**PlateMate**

***Taste the Difference***



**Csoporttagok:**

Barta Máté (Csapatvezető)

Gere Csanád

Bene Dominik

# Tartalomjegyzék

[A szoftver célja 3](#_Toc191895529)

[Kinek és miért? 3](#_Toc191895530)

[Fejlesztői dokumentáció 5](#_Toc191895531)

[Fejlesztőkörnyezet 5](#_Toc191895532)

[Használt IDE-k: 5](#_Toc191895533)

[Technikai Információk 6](#_Toc191895534)

[Adatszerkezetek 8](#_Toc191895535)

[Adatbázis dokumentációja 8](#_Toc191895536)

# A szoftver célja

A PlateMate egy innovatív megoldás, amely segíti az éttermeket a modernizálásban és a digitális átalakulásban. A rendszer célja, hogy egyszerűsítse és optimalizálja az éttermi működést, miközben kényelmes és gördülékeny élményt nyújt a vendégek számára. A szoftver segítségével az éttermek lépést tarthatnak a modern fogyasztói elvárásokkal, minimalizálva a manuális folyamatokat és csökkentve a hibalehetőségeket.

A PlateMate lehetővé teszi a gyorsabb és hatékonyabb kiszolgálást, csökkentve a várakozási időt és növelve az ügyfél elégedettségét. Az intelligens rendeléskezelési rendszernek köszönhetően a személyzet könnyebben és átláthatóbban tudja kezelni a rendeléseket, amely gyorsabb kiszolgálást és precízebb munkavégzést eredményez. Az automatizált folyamatok révén csökkenthető a személyzeti túlterheltség, és optimalizálható a munkafolyamatok hatékonysága.

A szoftver egy felhasználóbarát, akár mobilról is elérhető kezelőfelületet biztosít, amely lehetővé teszi a tulajdonosok és alkalmazottak számára, hogy a nap bármely pillanatában nyomon kövessék és módosítsák az éttermi működéshez kapcsolódó adatokat. Az élő frissítéseknek köszönhetően az étlapon vagy a foglalások kezelésében végzett módosítások azonnal érvénybe lépnek, megkönnyítve az üzleti folyamatok irányítását. A PlateMate telepítés után gyorsan és hatékonyan integrálható az étterem mindennapi működésébe, lehetővé téve a testreszabott beállításokat és a folyamatos fejlesztéseket.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Kinek és miért?

A PlateMate minden olyan étterem számára ajánlott, amely jelenleg nem rendelkezik digitális megoldásokkal, vagy elavult rendszereket használ. Az étterem tulajdonosok számára a szoftver lehetőséget nyújt a működés optimalizálására, az asztalfoglalások és rendelések automatizálására, valamint az ügyfélkapcsolatok hatékonyabb kezelésére. Ezáltal jelentős idő- és erőforrás-megtakarítást eredményez, miközben növeli az ügyfélélményt.

A rendszer előnyei az alkalmazottak számára is jelentősek. A rendelési folyamatok könnyebben követhetők, csökkentve az emberi hibák esélyét, és gyorsabb, pontosabb kiszolgálást biztosítva. Az étterem vezetősége könnyedén ellenőrizheti az üzlet forgalmát, a népszerűbb ételeket és az ügyfélpreferenciákat, így jobban alakíthatja az üzleti stratégiát és növelheti a profitabilitást.

A PlateMate a vendégek számára is számos előnnyel jár. Az okostelefonról elérhető funkciók lehetővé teszik az előzetes rendeléseket, gyors asztalfoglalásokat és az étlap böngészését, így az étkezési élmény gördülékenyebbé és kényelmesebbé válik. Az automatizált fizetési rendszer csökkenti a várakozási időt, és lehetőséget biztosít az érintésmentes fizetésre, amely különösen fontos a modern digitális korszakban.

A PlateMate nemcsak egy egyszerű rendeléskezelő rendszer, hanem egy átfogó, digitális megoldás, amely elősegíti az éttermek versenyképességének növelését, a hatékonyság javítását és az ügyfélélmény fejlesztését. A technológia alkalmazásával az éttermek képesek megfelelni a folyamatosan változó piaci igényeknek, és hosszú távon is sikeresen működni a vendéglátóiparban.

# Fejlesztői dokumentáció

A fejlesztői dokumentáció magába foglalja az összes eszközt, ami használva lett a projekt megvalósításához, illetve leírja: a program telepítését, annak futási környezetét, a kialakított adatbázis és backend adatszerkezeteket, bemutat néhány metódust a programból, leírja a program tesztelését, és kínál néhány fejlesztési lehetőséget a programhoz, amit a későbbiekben meg lehet még valósítani a program bővítéséhez.

## Fejlesztőkörnyezet

A fejlesztőkörnyezet tartalmazza azon eszközöket, programokat, struktúrákat, amelyek használva voltak a PlateMate megírásához, és így hozzájárultak/hozzájárulnak a program kiváló működéséhez. Ezen ponton szeretnénk megjegyezni, hogy ***a PlateMate csapata semmilyen felelősséget nem vállal a program értékelése során tapasztalt olyan problémákért, hibákért, amelyek a program ellenőrzése közben abból fakadnak***, hogy:

* Az applikáció komponenseit nem a dokumentációban megadott verzióval futtatják a programot (pl: MySQL, Xampp, NPM, NODE, VUE, TailwindCSS, Express) Ezek csupán példák az applikáció komponenseire, a részletes komponensek listáját lejjebb találják.
* Hibásan, nem megfelelően, vagy elavult applikáció verziót töltöttek le a megadott GitHub Repository-ból.
* Hibásan, vagy nem megfelelően lettek beimportálva a megadott adatok, file-ok.
* Hibásan, vagy nem megfelelően lett felépítve a .env file.
* Másik operációs rendszer verzión próbálták meg futtatni a programot.
* Másik operációs rendszeren próbálták meg futtatni a programot.
* Nem MySQL-t használtak Xampp-on keresztül az adatbázis futtatásához.

