**ENERGETIKAI TECHNIKUM ÉS KOLLÉGIUM**

**Vizsgaremek**

**Nutri applikáció**

**2025.**

|  |  |
| --- | --- |
| **Szakma:** | **Készítették:** |
| **Szoftverfejlesztő és -tesztelő** | **Aczélos Csaba** |
| **5 0613 12 03** | **Balogh Csabád** |
|  | **Rummler János** |

Tartalomjegyzék

[I. Bevezetés 4](#_Toc184887498)

[II. Témaválasztás indoklása 4](#_Toc184887499)

[III. Fejlesztői dokumentáció, technikai leírás 4](#_Toc184887500)

[1. Specifikáció 4](#_Toc184887501)

[2. Az alkalmazott fejlesztői eszközök 4](#_Toc184887502)

[3. Adatmodell leírása 4](#_Toc184887503)

[4. Algoritmusok 5](#_Toc184887504)

[5. Tesztelési dokumentáció 5](#_Toc184887505)

[IV. Felhasználói dokumentáció 5](#_Toc184887506)

[1. A program általános specifikációja, célja 5](#_Toc184887507)

[2. Rendszerkövetelmények 5](#_Toc184887508)

[3. A program telepítése 6](#_Toc184887509)

[4. A program használatának a részletes leírása 6](#_Toc184887510)

[V. Összegzés 6](#_Toc184887511)

[VI. Ábrajegyzék 7](#_Toc184887512)

[VII. Irodalomjegyzék, hivatkozásjegyzék 7](#_Toc184887513)

Nyilatkozat

Alulírott Név 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.

Aczélos Csaba

Név

# Bevezetés

A célunk az volt, hogy létrehozzunk egy olyan oldalt, ahol mindenki könnyen hozzáfér az országban kapható élelmiszerek tápértékéhez, és tudomást szerezzen az esetleges káros anyagokról, amik egyes élelmiszerekben találhatók.

A bevezetőnek tartalmaznia kell:

* a vizsgaremek témájának bemutatását.

# Témaválasztás indoklása

Azért ezt a témát választottuk, mert:

* Nincsenek híres, egyszerűen használható, jelentős adatmennyiséggel rendelkező oldalak
* Nagyon kevesen ismerik bizonyos éttermi ételek veszélyeit

A téma megvalósításához tökéletes eszköz volt a React és Django nevű keretrendszer, és a Visual Studio Code fejlesztői környezet.

