Ceglédi SZC Közgazdasági és Informatikai Technikum

Szoftverfejlesztő és -tesztelő

5 0613 1203

Fejlesztői dokumentáció a „Döner Cegléd” alkalmazáshoz

Jakab Botond, Krausz Márton Alajos, Tóth Dániel János

Cegléd

2025

# **Tartalom**

[**Tartalom** 2](#_Toc194078816)

[**Adatbázis adatmodellek** 4](#_Toc194078817)

[**Asztali alkalmazás** 5](#_Toc194078818)

[Feladat/cél 5](#_Toc194078819)

[Kommunikáció a backenddel és a használt osztályok 7](#_Toc194078820)

[Előkészítés: 7](#_Toc194078821)

[API kapcsolat 7](#_Toc194078822)

[Kapcsolati hiba jelzése 7](#_Toc194078823)

[Pár API hívás 8](#_Toc194078824)

[Főbb használt osztályok 11](#_Toc194078825)

[Használt programok, fejlesztői környezetek és futtatási környezet 14](#_Toc194078826)

[Futtatási környezet és hardware 14](#_Toc194078827)

[**Backend** 15](#_Toc194078828)

[Feladat/Cél 15](#_Toc194078829)

[Használt fejlesztési környezet, programok és csomagok 15](#_Toc194078830)

[File struktúra 16](#_Toc194078831)

[Generált dokumentáció 16](#_Toc194078832)

[A dokumentáció elérése 16](#_Toc194078833)

[**Frontend (weboldal)** 18](#_Toc194078834)

[Feladat/Cél 18](#_Toc194078835)

[Használt fejlesztési környezet, keretrendszerek és csomagok 18](#_Toc194078836)

[Kommunikáció a backendel 19](#_Toc194078837)

[Példa adatlekérésre 20](#_Toc194078838)

[Példa adat küldésre 22](#_Toc194078839)

[**Backend tesztek** 24](#_Toc194078840)

[Feladat/Cél 24](#_Toc194078841)

[Környezetleírás 24](#_Toc194078842)

[A fájl felépítése 24](#_Toc194078843)

[Fontosabb tesztek bemutatása 25](#_Toc194078844)

[**Források** 30](#_Toc194078845)

# **Adatbázis adatmodellek**

még meg lesz csinálva

# **Asztali alkalmazás**

## Feladat/cél

Az alkalmazás célja az éttermi működés hatékonyabbá tétele, az adatok gyors elérése és módosítása, valamint a felhasználók és rendelések egyszerű kezelése. A felhasználóbarát felület és a valós idejű kommunikáció biztosítja a gördülékeny munkafolyamatokat az étterem minden területén.

Rendelések megjelenítése és kiadása

* Az alkalmazás valós időben jeleníti meg az étteremhez beérkező rendeléseket.
* Lehetőség van a rendelések részletes megtekintésére, beleértve a termékek listáját, a mennyiségeket, az árakat és az esetleges megjegyzéseket.
* A rendelés kiadása után az állapot frissíthető az API-n keresztül.

Regisztrált felhasználók kezelése

* Az alkalmazás lehetőséget biztosít a regisztrált felhasználók kezelésére.
* A felhasználók deaktiválhatók, ha már nem aktívak az étteremben.
* A felhasználói adatok, például a név, e-mail cím, szerepkör vagy jogosultságok szerkeszthetők.

Új termékek felvitele

* Új termékek hozzáadása egyszerű és gyors folyamat.
* A következő adatok rögzíthetők az új termékhez:
  + Termék neve
  + Ár
  + Lehetséges módosítási opciók (pl. extra feltétek, szószok)
  + Termék kategóriája (pl. Wrap, Kebab, Drink, SideDish)
  + Termék képe feltölthető PNG formátumban
* Az új termékek mentése az API-n keresztül történik.

