**ENERGETIKAI TECHNIKUM ÉS KOLLÉGIUM**

**Vizsgaremek**

**Consumption Management System**

**2025.**

|  |  |
| --- | --- |
| **Szakma:** | **Készítették:** |
| **Szoftverfejlesztő és -tesztelő** | **Dobosi Gábor** |
| **5 0613 12 03** | **Mák Luca** |
|  | **Orbán Barnabás** |

Tartalomjegyzék

[1 Bevezetés 4](#_Toc192147898)

[2 Témaválasztás indoklása 4](#_Toc192147899)

[2.1 Indoklás és aktualitás 4](#_Toc192147900)

[2.2 Kutatások, követelmények 5](#_Toc192147901)

[2.3 Eszközök és eljárások 5](#_Toc192147902)

[2.3.1 Általános fejlesztéshez használt eszközök 5](#_Toc192147903)

[2.3.2 Frontend fejlesztéshez használt eszközök 6](#_Toc192147904)

[2.3.3 Adatbázis fejlesztéshez használt eszközök 7](#_Toc192147905)

[3 Fejlesztői dokumentáció 9](#_Toc192147906)

[3.1 Frontend 9](#_Toc192147907)

[3.2 Backend 9](#_Toc192147908)

[3.2.1 Specifikáció 9](#_Toc192147909)

[3.2.2 Az alkalmazott fejlesztői eszközök 9](#_Toc192147910)

[3.3 Adatmodell leírása 12](#_Toc192147911)

Alulírott **Dobosi Gábor** büntetőjogi felelősségem teljes tudatában nyilatkozom arról, hogy az itt szereplő vizsgaremek csoportmunka eredménye és sem részeiben sem egészében nem került még kereskedelmi forgalomba, ill. publikálásra, a GPL licenszelésű programrészek kivételével.

Paks, 2025. április 18.

**Dobosi Gábor**

Alulírott **Mák Luca** büntetőjogi felelősségem teljes tudatában nyilatkozom arról, hogy az itt szereplő vizsgaremek csoportmunka eredménye és sem részeiben sem egészében nem került még kereskedelmi forgalomba, ill. publikálásra, a GPL licenszelésű programrészek kivételével.

Paks, 2025. április 18.

**Mák Luca**

Alulírott **Orbán Barnabás** büntetőjogi felelősségem teljes tudatában nyilatkozom arról, hogy az itt szereplő vizsgaremek csoportmunka eredménye és sem részeiben sem egészében nem került még kereskedelmi forgalomba, ill. publikálásra, a GPL licenszelésű programrészek kivételével.

Paks, 2025. április 18.

**Orbán Barnabás**

# Bevezetés

Csapatunk egy valós problémára keresett megoldást, amely egy olyan szoftver kifejlesztéséhez vezetett, mely szállodákban, fürdőkben és egyéb komplex szolgáltatóhelyeken alkalmazható. A program célja az volt, hogy megkönnyítse az intézmény dolgozóinak munkáját, miközben a vendégek számára kényelmesebb élményt biztosít. A rendszer egy helyre gyűjti a felhasználók költéseit, melyet egy ID vagy vonalkód segítségével tudnak beazonosítani az alkalmazottak.

Példának okáért vegyünk egy fürdőt amire a programunk is alapul, de kis változtatással bármely más szolgáltatást nyújtó területre könnyen átalakítható. A vendégek belépéskor kapott karszalagjukhoz kapcsolódva igénybe vehetik a különböző szolgáltatásokat anélkül, hogy azonnali fizetésre lenne szükség. Úgy oldható meg hogyha a karszalagjukra vannak terhelve az igénybe vett szolgáltatások.

Ez több szempontból is előnyös. Először is nem kell egész nap magukkal hordani a pénztárcájukat vagy bankkártyájukat, ami kényelmi és biztonsági szempontból is praktikus. Emellett nem kell minden igénybe vett szolgáltatásnál fizetni külön-külön, hanem egyszerre kényelmesen eltudják intézni a látogatásuk végén. Fizikai eszközök is rendelkezésünkre álltak így felhasználtuk azokat is a munkához, ezzel is szimulálva az éles környezetet.