# Fejlesztői dokumentáció, technikai leírás

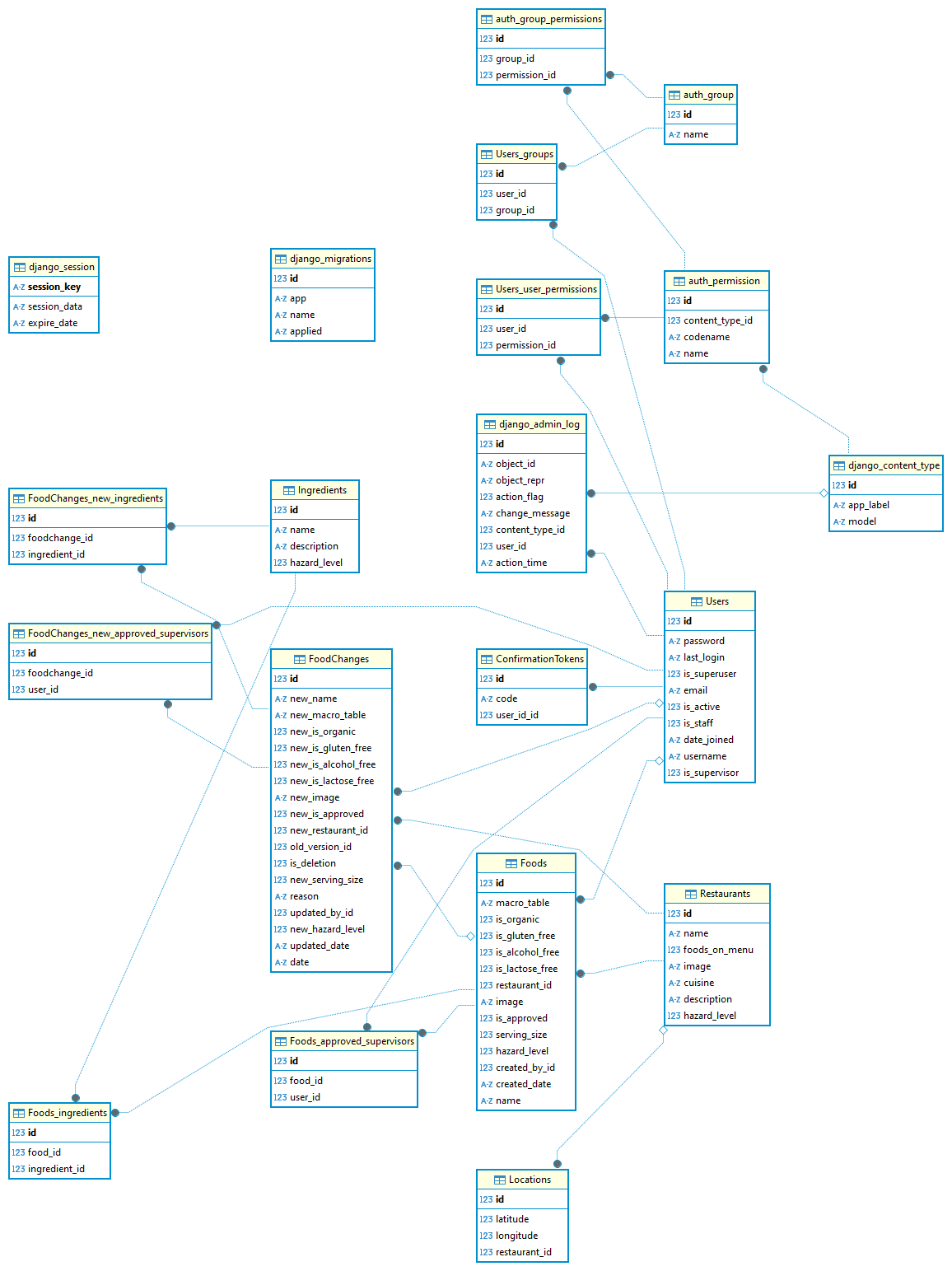
## Specifikáció

A projekt teljes kódja angolul kell, hogy legyen. A projekt fő feladata a felhasználók által feltöltött adatok pontos, és hibamentes kezelése. A felhasználót létre lehessen hozni, a felhasználónevet, jelszót, és emailt módosítani, és törölni. Regisztrációkor a rendszer egy megerősítő e-mailt kell küldjön a felhasználónak. A bejelentkezés megvalósítása JWT (JSON Web Tokenek)kel történik. A felhasználó adatai csak az adott felhasználó számára legyenek elérhetők. 2 “magasabb” felhasználói rang van: superuser (admin) és supervisor. Supervisor rangot e-mail írással lehet elérni, amit jóváhagyhatnak az adminisztrátori ranggal rendelkezők. Az élelmiszereket ne módosíthassa jóváhagyás nélkül egyetlen egy felhasználó. A létrehozások / módosítások / törlések jóváhagyását csak a „supervisor” és az e-feletti ranggal rendelkező felhasználók tehetik. Ugyan ez igaz a törlések jóváhagyására, és új élelmiszerek jóváhagyására. Új élelmiszer létrehozást / módosítást / törlést bármely felhasználó kezdeményezhet. Az élelmiszereknek tartalmazniuk kell a Magyarországon is használt tápérték táblázatot. Minden élelmiszerről lehet képet feltölteni. A hozzávalókat a már meglévő listából választhatják ki a felhasználók. A hozzávalóknak van egy veszélyességi szintjük, ami azt jelöli mennyire biztonságos az adott hozzávaló fogyasztása, kutatások szerint. A leírásba kutatások publikált linkjei is illeszthetők. Az éttermek és összetevők előre feltöltött adatokkal kell, hogy dolgozzanak. További rugalmasságot engedő tervezet a továbbfejlesztési tervek között van. Az alkalmazás bővítése esetén az éttermek városa és pontos helye is nagyobb jelenőséget kapna.

## Az alkalmazott fejlesztői eszközök

* A következő eszközöket használtuk a **Backend** implementálásához:
  + Python 3.12
  + Visual Studio Code
  + DB Browser for SQLite
  + Django
  + Djangohoz kapcsolódó csomagok:
    - djangorestframework
    - django-cors-headers
    - djangorestframework-simplejwt
  + Egyéb függőségek a requirements.txt fájlból:
    - python-dotenv (.env konfigurációk tesztelésére (ez a megerősítő e-mail kiküldéséhez szükséges)
    - pillow (képkezelés, függőségi problémák léphetnek fel nélküle)
    - factory\_boy (nagyobb tesztfolyamatok könnyítésére)
    - faker (a tesztadatok generálására)
    - faker-food (ételekkel való kiterjesztése a faker modulnak)
* A **Frontend** implementáláshoz:
  + Node.js 22.3
  + Visual Studio Code
  + React
  + Vite
  + react-router-dom
* Minden modul ingyenes letölthető csomagkezelők segítségével, és ingyenes használható nem-üzleti szoftverek fejlesztésére.

