat is rendeljenek. Az oldal felhasználóbarát módon van kialakítva, hogy egyszerűsíti a vásárlást, miközben személyre szabott rendeléseket is lehetővé tesz.

* Funkciók:
* Kosárba helyezés:

A felhasználók kiválaszthatják a kívánt pizzát a Menü Oldal segítségével, majd az adott pizzát a kosárba helyezhetik.

A kosárba tett pizzák darabszáma és mérete könnyedén módosítható. Ha a felhasználó többet szeretne rendelni egy adott pizzából, akkor növelheti a mennyiséget, illetve csökkentheti is.

A pizzák mérete is módosítható a rendelés során. A felhasználók választhatnak különböző méretek közül (pl. 32 cm, 40 cm), és ennek megfelelően a végső ár is frissül.

* Egyedi pizza rendelés:

Ha a felhasználó nem talál olyan pizzát, amit szeretne rendelni a menüről, lehetősége van saját, egyedi pizzát létrehozni.

Ezen a felületen a felhasználó kiválaszthatja a kívánt alapot (pl. tészta típus), majd hozzáadhatja a saját választott feltéteket, akár az összes elérhető közül.

A rendelési felület az egyedi pizza készítésekor folyamatosan frissíti az árakat, figyelembe véve az egyes feltétek többletköltségeit.

* Kosár kezelése:

A kosárba tett pizzák tételes listája mindig elérhető. A felhasználó láthatja a pizza nevét, mennyiségét, méretét és árát.

A kosár tartalmát a felhasználó szükség esetén módosíthatja, például eltávolíthat pizzákat, vagy az adott pizza mennyiségét változtathatja.

* Fizetési felület:

Miután a felhasználó befejezte a rendelést, és kosarában minden szükséges pizza megtalálható, rákattinthat a Fizetés gombra.

A fizetési felületen a felhasználó választhat a különböző fizetési lehetőségek közül (pl. bankkártyás fizetés, utánvétes fizetés), és itt láthatja az összesített rendelési költséget.

A fizetési információk megadása után a felhasználó véglegesen leadhatja a rendelését.

* Rendelés visszaigazolása:

A rendelés leadása után a felhasználó visszaigazolást kap, amely tartalmazza a rendelés részleteit (pizzák neve, mennyiség, méret, ár, fizetési mód).

A felhasználó ezen az oldalon is nyomon követheti a rendelés állapotát, hogy mikor kerül kiszállításra.

* Jellemzők:

Felhasználóbarát kezelőfelület: A rendelési felület átlátható és könnyen kezelhető, lehetőséget ad minden felhasználói igény kielégítésére, például pizza módosításokra vagy egyedi pizza készítésére.

Kosár kezelése és fizetés: A kosár részletesen megjeleníti a rendelés összes tételét, a végleges árakkal együtt, míg a fizetési felület biztosítja a kényelmes tranzakciót.

Személyre szabott rendelés: A felhasználók saját ízlésüknek megfelelő pizzákat is rendelhetnek, így nem kell ragaszkodniuk a menüben lévő kínálathoz.

# Összefoglalás

Végleg ebben a pontban lezáró gondolatokat osztunk meg. A project értékelésünk a legutolsó verzión az, hogy habár ez egy elkészült oldal, több aspektusát lehetne fejleszteni és csiszolni, ez természetesen nem egy probléma, hanem egy esély. A projekt készülése idején sokat tanult a csoport vue használatában, node backend használatában, illetve mysql adatbázis tervezést és létrehozást. A program használata a vizsgán kívül természetesen a vendéglátásban látható hiszen a programot azzal az ötlettel hoztuk létre, hogy egy bonyolítsuk le és egyszerűsítsük a felhasználónak.

# Irodalomjegyzék

* Záródolgozat témák, szempontok, ötletek  
  [http://**infojegyzet.hu/**webszerkesztes/zarodolgozat/](http://infojegyzet.hu/webszerkesztes/zarodolgozat/)
* Vue.js  
  <https://vuejs.org/guide/introduction.html>
* Node.js  
  <https://nodejs.org/en>
* Vuetify  
  <https://vuetifyjs.com/en/>
* Dbdiagram, relációs séma  
  <https://dbdiagram.io/>
* Trello, feladatfelosztás  
  <https://trello.com/b/HTyy5zsR/pollakpizza>
* Github, Verziókezelés  
  <https://github.com/>
* Stackoverflow  
  <https://stackoverflow.com/questions>
* Pinia  
  <https://pinia.vuejs.org/>
* Toast  
  <https://getbootstrap.com/docs/5.3/components/toasts/>
* XAMPP  
  <https://www.apachefriends.org/>
* Pizza videó  
  <https://www.youtube.com/watch?v=Y6MHE0-CQAs>
* Figma  
  <https://www.figma.com/design/3cn4MyvbE0Tu59KtHSmrFr/Vizsgaremek?node-id=0-1&p=f>
* Axios  
  <https://www.npmjs.com/package/axios>
* Vite  
  <https://vite.dev/guide/>
* W3schools  
  <https://www.w3schools.com/>
* EJS  
  <https://ejs.co/#install>