# Témaválasztás indoklása

## Indoklás és aktualitás

A témaválasztás nem volt egyszerű. Szerettünk volna egy olyan alkalmazást létrehozni, ami kihívások elé állít nem csak a minimumot teljesíteni. Számos ötlet is felmerült, például webshop, Rubik kocka időmérő. A választásunk végül egy szolgáltatás kezelő rendszerre esett. Éles környezet minél jobb reprezentációja érdekében fizikai eszközöket is bevontunk a folyamatba, így került sor a karszalagok, az RFID olvasó és a vonalkódolvasó alkalmazására. Figyelembe véve azt, hogy intézményünk területén jelenleg is alkalmaznak hasonló eszközöket a beléptetéshez, a kezdeti tesztek ezeknek a segítségével mentek végbe.

Aktualitás szempontjából is megállja a helyét, mivel mindenki egyre jobban törekszik a gyorsaságra, kényelemre, egyszerűségre. A programunk pontosan ezt kívánja megvalósítani.

A fejlesztés során a legkorszerűbb technológiákra alapoztunk, így a projekt .NET 8-ra és a React keretrendszerre épül. Ezáltal nem csak a legújabb fejlesztéseket használtuk ki, de a dinamikus, API-központú webfejlesztés irányelveit is követtük, amely a jövőbeli bővíthetőség szempontjából is kiemelten fontos.

## Kutatások, követelmények

Első lépésként áttekintettük a vizsgakövetelményeket annak érdekében, hogy azonosítsuk a kötelező elemeket. Az alkalmazásnak vagy weboldalnak tartalmaznia kellett egy frontend részt, amely a megjelenítésért és az adatok betöltéséért felelős. Emellett szükség volt egy backend rendszerre is, amely az adatok tárolását, a jogosultságkezelést, az adatbázis-táblák és kapcsolatok pontos definiálását, valamint az API végpontok meghatározását biztosítja, lehetővé téve a frontend és a backend közötti kommunikációt.

A hatékonyság érdekében frontend oldalról React keretrendszert alkalmaztunk. A backend fejlesztéséhez az ASP.NET Core Web API technológiát választottuk, amely a C# programozási nyelven alapul.

A szoftver adatbázis MSSQL alapú, amely az Entity Framework segítségével könnyen kezelhető. A programunk megfelel az alapvető adatkezelési műveleteknek (CRUD), amely az adatok létrehozását (Create), lekérdezését (Read), módosítását (Update) és törlését (Delete) foglalja magában. Ezek a backend oldalon GET, POST, PUT és DELETE HTTP-metódusokként valósultak meg.

Igyekeztünk a tiszta kód elveit alkalmazni, amelyek elősegítik a fejlesztés hatékonyságát, az átláthatóságot, valamint a későbbi módosítások egyszerűbb elvégzését. Ennek érdekében például az egyértelmű változóneveket a PascalCase elnevezési konvenció szerint definiáltuk, és logikus, jól strukturált mappaszerkezetet alakítottunk ki.

## Eszközök és eljárások

### Általános fejlesztéshez használt eszközök

#### Trello

A Trello egy vizuális projektmenedzsment eszköz, amely Kanban-alapú táblák, listák és kártyák segítségével segíti a feladatok szervezését. Fejlesztőcsapatok számára kiválóan alkalmas sprint tervezésre, backlog kezelésre és feladatkövetésre. Integrálható GitHubbal, Slackkel és más fejlesztői eszközökkel.

#### GitHub

A GitHub egy népszerű forráskód-kezelő és verziókövető platform, amely a Git rendszeren alapul. Lehetővé teszi a fejlesztők számára a kód tárolását, verziózását, csapatmunkát és CI/CD (folyamatos integráció és telepítés) folyamatok kezelését. Nyílt forráskódú és privát projektekhez egyaránt használható.