## Adatmodell leírása

A feladatban alkalmazott adatbázis relációs diagrammja:  


**Táblák összegzése és kapcsolataik:**

1. **User (Felhasználók):**
2. A rendszerben regisztrált felhasználókat tárolja e-mail cím alapján, valamint tartalmazza a belépési jogosultságokat és adminisztrációs szerepköröket. Kapcsolatok:
   1. **Food (Ételek):** Egy felhasználó több ételt is jóváhagyhat (approved\_supervisors).
   2. **Food (Ételek):** Egy felhasználó létrehozhat ételt (created\_by).
   3. **FoodChange (Ételváltozások):** Egy felhasználó frissítheti az ételeket (updated\_by).
   4. **FoodChange (Ételváltozások):** Jóváhagyhat ételváltozásokat (new\_approved\_supervisors).
   5. **ConfirmationToken (Megerősítő tokenek):** Egy felhasználóhoz több token is tartozhat.
3. **Restaurant (Éttermek):**

Az éttermek adatait tárolja, beleértve a nevüket, képüket, konyha típusukat és az ételeik veszélyességi szintjét. Kapcsolatok:

* 1. **Location (Helyszínek):** Egy étteremnek több helyszíne lehet.
  2. **Food (Ételek):** Egy étterem több ételt is kínálhat.
  3. **FoodChange (Ételváltozások):** Az ételváltozások egy étteremhez kötődnek (new\_restaurant).

1. **Location (Helyszínek):**

Egy étterem földrajzi elhelyezkedését tárolja hosszúsági és szélességi koordinátákkal. Kapcsolatok:

* 1. **Restaurant (Étterem):** Minden helyszín egy adott étteremhez tartozik.

1. **Ingredient (Hozzávalók):**

Az alapanyagokat és azok veszélyességi szintjét tárolja. Kapcsolatok:

* 1. **Food (Ételek):** Egy hozzávaló több ételben is szerepelhet, és egy ételben több hozzávaló is lehet.
  2. **FoodChange (Ételváltozások):** Az új ételek módosított hozzávalói is tárolásra kerülnek.

1. **Food (Ételek):**

Az ételek adatait tárolja, beleértve a hozzávalókat, makrotápanyagokat, allergén információkat és az elkészítő felhasználót. Kapcsolatok:

* 1. **Restaurant (Étterem):** Minden étel egy adott étteremhez tartozik.
  2. **Ingredient (Hozzávalók):** Egy étel több hozzávalóból állhat.
  3. **User (Felhasználók):** Egy felhasználó készítheti vagy jóváhagyhatja az ételt.
  4. **FoodChange (Ételváltozások):** Az ételek változásai itt kerülnek naplózásra.

1. **FoodChange (Ételváltozások):**

Az ételek változásait követi nyomon, beleértve a régi és az új adatokat, valamint a módosító felhasználót. Kapcsolatok:

* 1. **Food (Ételek):** Egy régi verzióra hivatkozhat az old\_version mezőben.
  2. **Restaurant (Étterem):** Az új ételváltozat egy adott étteremhez kapcsolódik.
  3. **Ingredient (Hozzávalók):** Az új változat hozzávalókat tartalmaz.
  4. **User (Felhasználók):** Egy felhasználó végrehajthatja a módosítást vagy jóváhagyhatja azt.

1. **ConfirmationToken (Megerősítő tokenek):**

A felhasználói fiókok hitelesítésére szolgáló tokeneket tárolja. Kapcsolatok:

* 1. **User (Felhasználók):** Minden token egy adott felhasználóhoz tartozik.