### Használt IDE-k:

#### **Visual Studio Code (VS Code)**

A Visual Studio Code (VS Code) egy ingyenes, nyílt forráskódú, könnyű, mégis erőteljes fejlesztői környezet, amelyet a Microsoft fejlesztett. A VS Code támogatja a JavaScript, Vue.js, Node.js és számos más programozási nyelv fejlesztését, így tökéletes választás egy full stack alkalmazás létrehozásához. Az intelligens kódkiegészítés (IntelliSense), a beépített Git-integráció és a terminál támogatás lehetővé teszik a fejlesztési folyamat gyorsítását és hatékony kezelését. Az eszköz támogatja a könnyen telepíthető kiegészítőket (extensions), amelyek lehetővé teszik a fejlesztők számára a környezet testre szabását, például a Vue.js DevTools vagy az ESLint használatával.

A VS Code emellett integrált debugging eszközzel is rendelkezik, amely lehetővé teszi a Node.js és a böngésző alapú hibakeresést. A Live Share funkció lehetővé teszi a valós idejű együttműködést, ami különösen hasznos csapatmunkánál. A könnyen kezelhető felület és a gyors teljesítmény miatt a VS Code az egyik legnépszerűbb fejlesztői eszköz, amely jól működik bármilyen operációs rendszeren, beleértve a Windows, macOS és Linux rendszereket is.

#### **JetBrains WebStorm**

A WebStorm egy prémium fejlesztői környezet, amelyet kifejezetten a modern JavaScript-alapú fejlesztéshez terveztek. A JetBrains által fejlesztett WebStorm mély integrációt kínál a Vue.js, Node.js, Express és más népszerű webes technológiák számára. Az intelligens kódkiegészítés, refaktorálási lehetőségek és fejlett hibakereső eszközök révén a WebStorm különösen hasznos nagyobb, komplex projektek esetében, ahol a kód minősége és karbantarthatósága kulcsfontosságú. Az IDE beépített támogatást nyújt a verziókezelő rendszerekhez, például Git, GitHub és GitLab, ami megkönnyíti a csapatmunkát és a verziókezelést.

A WebStorm egyik kiemelkedő előnye a beépített tesztelési támogatás, amely lehetővé teszi az egység- és integrációs tesztek futtatását közvetlenül a fejlesztői környezetből. A beépített REST kliens és adatbázis-kezelő eszközök segítenek a szerveroldali fejlesztések hatékony tesztelésében. Habár a WebStorm fizetős szoftver, sok fejlesztő számára megéri a befektetést a produktivitás növelése érdekében, különösen akkor, ha napi szinten dolgoznak JavaScript és TypeScript alapú alkalmazásokon.

### Technikai Információk

#### **Operációs Rendszer: Windows 10/11**

A fejlesztésre a Windows 10 és 11 operációs rendszerek használata ideális választás, mivel széles körben támogatottak és kompatibilisek a legtöbb modern fejlesztői eszközzel. A Windows környezet lehetőséget biztosít a fejlesztők számára, hogy könnyedén telepítsenek és konfiguráljanak különböző futtatókörnyezeteket, mint például a Node.js, MySQL és XAMPP. Továbbá, olyan népszerű fejlesztői eszközök is rendelkezésre állnak, mint a Visual Studio Code, amely beépített támogatást nyújt a JavaScript és a Vue.js számára, valamint a Webstorm. A Windows támogatja a WSL (Windows Subsystem for Linux) használatát is, amely lehetővé teszi a fejlesztők számára, hogy Linux-alapú parancssori eszközöket használjanak egy Windows rendszeren belül. A széleskörű hardver- és szoftverkompatibilitás miatt a Windows megbízható választás minden fejlesztési szakaszban.

#### **Front End**

A felhasználói felület fejlesztése során az alábbi technológiákat használjuk:

* **HTML** – A HyperText Markup Language (HTML) az alkalmazás szerkezeti alapját biztosítja. Lehetővé teszi az oldal különböző elemeinek meghatározását és a felhasználói interfész felépítését. Az HTML folyamatosan fejlődő szabványai biztosítják a kompatibilitást a modern böngészőkkel, valamint támogatják az akadálymentességet és a keresőmotor-optimalizálást (SEO). Az alaposan strukturált HTML-kód hozzájárul a könnyebb karbantarthatósághoz és a reszponzív dizájn kialakításához.
* **CSS** – A Cascading Style Sheets (CSS) segítségével formázzuk az oldal kinézetét, biztosítva a modern, reszponzív és esztétikus megjelenést. A CSS lehetővé teszi az animációk és interaktív elemek beépítését, növelve ezzel a felhasználói élményt. Az előre definiált stílusok és CSS preprocesszorok, mint például a SASS vagy LESS, tovább javíthatják a kód szervezhetőségét és hatékonyságát.
* **Vue.js (v5.0.8)** – A Vue.js egy könnyen tanulható és hatékony JavaScript keretrendszer, amely segít a felhasználói interfész dinamikus kezelésében. A Vue 5.0.8 verzió számos teljesítménybeli fejlesztést tartalmaz, valamint támogatja a legújabb modern webes szabványokat. A Vue ökoszisztémája folyamatosan bővül, így könnyedén integrálható Vuex vagy Pinia állapotkezelő, valamint Vue Router az útvonalkezeléshez. A reaktív adatkezelés és a komponensalapú architektúra segít a fejlesztőknek hatékony és jól skálázható alkalmazásokat létrehozni.
* **Tailwind CSS (v3.4.13)** – Egy utility-first CSS keretrendszer, amely lehetővé teszi az egyedi stílusok gyors alkalmazását anélkül, hogy külső CSS fájlokat kellene írni. A Tailwind 3.4.13 verziója optimalizált teljesítményt és testre szabható konfigurációs lehetőségeket kínál. Az előre definiált utility osztályok használatával gyorsítható a fejlesztés, miközben a kód továbbra is tiszta és jól olvasható marad. A Tailwind segítségével könnyedén valósíthatók meg reszponzív és modern webes felületek anélkül, hogy hosszadalmas egyedi CSS szabályokat kellene írni.
* **NPM (v10.9.0)** – A Node Package Manager (NPM) egy csomagkezelő, amely lehetővé teszi a szükséges front-end függőségek kezelését és telepítését. Az NPM segítségével könnyedén integrálhatók külső könyvtárak és eszközök a fejlesztési környezetbe. A gyors verziókezelésnek köszönhetően mindig naprakészen tarthatók a projekt függőségei, minimalizálva a kompatibilitási problémákat.