### Frontend fejlesztéshez használt eszközök

#### Visual Studio Code:

A **VSCode** egy ingyenes, könnyű, mégis erőteljes kódszerkesztő, amelyet a Microsoft fejlesztett. Beépített támogatást nyújt JavaScript, TypeScript és React fejlesztéshez, valamint számos kiegészítő (pl. ESLint, Prettier) elérhető hozzá a fejlesztési élmény javítása érdekében.

#### Vite

A Vite egy modern fejlesztési szerver és build eszköz, amely gyorsabb fejlesztői élményt nyújt a React és TypeScript alapú alkalmazások számára. Használata lehetővé teszi a gyorsabb hot-reloadot és optimalizált végső kódot.

#### Material-UI (MUI)

A Material-UI egy népszerű React UI komponenskönyvtár, amely a Google Material Design elveit követi. Segítségével modern, reszponzív és könnyen testreszabható felhasználói felületeket lehet fejleszteni.

#### Axios

Az Axios egy ígéreteken (Promises) alapuló HTTP kliens, amelyet adatok lekérésére és küldésére használnak API-khoz. Különösen népszerű a React alkalmazásokban, mivel egyszerűsíti az aszinkron hálózati kéréseket és kezelést.

#### React & React-DOM

A React egy népszerű JavaScript könyvtár interaktív felhasználói felületek fejlesztésére. A React-DOM a React komponensek DOM-ba való rendereléséért felelős, biztosítva a hatékony frissítéseket és a komponens-alapú fejlesztési modellt.

#### TypeScript

A TypeScript egy JavaScript alapú programozási nyelv, amely statikus típusellenőrzést biztosít. A React alkalmazásokban használva segít a hibák csökkentésében és a kód karbantarthatóságának javításában.

#### ESLint

Az ESLint egy JavaScript és TypeScript kódellenőrző eszköz, amely segít fenntartani a kódminőséget, betartani a kódolási szabványokat és azonosítani a lehetséges hibákat már fejlesztés közben.

#### Prettier

A Prettier egy automatikus kódformázó eszköz JavaScript, TypeScript, HTML, CSS és sok más nyelv számára. Segítségével egységes kódstílust tarthatunk fenn a projektben, anélkül hogy manuálisan kellene figyelni a formázásra.

#### Tailwind CSS

A Tailwind CSS egy utility-first megközelítésű CSS keretrendszer, amely előre definiált osztályokat biztosít a gyors és hatékony stílusozáshoz. Ahelyett, hogy kész komponenseket adna (mint pl. Bootstrap vagy Material-UI), kis, önálló osztályokat kínál, amelyekkel teljes mértékben testreszabható a dizájn.

#### Whimsical

A Whimsical egy vizuális tervezőeszköz, amely támogatja diagramok, drótvázak, folyamatábrák és gondolattérképek készítését. Ideális UX/UI tervezéshez, rendszerarchitektúrák megjelenítéséhez és csapatmunkában történő ötleteléshez.

### Adatbázis fejlesztéshez használt eszközök

#### SQLEXPRESS

A Microsoft SQL Server ingyenes, könnyített verziója, amely fejlesztési célokra ideális. Jól integrálható ASP.NET Web API alkalmazásokkal az Entity Framework Core használatával.

#### SQL Server Management Studio

Grafikus felhasználói felületet biztosító eszköz a Microsoft SQL Server kezelésére. Lehetővé teszi az adatbázisok kezelését, lekérdezések futtatását, tárolt eljárások létrehozását.

#### Dbeaver

Ingyenes, többplatformos adatbázis-kezelő eszköz, amely támogatja a SQL Server, MySQL, PostgreSQL, SQLite és más adatbázisokat, vizuális szerkesztővel és lekérdezési funkciókkal.

#### drawDB

A drawDB egy adatbázis-tervező eszköz, amely lehetővé teszi az ER-diagramok (Entity-Relationship) és adatbázisstruktúrák vizuális modellezését. Segítségével könnyedén megtervezhető egy rendszer adatmodellje, amely később SQL-kóddá alakítható.