**Leírás, egyéb említésre méltó információ:**

* A legtöbb helyen egyértelmű az adatok közti reláció, és a mögöttes gondolatmenet is. (Például: Egy ételben lehet több hozzávaló is, és a hozzávalók tartozhatnak másik ételhez is.)
* Ebben benne vannak a Django által létrehozott adatmodellek is.
* A beépített, adminisztráció funkcionalitáshoz szükséges táblák:
  + Django\_admin\_log
  + Django\_session
  + Django\_migrations
  + Django\_content\_type
  + Users\_user\_permissions
  + Auth\_permissions
  + Auth\_group
  + Auth\_group\_permissons
* A User model az alapvető Djangos adatmodelltől öröklődik, de át lett konfigurálva hogy az e-mail legyen a meghatározó egyedi adat.
* A models.Model-ből öröklődve minden adatnak van egy id mezője.
* Djangoban rendkívül egyszerűen lehet modellek közti kapcsolatot megadni, a keretrendszer boilerplate sorok nélkül felismeri.
* A kapcsolótáblákat ManyToManyField generálja, úgyanúgy boilerplate nélkül, pl: Foods\_ingredients.
* Igazi JSON támogatás nincs, mivel a JSON adatformátum szöveges alapú, így textként tárolja az adatbázis is.
* Mivel az alkalmazás odafigyel az adatok integritására, ezért külön van választva a Food és a FoodChange modell. Egy foodnak több foodchange is lehet, mindig a legutóbb elfogadott fogja “átvenni” az eredeti food helyét.
* A locations csak a földrajzi helyet jelöli, ezen a későbbiekben lehet változtatni.
* Az is\_alcohol\_free, is\_gluten\_free, is\_organic, is\_lactose\_free boolean értékeket egyetlen egy bájttá össze lehetne vonni, viszont ez extra bitmanipulációval járni, ami növeli az alkalmazás hibalehetőségeit.
* Erre megoldás lehet egy enum ami kombinációkon alapul, viszont ennek a kiszámításához rengeteg manuális (értelmetlen) munkára lenne szükség
* A ranggal kapcsolatos flag-eket is össze lehetne vonni egyetlen enum-ba / bájtba de az előbb felsorolt okok miatt ez sem fog megörténni.
* A hazard\_level igazából számokat tárol, viszont ez a mező Djangoban enumként lett definiálva, tehát validálásra kerülnek a beleírt számok. 0-tól 3-ig lehet értéket adni ennek az enumnak.
* **Megjegyzés**: az adatbázisban vannak blank=true flaggel ellátott mezők is. Ez megnehezítheti a lekérdezést bizonyos szigorúan típusos nyelvekben, mivel az érték nem lesz semmi. Ilyenkor ajánlott a COALESCE kulcsszó használata. A COALESCE fügvény az első létező értéket returnöli, tehát üres érték nem lesz a lekérdezésben. Szinte bármennyi paramétere lehet ennek a függvénymeghívásnak.
* **Megjegyzés**: A képeknél elérési utat tárol a tábla. Ezeket képeket a Django-n keresztül statikus fájl szerverrel képes elérni a backend

## Algoritmusok

A program rengeteg előre megírt objektumot, algoritmust használ. Az összes model, és view az alapvető django-s modellek és view-k funkcionalitására alapszik. A valós problémák megoldására nem mindig szükséges új algoritmust kitalálni, vagy „újra feltalálni a kereket”.

Általánosságban a backend több algoritumust, függvényt igényel.

**Django REST Framework Nézetek Kulcsfontosságú Algoritmusai**

**Felhasználókezelési Algoritmusok**

1. Felhasználói Regisztráció és Email Megerősítés
   1. A **CreateUserView** osztály feladata az új felhasználók létrehozása és email-címük ellenőrzése:
   2. Először ellenőrzi és érvényesíti a felhasználói adatokat
   3. Létrehoz egy véletlenszerű megerősítési tokent
   4. A tokent eltárolja az adatbázisban a felhasználóhoz kapcsolva
   5. Formázott HTML emailt küld a felhasználónak a megerősítő linkkel
   6. Sikeres vagy hibaüzenetet ad vissza a kérés eredményétől függően
2. Email Megerősítési Mechanizmus
   1. A ConfirmEmail osztály a regisztrációs folyamat második lépését kezeli:
   2. Kivonja a tokent a kérésből
   3. Lekérdezi a tokent az adatbázisból
   4. Ellenőrzi a kapcsolódó felhasználót és inaktív státuszát
   5. Aktiválja a felhasználót és elmenti a változásokat
   6. Törli a tokent (egyszeri használat)
   7. Sikeres válasszal tér vissza
3. Felhasználói Hitelesítés SimpleJWT Implementációval
   1. A **CustomTokenObtainPairView** osztály biztosítja a hitelesítési mechanizmust:
   2. Egyedi JWT token generálást hajt végre
   3. Nem igényel előzetes hitelesítést (bejelentkezéshez szükséges)