#### **Back End**

A szerveroldali logika és az API-k fejlesztése a következő technológiákkal történik:

* **JavaScript** – A teljes stack fejlesztést megkönnyítve egyetlen programozási nyelvet használunk mind a front-end, mind a back-end fejlesztésére. A JavaScript dinamikus, könnyen skálázható, és széles körű támogatással rendelkezik. A Node.js környezetben futtatva lehetővé teszi aszinkron műveletek gyors végrehajtását, amely jelentős teljesítménybeli előnyt nyújt a hagyományos szinkron rendszerekkel szemben.
* **Node.js (v20.17.0)** – A Node.js egy gyors, aszinkron működésű JavaScript futtatókörnyezet, amely lehetővé teszi szerveroldali alkalmazások fejlesztését. A 20.17.0 verzió fejlett teljesítményoptimalizálásokat és stabilitást biztosít. A V8 motorra épülve hatékonyan kezeli a nagy terhelésű, real-time adatfeldolgozási feladatokat, így ideális választás webszerverekhez és API-khoz.
* **Express.js (v4.21.1)** – Az Express.js egy minimalista és rugalmas Node.js keretrendszer, amely segít az API-k és szerveroldali alkalmazások gyors fejlesztésében. A 4.21.1 verzió biztosítja a legújabb hibajavításokat és kompatibilitást a Node.js újabb verzióival. Az Express könnyen integrálható middleware-ekkel és adatbázis-kezelőkkel, ami tovább növeli a fejlesztés hatékonyságát.
* **NPM (v10.9.0)** – A backend függőségek kezelése és automatizálása NPM segítségével történik, lehetővé téve az Express.js és egyéb csomagok egyszerű telepítését és frissítését. Az NPM CLI (parancssoros felület) segítségével gyorsan konfigurálhatók a projektek, és biztosítható a stabil működés.

#### **Adatbázis**

Az alkalmazás adatkezelése és tárolása MySQL alapú adatbázis segítségével történik:

* **XAMPP Control Panel (v3.3.0)** – Egy könnyen használható helyi fejlesztői környezet, amely tartalmazza az Apache webszervert és a MySQL/MariaDB adatbáziskezelőt. A XAMPP használata gyors beállítást és kényelmes adatbázis-kezelést biztosít. A lokális fejlesztési környezet lehetővé teszi az adatbázis-struktúra gyors tesztelését és módosítását, mielőtt az éles szerverre kerülne.
* **MySQL (10.4.32-MariaDB)** – A MariaDB a MySQL egyik legnépszerűbb nyílt forráskódú változata, amely kiváló teljesítményt, megbízhatóságot és skálázhatóságot kínál. A 10.4.32 verzió stabil és optimalizált működést biztosít az adatok gyors és biztonságos kezelésére. Az ACID kompatibilitásnak és fejlett indexelési mechanizmusoknak köszönhetően hatékonyan kezeli a nagy mennyiségű adatot és összetett lekérdezéseket.

#### Hardveres igények

* **CPU**
  + Intel Core i3-7100 3.9GHz vagy jobb, vagy
  + AMD Ryzen 3 1200 3.1GHz vagy jobb
* **GPU**
  + Intel HD Graphics 5500 vagy jobb, vagy
  + AMD Radeon Graphics (Ryzen 7000) vagy jobb
* **RAM**: 4GB of 1600Mhz vagy jobb
* **Tárhely**: ~50MB szabad tárhely

## Adatszerkezetek

Ebben a fejezetben az applikációhoz használt adatszerkezetek kerülnek bemutatásra.

### Adatbázis dokumentációja

A vizsgaremek adatbázis az éttermi rendelési rendszer támogatására készült. A rendszer a rendelések, fizetések, foglalások és felhasználók kezelésére szolgál. Az alábbiakban részletezzük az adatbázis szerkezetét és a benne található táblákat.

**Táblák és oszlopok**

**1. category (Kategóriák)**

Tárolja az ételek kategóriáit (pl. előétel, főétel, desszert).

* id (int): Egyedi azonosító.
* (255): A kategória neve.

**2. item (Ételek és italok)**

Tárolja az étlapon szereplő tételeket.

* id (int): Egyedi azonosító.
* name (varchar (255)): Az étel vagy ital neve.
* price (int): Az étel vagy ital ára.
* categoryId (int): A kategória azonosítója (category.id kapcsolódó mező).

**3. openinghours (Nyitvatartás)**

Tárolja az étterem nyitvatartási adatait.

* id (int): Egyedi azonosító.
* dayName (varchar (255)): A hét napja.
* fromHour (varchar (255)): Nyitási idő.
* untilHour (varchar (255)): Zárási idő.

**4. orders (Rendelések)**

Tárolja a leadott rendeléseket.

* id (int): Egyedi azonosító.
* tableId (int): Asztal azonosítója (tables.id kapcsolódó mező).
* itemId (int): Rendelési tétel azonosítója (item.id kapcsolódó mező).
* isDone (tinyint (1)): Kész van-e a rendelés (0 = nem, 1 = igen).
* isServed (tinyint (1)): Kiszolgálták-e a rendelést (0 = nem, 1 = igen).
* orderedAt (datetime): Rendelés időpontja.

**5. paid (Kifizetések)**

Tárolja a kifizetett rendeléseket.

* id (int): Egyedi azonosító.
* tableId (int): Asztal azonosítója.
* itemId (int): Rendelési tétel azonosítója.
* paymentMethodId (int): Fizetési mód azonosítója (paymentmethods.id kapcsolódó mező).
* paidAt (datetime): Kifizetés időpontja.

**6. paymentmethods (Fizetési módok)**

Tárolja az elérhető fizetési módokat.

* id (int): Egyedi azonosító.
* name (varchar (255)): Fizetési mód neve (pl. készpénz, bankkártya).

**7. permissionsettings (Jogosultságok)**

Tárolja a különböző jogosultsági szinteket.

* id (int): Egyedi azonosító.
* section (varchar (255)): Jogosultsági szekció neve.

**8. reservedtable (Foglalások)**

Tárolja az asztalfoglalásokat.