# Fejlesztői dokumentáció

## Frontend

### A programmal kapcsolatos követelmény:

A program egy fürdő dolgozói számára lett kifejlesztve, figyelembe véve a felhasználói kör sajátosságait és igényeit. A fejlesztés során az összes lehetséges hibát és problémát feltérképeztük, amikor a felhasználó figyelmetlenségből vagy hibás döntés következtében súlyos problémákat okozhat. Ezek közé tartozhatnak kisebb működési zűrzavartól kezdve egészen a rendszer működését teljesen leállító vagy visszafordíthatatlan hibákig terjedő következmények. A kockázatok minimalizálása érdekében a programot alapos tesztelésnek vetettük alá, többféle tesztelési módszert alkalmazva annak érdekében, hogy a rendszer minden körülmény között megbízhatóan működjön.

### Megjelenési követelmény:

* Bejelentkezési felület:

A rendszer tartalmaz működő beviteli mezőket, ahol a felhasználónevet és a jelszót lehet megadni. Ezen kívül egy "Küldés" gomb is elérhető, amely lehetővé teszi a megadott adatok elküldését. Amennyiben hibás vagy nem létező adat kerül megadásra, egy felugró figyelmeztető üzenet jelenik meg. Hasonló figyelmeztetés jelenik meg abban az esetben is, ha a felhasználó nem ad meg felhasználónevet vagy jelszót. Ha a megadott adatok helyesek, a rendszer a kezdőoldalra irányítja a felhasználót. Kis képernyőn vagy mobil eszközön a bejelentkezési felület melletti kép a képernyő tetejére csúszik.

* Regisztrációs felület:

A felületen található felhasználónevet, vezetéknevet, keresztnevet, jelszót és feladatkört tartalmazó beviteli mező. A regisztrációs folyamatot egy gomb indítja, amely elküldi a megadott adatokat. Sikeres regisztráció után egy felugró ablak tájékoztatja a felhasználót a sikeres regisztrációról. Ez a felület kizárólag az adminisztrátorok számára elérhető, azonban a kötelező mezők kitöltését a rendszer minden esetben ellenőrzi, és hibaüzenetet küld, ha valamelyik mező üres marad. Kis képernyőn vagy mobil eszközön a regisztrációhoz tartozó kép a képernyő tetejére csúszik.

* Főképernyő:

Jogosultságtól függően megjelenő kártyák, amikre kattintva más oldalakra lehet továbbjutni. A kártyákra ráhúzott egér esetén megjelenő szöveg, amin az adott munkakör vagy munkafelület neve látszódik.

* Étterem felület:

Bal oldalt egy navigációs sáv, amin a különböző ételtípusok nevei szerepelnek. Egy ételtípus csak akkor jelenik meg hogyha van olyan étel, ami abba az étel típusba tartozik. Az ikonokra kattintva a termék sávban csak az adott termékcsoporthoz tartozó ételek lesznek megjelenítve. Ez a sáv kis képernyőn felülre csúszik.

Egy Rendelés sáv, ahol a folyamatban lévő rendelés ételei vannak felsorolva. A felsorolásban szerepel az étel ikonja, neve, darabszám egy kuka gomb, és egy ár. Egy összeg, ami a rendelésbe berakott termékek árát írja forintban. Egy rendelés leadása gomb, ami kattintásra feldob egy párbeszéd ablakot, amibe be kell írni a karszalag számát így kapcsolva a felhasználóhoz az adott rendelést. Egy „mégse” és egy „leadás” gomb. A „leadás” gombra kattintva sikeres rendelés esetén egy felugró ablak, ami jelzi, hogy a rendelés sikeresen le lett adva.

A megjelenő termékek sávja. Alapértelmezett képernyőméret esetén 3 kártya van egymás mellett ez a szám a képernyő méretének csökkentésével szintén csökken. Egy kártyán szerepel egy név, kategória, kép, rövid leírás, egy jelzés, hogy rendelhető-e, egy plusz, mínusz gomb, amivel a rendeléshez lehet adni az ételt. A gombokra húzva az egeret a szín megváltozik. Nem rendelhető státusz esetén a kártya inaktív lesz és nem lehet hozzáadni a rendeléshez azt az ételt.