Már meglévő termékek teljes körű szerkesztése

* Az alkalmazás támogatja a termékek teljes körű szerkesztését.
* A meglévő termékek adatai bármikor módosíthatók, beleértve:
  + Nevet
  + Árat
  + Módosítási lehetőségeket
  + Kategóriát
  + Termékképet
* Lehetőség van a termékek archiválására vagy inaktiválására, ha már nem elérhetők.

Felület minta

A képen szöveg, menü, Számítógép-monitor, fedett pályás látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## Kommunikáció a backenddel és a használt osztályok

### Előkészítés:

1. Győződjön meg róla hogy fut az API (ennek indítása megtalálható a backend részen)

2. Lehetőleg ne hálózati meghajtóról futtassa az alkalmazást vagy győződjön meg róla, hogy külső forrásból származó alkalmazások használhatják e a helyi hálózatot az adott meghajtón.

### API kapcsolat

Mivel folyamatosan érkezhetnek új megrendelések ezért adott időnként „meg kell kérdezni” az API-tól hogy van e új rendelés és ha igen akkor megjeleníteni azt.



pl.: 2000 ezred másodperc 🡪 2 másodperc



Ez a változó felel azért hogyha érkezik új rendelés tudja az alkalmazás hogy frissítenie kell a rendeléseket és a sikeres frissítés után ez false-ra kell változzon

### Kapcsolati hiba jelzése

Ugyan így folyamatosan ellenőrzésre szorul, hogy él e a kapcsolat az alkalmazás és az API között mivel ha ez megszűnik nem érkeznek be a rendelések és ellenőrizni kell a szervert/adatbázist/backendet



Ez a változó van ellenőrizve és állítva megadott időnként

### Pár API hívás

1. Kapcsolat lekérdezés GET

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Ez az API hívás azért felel hogy ellenőrizze a kapcsolatot, igazából erre bármilyen GET kérés megfelelő mivel adatot nem várunk vissza csak a státusz kódot figyeljük, hogy OK legyen.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Az alkalmazás indításakor meg kell hívni és adni egy értéket a kapcsolat meglétének majd ezután jön a folyamatos ellenőrzés

2. Rendelések lekérdezése GET

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Ez a kérés az aktív rendeléseket kapja meg és adja vissza JSON formátumban. Részletes hiba kezelés van beépítve, hogy a felhasználó tudja milyen hibával keresse meg a fejlesztőt.

3. Admin bejelentkezés POST

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Az admin bejelentkezése azért POST metódus, mivel az express API beküldött adatot csak így tud elfogadni és biztonsági szempontból az adatot csomagban kell küldeni nem az URL-ben megadni paraméterként.

### Főbb használt osztályok

1. FullUser és a User osztályok

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.A képen szöveg, képernyőkép, Betűtípus, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Szerepük megegyezik csak a felhasználási körük más amikor szükség van egy adott felhasználó összes adatának tárolására akkor van használatban a FullUser osztály (pl.: egy felhasználó módosításakor), ellenkező esetben a sima User osztály. Erre azért van szükség hogy ne legyenek üres mezők tárolva feleslegesen.

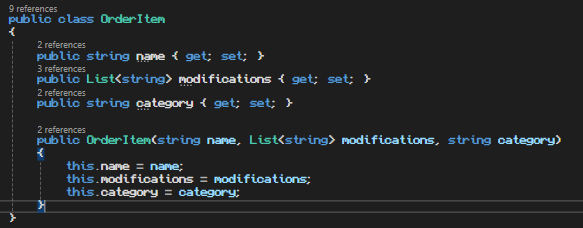
2. MenuItem osztály

A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Amikor egy terméket minden adatával szeretnénk tárolni pl.: annak felvitelekor/módosításakor

3. Order és OrderItem osztályok

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Egy rendelés megjelenítésekor több termék kerülhet egy rendelésbe és ezeknek minden adatát tárolni szeretnénk külön-külön erre szolgál ez a két osztály.