* id (int): Egyedi azonosító.
* name (varchar (255)): Foglaló neve.
* numberOfCustomers (int): Foglalásban részt vevő vendégek száma.
* tableId (int): Asztal azonosítója.
* reservedAt (datetime): Foglalás időpontja.
* reservedUntil (datetime): Foglalás lejárati ideje.

**9. sessions (Munkamenetek)**

Tárolja a felhasználók aktív munkameneteit.

* id (varchar (255)): Egyedi azonosító.
* userId (int): Felhasználó azonosítója (user.id kapcsolódó mező).
* ip (varchar (255)): Felhasználó IP címe.
* expires (datetime): Munkamenet lejárati ideje.

**10. tables (Asztalok)**

Tárolja az étterem asztalait.

* id (int): Egyedi azonosító.
* tableNumber (int): Asztal száma.

**11. user (Felhasználók)**

Tárolja az éttermi rendszerben regisztrált felhasználókat.

* id (int): Egyedi azonosító.
* name (varchar (255)): Felhasználó neve.
* email (varchar (255)): Felhasználó e-mail címe.
* hashedPassword (varchar (255)): Jelszó hash-elve.
* permissionId (int): Jogosultsági szint (permissionsettings.id kapcsolódó mező).
* createdAt (datetime): Fiók létrehozásának időpontja.

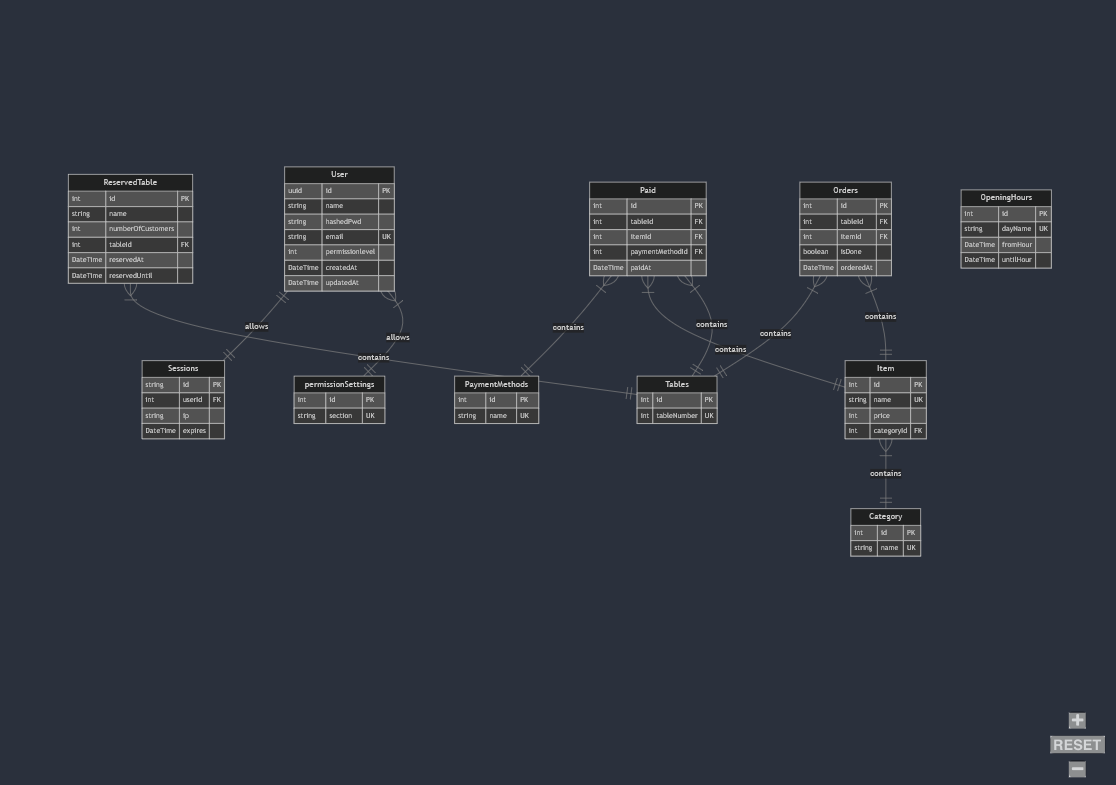
**Kapcsolatok az adatbázisban**

* Az item tábla a category táblára hivatkozik.
* Az orders és paid táblák az item, tables és paymentmethods táblákkal kapcsolódnak.
* A reservedtable és orders táblák az tables táblával állnak kapcsolatban.
* A user táblában lévő jogosultságokat a permissionsettings táblában tároljuk.

Ez az adatbázis-struktúra biztosítja az éttermi rendelési rendszer hatékony működését, lehetővé téve a rendelések, fizetések és foglalások pontos kezelését.

### ER (Entity Relationship) Diagramm

Ez az ER-diagram egy éttermi rendelési rendszert modellez. A **User** entitás kezeli a felhasználókat, jogosultságokkal és munkamenetekkel kapcsolódva. Az **Orders** és **Paid** táblák követik a rendeléseket és a fizetéseket, összekötve az **Item**, **Tables**, és **PaymentMethods** entitásokkal. Az **Item** kategóriákba sorolható, és a **ReservedTable** az asztalfoglalásokat kezeli. Az **OpeningHours** tárolja a nyitvatartást. Az adatmodell biztosítja az éttermi rendelési és foglalási folyamatok átlátható kezelését.

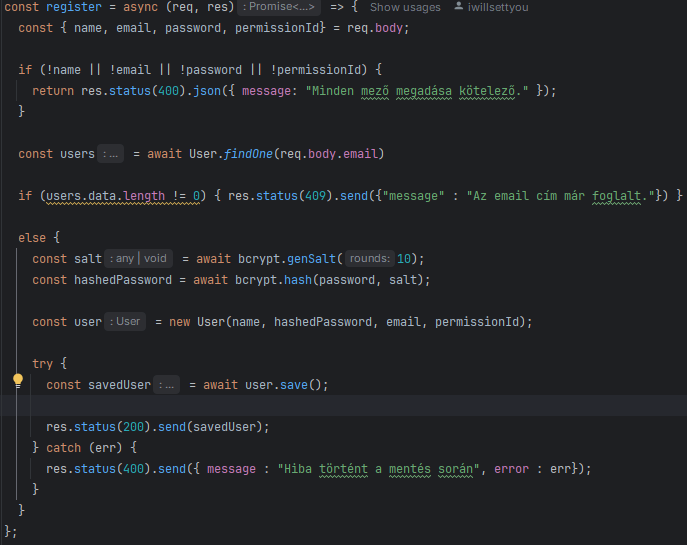


## Egyedi algoritmusok

Ebben a fejezetben egy-két érdekesebb, egyedi algoritmus van bemutatva. Ezek az algoritmusok szerves részét képezik az applikációnak, felépítésük a programozási alapelveknek megfelelően lett létrehozva, kialakítva, és implementálva. A továbbiakban képeket láthatnak a kódrészletről, illetve rövid leírást, hogy hogyan működik a program és mi a célja.