Fejléc, ahol logó, név, egy kezdőlap gomb, menü gomb és a bejelentkezett ember ikonja jelenik meg. Az ikonra kattintva az általános gombok jelennek meg; kijelentkezés, fiók.

Lábléc.

## Backend

### Specifikáció

A projektünk célja egy ASP.NET 8 backend alkalmazás fejlesztése, amely képes a fogyasztásnyilvántartó rendszer adatait szolgáltatni a frontend felé.

**Technológiák:**

* RESTful API-k, hitelesítési folyamatok JWT és token használata

**Funkcionalitások:**

* CRUD műveletek, melyet a felhasználó jogosultságától függően tud elvégezni.

**Biztonsági szempontok**

* Érzékeny adatok, mint például a jelszavak hashelve kerülnek eltárolásra az adatbázisban. DTO-k használata, mely segítségével kizárólag a megfelelő adatok kerülnek megjelenítésre.

**Teljesítmény-igények**

* Gyors válaszidő, illetve több felhasználó egyidejűleg való kiszolgálása.

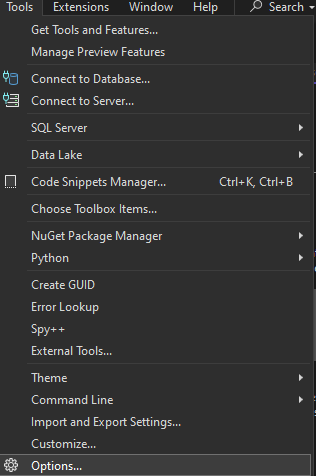
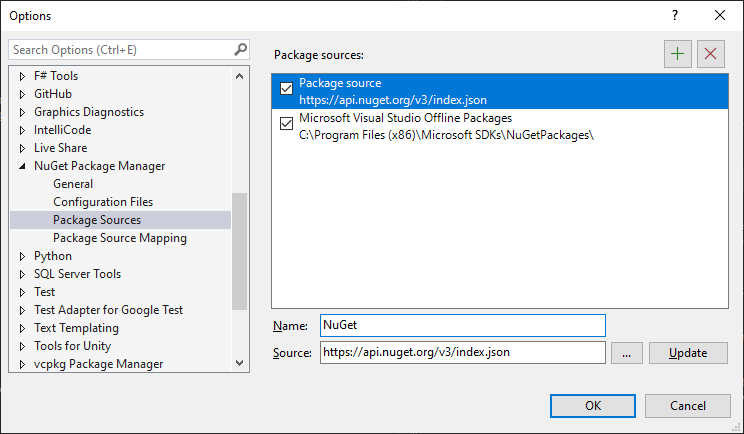
### Az alkalmazott fejlesztői eszközök

**Visual Studio 2022:**

A Visual Studio 2022 egy fejlett fejlesztőkörnyezet, amely támogatja az ASP.NET Web API fejlesztését C# nyelven. Beépített NuGet csomagkezelővel könnyedén hozzáadhatók és frissíthetők a szükséges csomagok, például Entity Framework Core, Swashbuckle (Swagger) és más API-fejlesztést segítő könyvtárak.

**Általunk alkalmazott NuGet csomagok (Dependency Management):**

A NuGet csomagokat a Visual Studio-ba a NuGet Package Manager segítségével telepítjük és használjuk fel a programunk során. Alapértelmezetten a Visual Studio nem minden esetben tartalmazza a NuGet Package forrását. Ezt manuálisan kell beállítani. Melyet a *Tools* 🡪 *Options* 🡪*NuGet Package Manager* 🡪*Package Sources* alatt találunk. Itt vegyük fel a következő értékekkel új csomag forrást:

* **Name:** *NuGet*
* **Source:** *https://api.nuget.org/v3/index.json*
* BCrypt.Net-Next (by Chris McKee, Ryan D. Emerl, Damien Miller) – 4.0.3:
* .NET-ben használható könyvtár, amely a BCrypt hash-elési algoritmus implementációját biztosítja. Jelszóhash-ek generálására és ellenőrzésére, mivel erős védelemmel rendelkezik brute-force támadások ellen.
* Microsoft.AspNetCore.Authentication.JwtBearer (by Microsoft) – 8.0.13:
* Entity Framework Core egy modern objektum-relációs adatbázis leképező .NET-hez, támogatja a LINQ lekérdezéseket, módosításokat, sémamigrációkat.
* Microsoft.EntityFrameworkCore (by Microsoft) – 9.0.2:
* Ez a middleware lehetővé teszi a JWT tokenek érvényesítését HTTP-kérések során, biztosítva a felhasználók hozzáférésének biztonságos kezelését
* Microsoft.EntityFrameworkCore.Design (by Microsoft) – 9.0.2
* Az Entity Framework Core eszközeihez szükséges fejlesztési időben használt komponensek.
* Microsoft.EntityFrameworkCore.Sqlite (by Microsoft) – 9.0.2
* Sqlite adatbázis kezelő Entity Framework Core-hoz.
* Microsoft.EntityFrameworkCore.Tools (by Microsoft) – 9.0.2
* Az Entity Framework Core eszközeihez szükséges NuGet csomag.
* Microsoft.VisualStudio.Web.CodeGeneration.Design (by Microsoft) – 8.0.7 / 9.0.0:
* Kódgenerálás az ASP.NET Core-hoz, tartalmazza a dotnet-aspnet-codegenerator parancsot vezérlők és nézetek generálásához.
* Swashbuckle.AspNetCore (by domaindrivendev) – 6.6.2 / 7.2.0
* Kizárólag fejlesztői környezethez használt Swagger eszköz, mely képes a végpontok tesztelésére.

**Postman**

A Postman egy népszerű API-fejlesztő eszköz, amely lehetővé teszi HTTP-kérések küldését, API-k tesztelését és dokumentálását, támogatva az általunk is használt JWT hitelesítési módszert.

**Swagger UI**

A Swagger UI egy könnyen használható eszköz, amely lehetővé teszi a RESTful API-k dokumentációját. A fejlesztők megtekinthetik, tesztelhetik és kipróbálhatják az API végpontokat közvetlenül egy böngészőből, mely fejlesztési módban a backend szerver indításával automatikusan indul.

**Package Management Console**

A Package Management Console a Visual Studio eszköze, amellyel NuGet csomagokat kezelhetünk és migrációkat futtathatunk, például Add-Migration és Update-Database parancsokkal.

## Adatmodell leírása

Az adatbázisunk relációs adatbázison alapul. A fejlesztés kezdeti fázisában az MSSQL-t használtunk, majd az egyszerűség miatt áttért a csapatunk az SQlite-ra. Természetesen figyelve, hogy minden egyes követelménynek megfeleljen az új adatbázis típus. Sqlite egyik előnye, hogy az adatok eléréséhez nem szükséges adatbázis szervert futtatni, egy .db kiterjesztésű fájlban tárolja az adatokat, mely szintén egy relációs adatbázist hoz létre. Az ASP .NET lehetővé teszi, hogy szinte azonnal, pár sor módosításával átálljon a rendszerünk egy másik adatbázis típusra is, mint például: *MySql, MSSQL*….

A csapatunk által használt programozási nyelv a C# volt. Az adatbázis tábláit külön-külön egy a *models* mappában lévő osztályokból tevődnek össze. A táblák mezői pedig az *Entity Framework* ***Migration***parancsa dolgozza át az adott adatbáziskezelő rendszer nyelvére, mely létrehozza a táblákat.

Táblák:

### Admission

Ebben a táblában kerülnek eltárolásra a különféle jogosultságok, melyet a vendég tud megvásárolni. Belépők és kiegészítő jegyek típusai.

|  |  |  |  |
| --- | --- | --- | --- |
| Admissions | | | |
|  | Típus | Mező név | Megjegyzés |
| 🔑 | int | Admission Id | Auto increment |
|  | string | AdmissionName | K |