## Használt programok, fejlesztői környezetek és futtatási környezet

Használt fejlesztői környezetek





A fejlesztéshez használt fejlesztői környezet

Letöltés:

<https://visualstudio.microsoft.com/downloads/>



### Futtatási környezet és hardware

Windows 11-es PC 32GB memóriával

Fejlesztői környezet: Visual Studio 2022 Community Edition

Processzor: AMD Ryzen 5600X

Memória: 2X16GB 3200Mhz DDR4 ram

Tárhely: 1TB NVME SSD

GPU: Nvidia RTX 3060Ti

Megjelenítés: 1920\*1080

Egér: Szükséges

Billentyűzet: Szükséges

# **Backend**

## Feladat/Cél

A backend célja egy REST API készítése volt, amit az express csomag használatával valósítottunk meg. Az API a háttérben fut és végpontok segítségével végez el különféle műveleteket az Asztali alkalmazás és a weboldal számára. Ezek a műveletek adatbázissal kapcsolatosak. A backend az adatbázisunkkal 4 fő műveletet tud végrehajtani, adat lekérés, adat feltöltés, adat módosítás vagy adat törlés. Az adatbázis megvalósításához a MySQL-t választottuk, az adatbázis modelleket, kapcsolatot és egyebeket a sequelize csomag használatával backenden valósítottuk meg.

## Használt fejlesztési környezet, programok és csomagok

Fejlesztési környezet: Visual Studio Code

Használt programok:

* futtatáshoz: XAMPP
* teszteléshez: Postman

Használt (npm) csomagok:

* Express
* Sequelize
* jsonwebtoken (JWT)
* bcrypt
* nodemon
* dotenv
* cors
* body-parser
* cookie-parser
* nodemailer
* swagger-autogen

## File struktúra

A file struktúrát tekintve különböző szintekhez kirendezve dolgozunk. A programunk a server.js file indításával működtethető, ez elindítja az app.js-t, amiben a routes mappa fájljai hívódnak meg. A különböző útvonalak hívásai mellé esetleges middleware file tartozik, majd egy funkció, ami az adott controllerben van definiálva. A controllerekből a service fájlok hívodnak meg, amik végül a repository fileokat hívják meg. Mindegyik szint a saját feladatát ellátva oldja meg az egységes működést.

## Generált dokumentáció

A dokumentációt a swagger nevű csomag segítségével valósítottuk meg, ami az összes végpontra részletes leírást biztosít. A dokumentáció minden végpontra tartalmazza a bekért paramétereket (amennyiben vannak) és a kimeneti értékeket (amennyiben vannak), mindkettőt azonos formátumban, mint amit a program fejlesztéséhez használtunk.

## A dokumentáció elérése

A dokumentációt a backend futtatása után az alábbi URL-en érhetjük el, bármelyik böngészővel: localhost:3000/api-docs

A képen képernyőkép, Multimédiás szoftver, szöveg látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Amennyiben sikeresen elindítottuk a backendet és helyesen beírtuk az oldal címét az alábbi felület fogad minket:

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A sorok lenyitásával részletes leírást kaphatunk az adott végpontról:

A képen szöveg, képernyőkép, szoftver, képernyő látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

# **Frontend (weboldal)**

## Feladat/Cél

A weboldal célja egy felhasználóbarátként kialakított rendelőfelület, ahol a felhasználó regisztráció és bejelentkezés után különféle ételeket tud kiválasztani és módosítani kínálatunkból, amelyeket a virtuális kosarába rakva később megrendelhet. Vásárlásaival pontokat gyűjthet, amelyeket későbbi vásárlásoknál beválthat. Módosíthatja különböző adatait, illetve megtekintheti és újra rendelheti előző vásárlásait. A megvalósításhoz Vue js-t használtunk, mint keretrendszer és a mellé társuló Vueitfy-t designoláshoz.