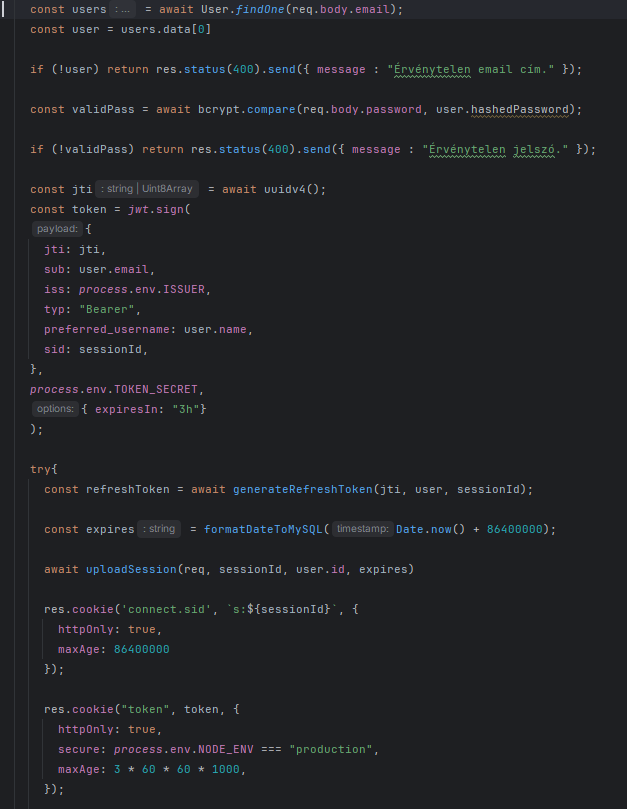
### authController.js

A ***register*** függvény egy új felhasználót hoz létre, ellenőrzi, hogy az e-mail cím már foglalt-e, majd a jelszót biztonságosan hashelve menti az adatbázisba.



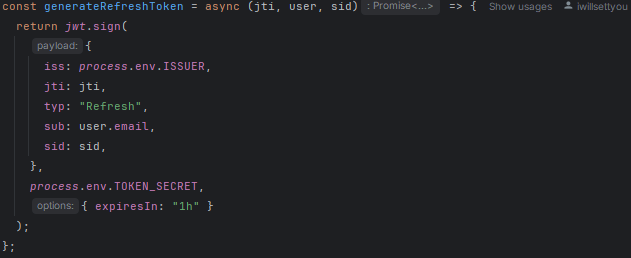
A ***login*** függvény először ellenőrzi, hogy van-e érvényes munkamenet, és ha nincs, akkor a felhasználó hitelesítő adatait ellenőrzi, JSON Web Tokent generál számára, és azt sütikben tárolja.



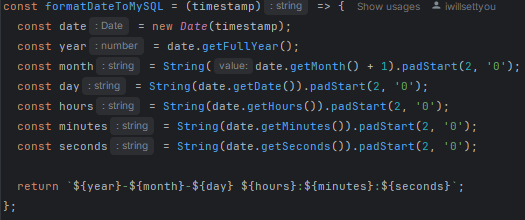




A ***logout*** függvény törli a felhasználó munkamenetét, míg a generateRefreshToken egy új frissítő tokent generál, amely lehetővé teszi a hozzáférési token megújítását anélkül, hogy újra be kellene jelentkezni.



A ***formatDateToMySQL*** segédfüggvény az időbélyegeket MySQL-kompatibilis formátumba alakítja.



## Tesztdokumentáció

Ebben a fejezetben néhány teszt kerül bemutatásra, mind a Backend-ből, mind a Frontend-ből.A Jest és Supertest kombinációja ideális választás a JavaScript backend teszteléséhez, mivel mindkét eszköz erőteljes funkcionalitást és egyszerűsített tesztelési folyamatot kínál. A Jest egy könnyen konfigurálható és gyors tesztelő keretrendszer, amely lehetővé teszi a szinkron és aszinkron tesztek egyszerű írását, miközben beépített eszközöket biztosít a mockoláshoz, kémkedéshez és időzítők kezeléséhez.

A Supertest pedig kifejezetten HTTP-kérések küldésére és válaszok tesztelésére lett tervezve, így ideális választás API végpontok tesztelésére. A két eszköz együtt biztosítja, hogy könnyedén írhatunk integrációs teszteket, amelyek ellenőrzik a backend különböző komponenseinek együttműködését, miközben a tesztek jól olvashatóak és gyorsan futtathatóak maradnak. Ezen kívül a Jest részletes tesztjelentései és a Supertest könnyed HTTP kérés-képességei gyorsabb hibakeresést és fejlesztést tesznek lehetővé, így biztosítva a hatékony és robusztus backend fejlesztést.

### openingHourController.test.js

Ez a fájl az **Opening Hour API** tesztelésére szolgál a **Jest** és **Supertest** segítségével. A tesztkörnyezet beállítására, az API végpontjainak működésének ellenőrzésére, valamint az adatbázis inicializálására és tisztítására szolgál.