**Étel Kezelési Algoritmusok**

1. Étel Létrehozás Automatikus Képletöltéssel
   1. A **FoodCreateView** osztály új ételek létrehozásáért felelős:
   2. Feldolgozza és érvényesíti a beérkező adatokat
   3. Létrehozza az étel rekordot az adatbázisban
   4. Ha nem töltöttek fel képet, automatikusan keres egyet az étel neve és az étterem neve alapján
   5. Kiszámítja a veszélyességi szintet az összetevők alapján
   6. Sikeres létrehozás esetén visszaadja az étel adatait, vagy hibaüzenetet küld
   7. Ezen kívül a perform\_create metódus:
      1. Beállítja a jóváhagyási státuszt és a létrehozót
      2. Ha a felhasználó supervisor, hozzáadja a jóváhagyók listájához
2. Étel Módosítási Javaslati Rendszer
   1. A **CreateFoodChange** osztály az ételek módosítási javaslatait kezeli:
   2. Ellenőrzi, hogy az étel létezik-e
   3. Kivonja és érvényesíti az új adatokat, beleértve az összetevőket
   4. Az értékeket megfelelő típusokra konvertálja (segédfunkciók használatával)
   5. Létrehoz egy étel változtatási rekordot a régi és új értékekkel
   6. Beállítja az összetevőket és kiszámítja az új veszélyességi szintet
   7. Sikeres vagy hibaüzenetet ad vissza
3. Étel Törlési Javaslati Rendszer
   1. A **CreateFoodRemoval** osztály az ételek törlési javaslataiért felelős:
   2. Azonosító alapján lekérdezi az ételt
   3. Ellenőrzi, hogy van-e már meglévő törlési javaslat
   4. Létrehoz egy étel változtatási rekordot törlési jelzővel
   5. Átmásolja a meglévő étel adatait a javaslatba
   6. Ha supervisor kéri a törlést, hozzáadja kezdeti jóváhagyását
   7. Sikeres vagy hibaüzenetet ad vissza

**Jóváhagyási Munkafolyamat Algoritmusok**

1. Étel Jóváhagyási Rendszer
   1. Az **AcceptFood** osztály a több supervisor által történő jóváhagyási folyamatot kezeli:
   2. Ellenőrzi, hogy a felhasználó supervisor-e
   3. Ellenőrzi, hogy a supervisor jóváhagyta-e már az ételt
   4. Hozzáadja a supervisort a jóváhagyók listájához
   5. Megszámolja a jóváhagyásokat és összeveti a küszöbértékkel
   6. Ha eléri a küszöböt, jóváhagyottként jelöli az ételt
   7. Frissíti az étterem veszélyességi szintjét
   8. Sikeres vagy hibaüzenetet ad vissza
2. Változtatási Javaslat Jóváhagyási Rendszer
   1. Az ApproveProposal osztály a változtatási javaslatok jóváhagyását kezeli:
   2. Lekérdezi az étel változtatási javaslatot
   3. Ellenőrzi, hogy a felhasználó supervisor-e
   4. Hozzáadja a supervisort a jóváhagyók listájához
   5. Összeveti a jóváhagyások számát a küszöbértékkel (2 szükséges)
   6. Ha eléri a küszöböt, jóváhagyottként jelöli a javaslatot
   7. Egy signal fogja kezelni a változtatások alkalmazását
   8. Sikeres vagy hibaüzenetet ad vissza

**Adatfeldolgozási Algoritmusok**

1. Típuskonverziós Segédprogram
   1. A convert\_value függvény intelligens típuskonverziót végez alapértelmezett értékekkel:
   2. Kezeli a logikai értékek konverzióját (a "true", "1", "yes" sztringek esetén)
   3. Kezeli az egész számok konverzióját (hiba esetén 0-ra állítja)
   4. Kezeli a lebegőpontos számok konverzióját (hiba esetén 0.0-ra állítja)
   5. Visszaadja az eredeti értéket, ha nincs szükség konverzióra
2. JSON Feldolgozó Segédprogram
   1. A parse\_json függvény biztosítja a megfelelő JSON formázást:
   2. Visszaadja az értéket, ha már szótár
   3. Megpróbálja a sztringet JSON-ként értelmezni
   4. Hiba esetén üres szótárat ad vissza