## Használt fejlesztési környezet, keretrendszerek és csomagok

Fejlesztési környezet: Visual Studio Code

Használt keretrendszerek:

* Vue.js
* Vuetify

Használt (npm) csomagok:

* Axios
* TanStack

## Kommunikáció a backendel

A backendel való kommunikáció az Axios segítségével történik, a hívások számának csökkentése érdekében pedig a TanStack Query-t használjuk az adatok cache-elésére. A redundáns sorok elkerülése érdekében az Axios kliens létrehozáskor beállítjuk az opciók értékeit, így azok minden híváskor automatikusan érvényesülnek és nem szükséges őket ismételten konfigurálni.

A képen szöveg, képernyőkép, szoftver, Operációs rendszer látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## Példa adatlekérésre

Az alábbi kódrészlettel a menün lévő ételek információit kérjük le egy Axios hívás segítségével, majd cacheljük a TanStack Query használatával, végül megjelenítjük az oldalon.

Először létrehozunk egy típust, a lekért étel adataival.

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Az étel képének mérete miatt a hívás kap kiegészítő fejléceket, hogy az oldal probléma nélkül megkapja a nagyobb képeket is és meg tudja jeleníteni őket.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A függvényt ezután meghívjuk a kívánt oldalon.



A kapott adatot a Vue beépített „v-for” ciklusának segítségével fogjuk megjeleníteni.

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Eredmény:

A képen képernyőkép, Konyha, szöveg, étel látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## Példa adat küldésre

Az alábbi kód részlettel oldjuk meg az új felhasználók regisztrációját, egy Axios post hívást követően megadjuk a végpontot, majd átküldjük a regisztrációs adatot.

Először létrehozunk egy típust a regisztráció adataival.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Megírjuk az Axios függvényt, ami elküldi az adatokat.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Meghívjuk a függvényt és létrehozunk egy állapotváltozót, amiben tároljuk majd elküldjük az adatokat.



A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

A kellő adatoknak bemeneti mezőket helyezünk el, amikor változáskor frissítik az állapotváltozónk értékét.

A regisztráció gombhoz hozzáadunk egy függvényt, ami a regisztrációs adatot várja paraméterben.



Használjuk az Axiosos mutációnkat, ami elküldi az adatot a backend részére.

A képen szöveg, képernyőkép, Betűtípus látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Ha minden sikeres volt, átkerülünk egy másik oldalra és értesítést kapunk a sikeres regisztrációról. Amennyiben valami hiba történt, ha kapunk válaszul hiba üzenetet a backendtől, akkor azt jelenítjük meg. Amennyiben viszont más hiba történt, aminek okát nem tudjuk, egy alap hibaüzenetet fogunk megjeleníteni a felhasználónak.

# **Backend tesztek**

## Feladat/Cél

Ez a dokumentáció részlet a main.test.js teszt fájlt mutatja be, amely a projekt backend működésének menetét teszteli, evvel biztosítva a funkciók hibamentes működését, és megfelelően kezelje a hibákat és hibaüzeneteket.

## Környezetleírás

A teszt futtatása közben használt környezet:

* Windows 10 Operációs Rendszer
* Visual Studio Code 1.98.2-as verzió
* Supertest a tesztelés lefuttatásához

## A fájl felépítése

A main.test.js fájl struktúrája következő féleképpen néz ki:

* Backend tesztek:
  + Controller tesztek: (Például Dish Controller teszt)
  + Middleware tesztek: (Például errorHandler teszt)
  + Repository tesztek: (Például Dish Repository teszt)

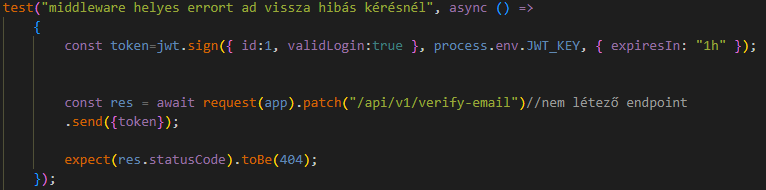
## Fontosabb tesztek bemutatása

Middlewares tesztek

A Middlewares tesztek fő célja a hibák helyesen tovább küldése, hogy a felhasználó vagy fejlesztő felismerje a hiba eredetét.