**Módszertan és Módszerek**

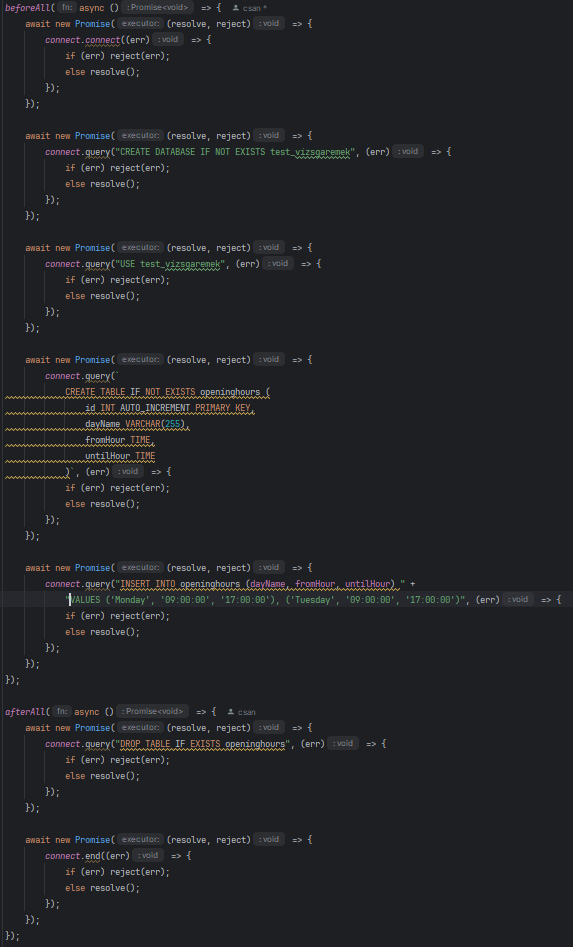
* **Supertest és Jest**: A tesztelés során a Supertest könyvtárat használjuk az API végpontok kérésének küldésére és az HTTP válaszok ellenőrzésére. A Jest segítségével pedig a teszteket futtatjuk és validáljuk az eredményeket.
* **Aszinkron tesztelés:** Mivel az API végpontok aszinkron műveleteket végeznek (adatbázis műveletek, HTTP kérések), a tesztek aszinkron módon lettek megírva, és a async/await szintaxist alkalmazzuk a megfelelő válaszok megvárására és az ellenőrzések végrehajtására.
* **Tesztelési scenáriók:** A tesztek minden fontos forgatókönyvet lefednek, beleértve a sikeres nyitvatartási idő létrehozását, lekérését és törlését, biztosítva ezzel, hogy a Controller minden funkciója megfelelően működjön.

**Tesztkörnyezet és Beállítások**

* **Kérések küldése:** A tesztkészlet POST, GET, és DELETE HTTP metódusokkal dolgozik az openingHours végpontokkal.
* **Válaszok ellenőrzése:** Minden teszt biztosítja, hogy a válaszok a megfelelő státuszkódot tartalmazzák, és a válaszban a szükséges adatok, például az insertId vagy a törlés üzenete megtalálhatók.
* **Hibaellenőrzés:** Ha a válasz nem a várt státuszkóddal érkezik, a tesztek hibaüzenetet generálnak, amely segít a hiba forrásának azonosításában.

**Tesztek előkészítése**

* **Adatbázis inicializálás (beforeAll)**
  + Kapcsolódik az adatbázishoz.
  + Létrehozza a test\_vizsgaremek tesztadatbázist és az openinghours táblát.
  + Előre beállított nyitvatartási adatokkal tölti fel az adatbázist.
* **Adatbázis tisztítása (afterAll)**
  + A tesztek lefutása után törli az openinghours táblát.
  + Megszünteti az adatbázis-kapcsolatot.



**Főbb Funkciók és Tesztelt Funkciók**

* **POST /api/openingHours –** Új nyitvatartási idő létrehozása
  + **Sikeres nyitvatartási idő létrehozása:** A teszt ellenőrzi, hogy a rendszer képes új nyitvatartási idő hozzáadására egy POST kéréssel, és helyes választ ad. A nyitvatartási idő tartalmazza a napot, az induló és befejező időpontot.
    - **Várt válasz:** 200 OK státuszkód, valamint a válaszban lévő data objektumban egy insertId mező.
* **GET /api/openingHours –** Minden nyitvatartási idő lekérése
  + **Nyitvatartási idő lekérése:** A teszt azt ellenőrzi, hogy a rendszer sikeresen visszaadja az összes nyitvatartási időt egy GET kéréssel.
    - **Várt válasz:** 200 OK státuszkód, valamint a válaszban lévő data objektumban található nyitvatartási idők listája.
* **DELETE /api/openingHours/:id –** Nyitvatartási idő törlése az adott ID alapján
  + **Sikeres törlés:** A teszt biztosítja, hogy a rendszer képes törölni egy adott nyitvatartási időt az id alapján, és a válasz megfelelő üzenetet küld.
    - **Várt válasz:** 200 OK státuszkód, valamint a válaszban lévő message mezőben szerepelnie kell a "Időpont sikeresen törölve." üzenetnek.

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

**Következtetés**

Ez a tesztkészlet biztosítja, hogy az Opening Hour Controller API végpontjai, amelyek a nyitvatartási időket kezelik, helyesen működnek. A tesztek figyelmet fordítanak a sikeres műveletek mellett a válaszok épségére, így biztosítva, hogy minden API végpont helyesen kezeli a nyitvatartási idők létrehozását, lekérését és törlését. Az izolált tesztelés és az aszinkron működés biztosítja, hogy a tesztkészlet megbízható és gyors legyen.

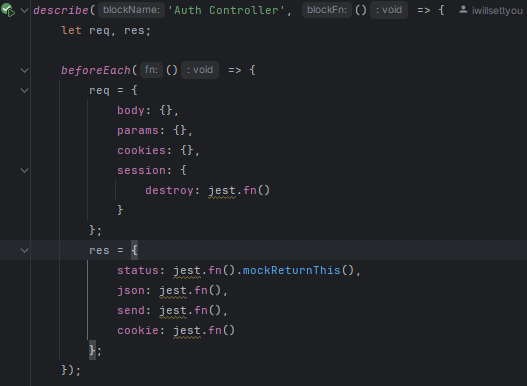
### authController.test.js

Ez a fájl a **Jest** és a **mocking** technikák segítségével teszteli az **authController** működését, amely a felhasználói regisztrációt, bejelentkezést és kijelentkezést kezeli. A tesztkörnyezet beállítása során a User modellt, a bcrypt jelszókezelő könyvtárat, a jsonwebtoken token generálót, az uuid generátort és a sessionHandler middleware-t **mock**-oljuk, hogy a tesztek izoláltak legyenek, és a tényleges adatbázis vagy külső szolgáltatások ne befolyásolják őket.