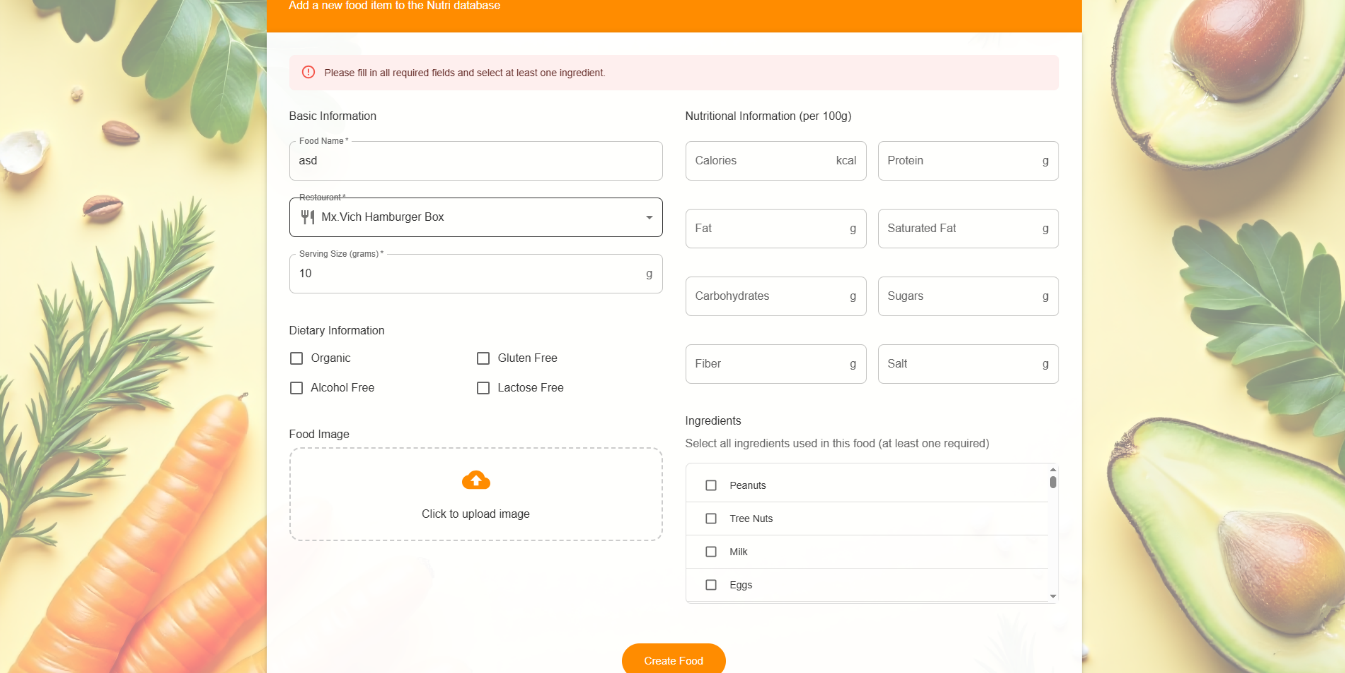
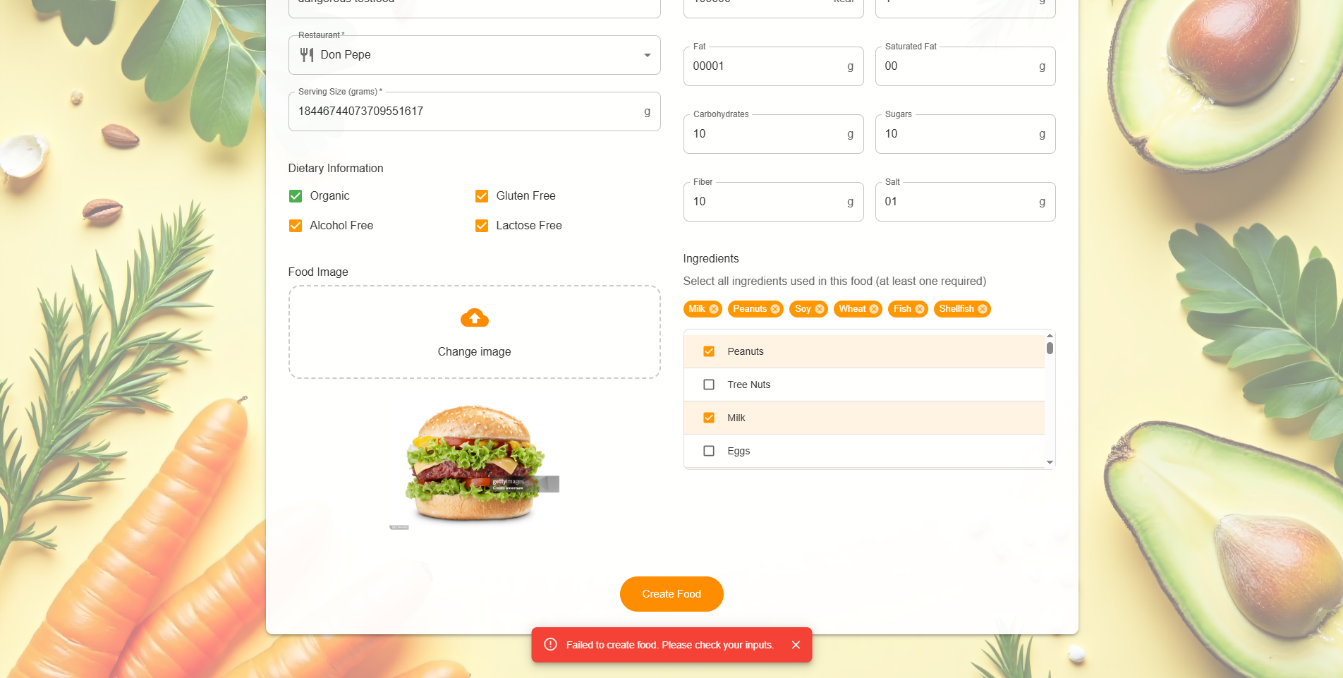
**Helyszínkezelési Algoritmusok**