Error Handler tesztek

Az Error Handler, ahogy a neve is sugallja, a hibákat kezeli és küldi azokat tovább a kód biztonsága és működése érdekében, példa erre:

Ebben a példában megpróbálunk egy Controller tesztet lefuttatni, azonban egy nem létező végpontra, ami miatt hibába ütközik a folyamat és 404-es státuszkódot, hibakódot várunk vissza.

Ezt a tesztet azért kell külön “köztes réteggel” (middleware-el) letesztelni, mível a controllerben nem tudjuk azt az esetet letesztelni, ha rossz a végpont, mert nem éri el, ezért az errort tovább küldjük és visszaadja a middleware.

Repository tesztek

A repository, “tároló” egységek tesztelésének a célja az adattárolás működésének és hibakezelésének biztosítása, emellett biztosítja, hogy az előforduló hibák megelőzhetőek legyenek és stabil maradjon a kód.

User Repository tesztek

A User Repository tesztek főbb célja a felhasználóval kapcsolatos funkciók kezelését biztosítja, példa erre:

A képen szöveg, képernyőkép, szoftver látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Ebben a példában leteszteljük a Felhasználó létrehozását és annak lekérését.  
Előre beégetett adatokat használunk, így könyebben ráláthatunk a lehetséges hibákra.

Dish Repository tesztek

A Dish Repository tesztek főbb célja a fogásokkal, ételekkel kapcsolatos funkciók kezelését biztosítja, példa erre:

A képen szöveg, képernyőkép látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Kettő példa látható, az első példában leteszteljük a fogás létrehozását és annak lekérését.  
Előre beégetett adatokat használunk, így könnyebben ráláthatunk a lehetséges hibákra.

A második példában az összes leteszteljük az összes fogás lekérését és elvárjuk, hogy a visszakapott hossz 1 legyen, mivel az előző tesztben készítettünk egyet.

Controller tesztek

A vezérlőrétegen végzett tesztek biztosítják a kód működését és a kérések, válaszok helyességét, emellett a bemeneti adatokat is vizsgáljuk és több esetre figyelünk tesztelés során.

User Controller tesztek

A User Controller tesztekkel biztosítjuk, hogy a felhasználóhoz kapcsolódó kérések és válaszok helyes értékkel térnek vissza és hiba nélkül mennek végbe, példa erre:

Ebben a példában kettő tesztet láthatunk, amelyek a felhasználó létrehozását kezelik, az egyiknél helyes választ várunk, míg a másiknál helytelent. Az első tesztben az” /api/v1/register” végponton a kérés testében beküldjük a létrehozáshoz, regisztráláshoz elvárt szükséges adatokat, ami után elvárjuk, hogy a válasz státuszkódja a helyzetnek megfelelően helyesen térjen vissza. A második tesztben ugyanazt a kérést teszteljük, azonban ebben az esetben egy szándékosan hibás kérést küldünk be (hiányzó adat) és várjuk az adott helyzethez megfelelő választ.

Dish Controller tesztek

A Dish Controller tesztekkel biztosítjuk, hogy a fogásokhoz kapcsolódó kérések és válaszok helyes értékkel térnek vissza és hiba nélkül mennek végbe, példa erre:

A képen szöveg, képernyőkép, Betűtípus, képernyő látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

Ebben a példában egy fogás, étel létrehozását láthatjuk, ahol helyes értéket várunk vissza.  
Az” /api/v1/dish” végponton a kérés testében beküldjük a létrehozáshoz elvárt szükséges adatokat.

# **Források**

McDonalds rendelés felület (inspiráció)

Postman

Xampp

Visual studio code

Microsoft Visual studio

CocaCola hu (termék fotók)

Leonardo.ai (generált termék fotók)