**Módszertan és Módszerek**

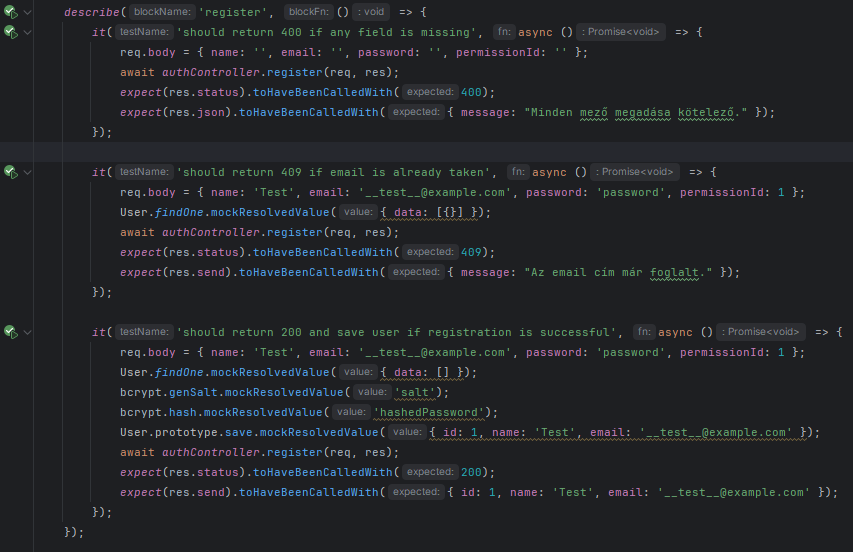
* **Mocking:** Az összes külső függőség (például User, bcrypt, jsonwebtoken, uuid, sessionHandler) mock-olása biztosítja, hogy a tesztek izoláltak legyenek, és ne függjenek az adatbázistól vagy más külső rendszerektől.
* **Jest és Supertest:** A tesztek a **Jest** keretrendszer segítségével készülnek, amely biztosítja a könnyű tesztelést és az aszinkron kód kezelését. A **Supertest** és az integrációs tesztek segítségével az API végpontok viselkedése ellenőrizhető.
* **Tesztelési scenáriók:** A tesztek minden fontos forgatókönyvet lefednek, beleértve a hibakezelést, sikeres regisztrációt és bejelentkezést, valamint a session kezelést.

A mock-olás bemutatása:



**Főbb Funkciók és Tesztelt Funkciók**

* **register funkció tesztelése**
  + **Hiányzó mezők ellenőrzése:** A teszt ellenőrzi, hogy ha bármelyik kötelező mező (pl. név, email, jelszó) hiányzik, akkor a válasz 400 státuszkóddal és megfelelő hibaüzenettel tér vissza.
  + **Már regisztrált email:** Ha a regisztrált email már létezik az adatbázisban, akkor 409 státuszkóddal és a "email már foglalt" üzenettel válaszol.
  + **Sikeres regisztráció:** Ha minden adat helyes, a teszt ellenőrzi, hogy a felhasználó sikeresen regisztrálódik, és a megfelelő adatokat (felhasználó id, név, email) kapja vissza.



* **login funkció tesztelése**
  + **Session ellenőrzés:** A teszt figyeli, hogy ha a session ellenőrzés nem sikerül, akkor a válasz 400 státuszkóddal és "Session check failed" üzenettel érkezik.
  + **Érvénytelen email vagy jelszó:** Ha az email nem található, vagy a megadott jelszó nem egyezik a mentett jelszóval, a rendszer megfelelő 400 státuszkóddal és a "Érvénytelen email cím" vagy "Érvénytelen jelszó" üzenetekkel válaszol.
  + **Sikeres bejelentkezés:** Sikeres bejelentkezés esetén a teszt ellenőrzi, hogy a rendszer helyesen generálja le a JWT token-t és a megfelelő cookie-kat beállítja.

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

* **logout funkció tesztelése**
  + **Sikeres kijelentkezés:** A teszt biztosítja, hogy ha a kijelentkezés sikeresen végrehajtódik, akkor a megfelelő státuszkóddal és üzenettel válaszol.
  + **Hiba a kijelentkezés során:** Ha valamilyen hiba lép fel a session törlése közben, a válasz 500 státuszkóddal és a hiba részletezésével érkezik.

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Ez a tesztkészlet biztosítja, hogy az **authController** minden funkciója helyesen működik, figyelve a felhasználói regisztráció, bejelentkezés, és kijelentkezés megfelelő működésére, miközben ellenőrzi a hibakezelést és az adatbázis-függőségeket. A mockolt komponensek használata biztosítja a tesztek izoláltságát, gyorsaságát és megbízhatóságát.

## Fejlesztési lehetőségek

Jelenlegi éttermi rendszerünk egyik legfontosabb hiányossága, hogy a foglalási oldalon nincs lehetőség az asztalok helyének módosítására. A foglalásokat jelenleg az asztalok fix helyeihez rendeljük, de ez nem biztosít kellő rugalmasságot a rendszer használói számára. Például előfordulhat, hogy egy vendég szeretne egy nagyobb asztalt, amely több szék köré van elrendezve, de mivel az asztalok elhelyezkedése nem módosítható a rendszerben, ez nem lehetséges. Ez különösen problémás lehet csúcsidőszakokban, amikor több foglalás is egyes asztalokhoz van rendelve, és a rendelkezésre álló helyek nem biztosítanak elegendő rugalmasságot a vendégek igényeinek megfelelően.

A jelenlegi rendszer nem veszi figyelembe azt a fontos igényt sem, hogy az asztalok helyét az étterem tulajdonosa, vagy a foglalási menedzser könnyen át tudja alakítani, hogy a vendégek számára ideális környezetet biztosítson. Az asztalok mozgathatóságának bevezetése lehetővé tenné az étterem számára, hogy rugalmasabban kezelje a helyeket, és jobban igazodjon a vendégek igényeihez.