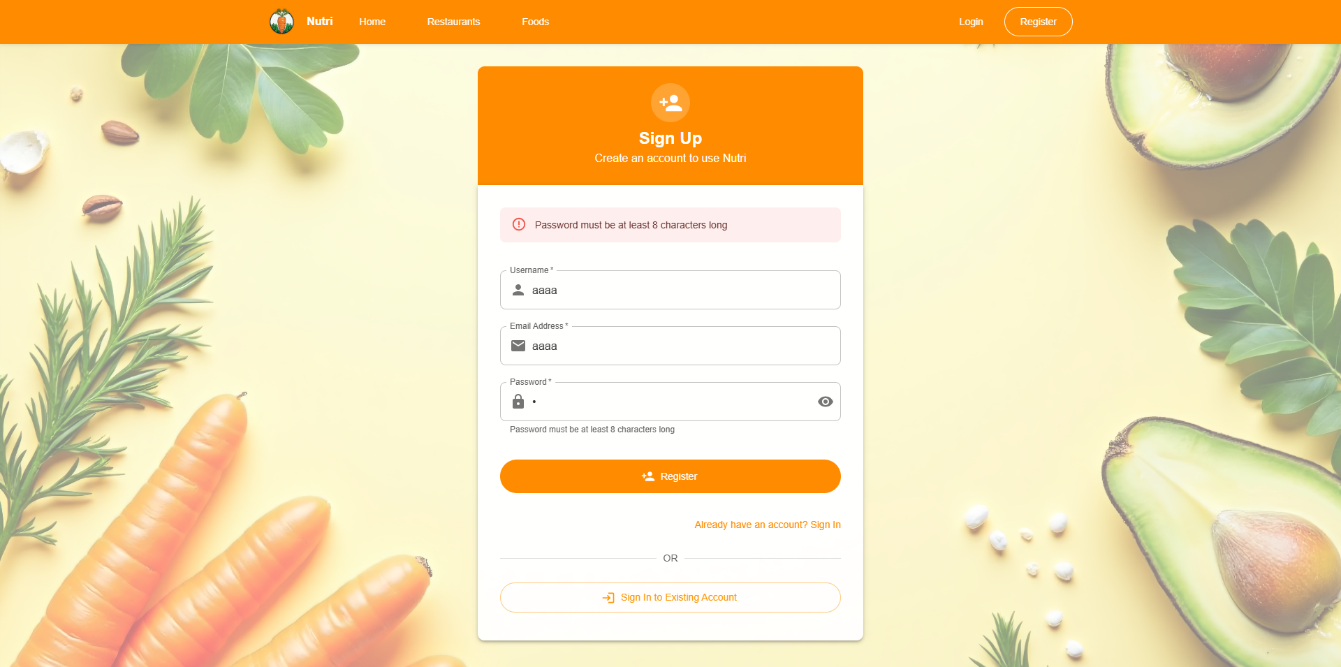
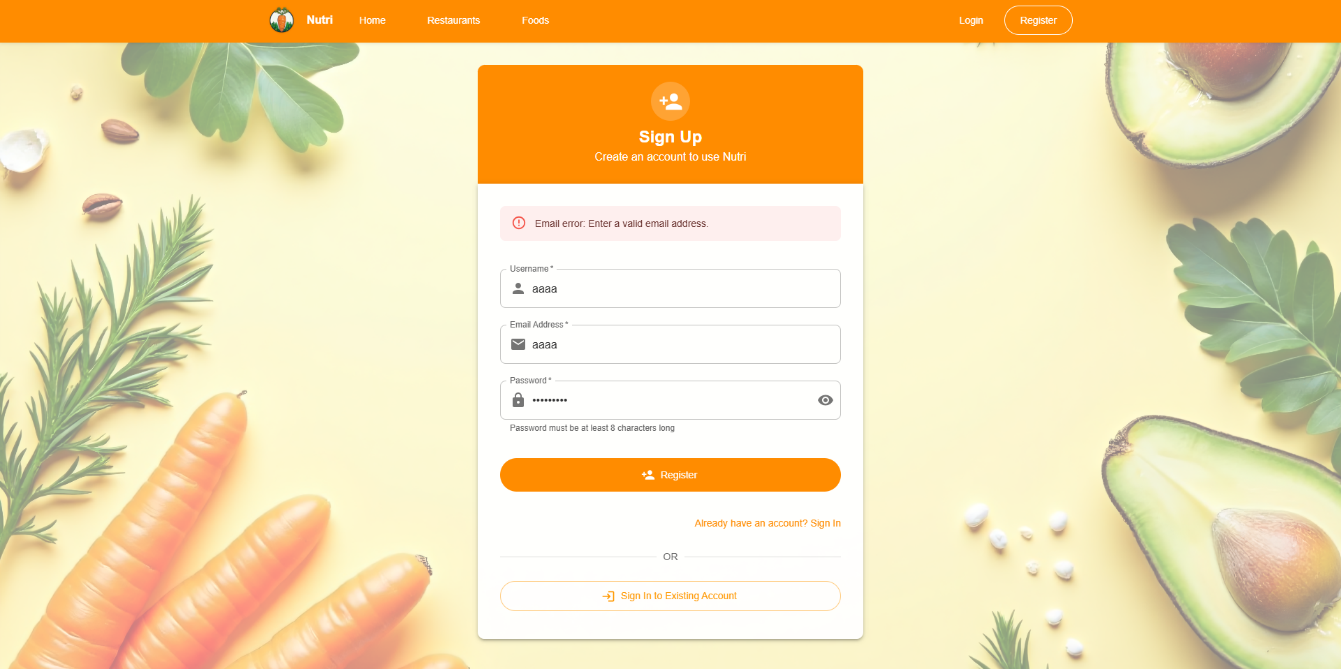