A rendszerünk másik fejlesztési lehetősége a felhasználói szerepkörök rugalmasabb kezelése. Jelenleg csak négy féle felhasználói típus van bevezetve: adminisztrátor, szakács, pincér és kasszás. Ezen felhasználói típusok mindegyike előre meghatározott jogosultságokkal rendelkezik, de a különböző szerepkörök nincsenek kellően testre szabva a változó üzleti igényekhez. Az adminisztrátor jogosultságaival például túl széleskörű hozzáférést ad, míg a pincérek és szakácsok jogosultságai túlzottan korlátozottak lehetnek, ha valaki például az étlapot vagy egyéb operatív adatokat szeretne módosítani.

Ez a korlátozottság különösen akkor válik problémává, amikor egy nagyobb étterem vagy lánc operációit kell támogatni, ahol a szerepkörök között több szintű, finomabb jogosultságokra lenne szükség. Például a pincérek jogosultságait a rendszerben jobban differenciálhatnánk, hogy csak azokat a műveleteket hajthassák végre, amelyek szükségesek számukra, miközben a szakácsok számára más típusú jogosultságokat adnánk.

Továbbá az adminisztrátor szerepkörön belül is különböző altípusok bevezetésére lenne lehetőség, például egy adminisztrátor, aki kizárólag a rendszer felhasználóit kezeli, míg egy másik adminisztrátor csak az étterem pénzügyi adataihoz férhet hozzá. A szerepkörök testreszabásának fejlesztése jelentősen javítaná a rendszer rugalmasságát, és jobb biztonságot biztosítana a különböző típusú felhasználók számára.

Ezen fejlesztési lehetőségek bevezetése nemcsak az étterem napi működését könnyítené meg, hanem a felhasználói élményt is jelentősen javítaná. A foglalási rendszerben az asztalok mozgathatóságának biztosítása lehetővé tenné a gyors alkalmazkodást a különböző vendégcsoportok igényeihez, így az étterem alkalmazottai könnyebben kezelhetnék a helyeket és biztosíthatnák a vendégek kényelmét. A felhasználói szerepkörök testre szabása pedig segítené az étterem vezetőségét abban, hogy jobban kontrollálhassa a hozzáféréseket és biztonságot, miközben az operatív munkafolyamatokat is rugalmasabban kezelhetné. Ezáltal a rendszer sokkal skálázhatóbbá válna, és a jövőbeni növekedésre is felkészíthetné az étterem működését.

# Felhasználói dokumentáció

Ebben a fejezetben a felhasználót segítő információk közlése lesz a fő szempont, egészen a telepítéstől, a használaton keresztül a helytelen használatból adódott hibák elhárításáig. Többek között bemutatjuk a felhasználók, rendelések, fizetések és foglalások kezelését, valamint a jogosultságokat és a munkameneteket A dokumentáció lépésről lépésre ismerteti a funkciókat, biztosítva a gördülékeny működést és a felhasználói élmény optimalizálását.

## A program funkciói

A **PlateMate** éttermi rendszer célja, hogy a programot használó éttermeknek egy gördülékeny, könnyű használatot nyújtson az éttermi rendelések, foglalások, számla rendezések lebonyolításában. A program többek között képes: foglalásokat kezelni, rendeléseket felvenni az adott számú asztalhoz, egyidőben (real-time) megjeleníteni az éppen felvett rendeléseket a konyha számára, majd kiszolgálás után az összes elfogyasztott terméket asztal szerint kifizetésre bocsájtani az étterem által megadott ár lista szerint.

A program lehetőséget nyújt 4 féle munkakört ellátó alkalmazottaknak fiók létrehozásra, így lehetővé téve a párhuzamos munkavégzést akár egy nagyobb méretű étterem számára is.

**A 4 munkakör:**

* **Adminisztrátor**: Mindent elér, amit a többi munkakörrel el lehet érni, tudja nézni, módosítani, törölni.
* **Séf**: Kizárólag a séfek munkaköréhez tartozó

**Tartalom**

4 féle fióknak ad használatra lehetőséget a program, ezekre a fajt fiókokra mostantól, mint „rang” lesz hivatkozva:

* Admin
  + Mindent elér, amit a többi ranggal el lehet érni, tudja nézni, módosítani stb.
* Séf
  + Csak a séfekhez tartozó route-okat éri el, vagyis a:
    - Rendeléseket
* Pincér
  + Csak a pincérekhez tartozó rouet-okat éri el, vagyis a(z):
    - Étel kategóriákat,
    - Étel fajtákat,
    - Nyitvatartási időt,
    - Rendeléseket,
    - Asztal foglalásokat
* Szakács

**Kezelőfelület**

Az étterem vezetőinek szánt felület ahol kezelni tudják az étterembe található ételeket és azoknak az árukat. Lehetőséget nyújt X időintervallumba tekintő visszamenőleges ellenőrzésre melyben megtekinthetők a rendelések asztalhoz, névhez és fizetési módszerhez köthetően.

**Alkalmazotti felület**

Az étterem alkalmazottainak szánt felület ahol kezelhetik az online foglalásokat, láthatják a jelenlegi foglalt asztalokat, ezzel egyszerűsítve az újonnan érkező, foglalással nem rendelkező vendék elhelyezését is. Ezenfelül a rendeléseket kezelhetik, felvehetnek és törölhetnek melyet követően tovább küldhetik a végleges rendeléseket a konyhai felületre elkészítésre.

**Konyhai felület**

A konyhai séfek számára fenttartott felület melyen követhetik a már elkészült és a még elkészítésre váró rendeléseket az esetleges egyéb hozzáfűzésekkel (xy nélkül kért xy etc.). Egy gombnyomásra áthelyezhetik az ételeket elkészült állapotba, jelezve a pincéreknek a kiszolgálás szükségességét. Egy egyszerű, de létfontosságú felület.

**Fizetési felület**

Biztosítja a fizetendő összeg kimutatását asztalhoz és névhez kötve, egy részletes „nyugtát” biztosítva a rendelt ételekről, italokról a hozzájuk tartozó megjegyzésekkel. Fizetést követően az adatbázisban eltárolja a fizetett termékeket szintén asztalhoz és névhez kötve a fizetés időpontjával.

**Étterem weboldal (Csak vizsgára)**

Az online asztalfoglalás és étlap megtekintésére szánt egyszerűsített weboldal amely alkalmas a fentebb említett funkciók tesztelésére és bemutatására.