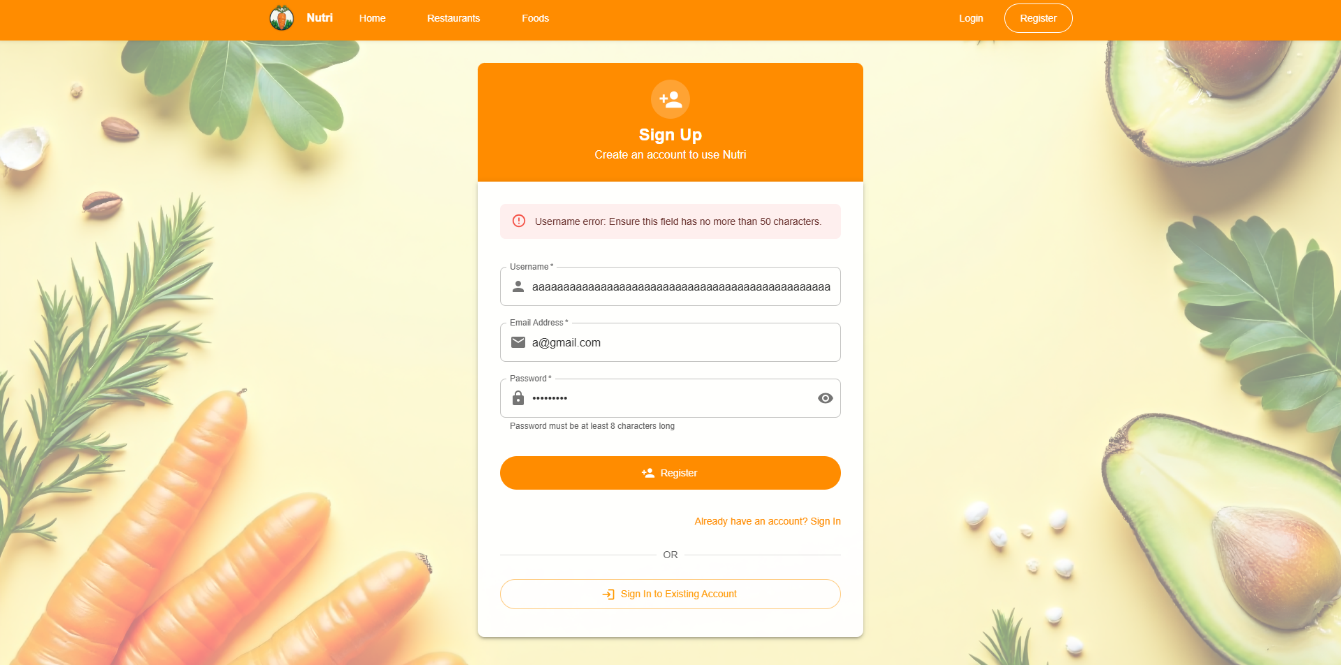
1. Étterem Létrehozása Helyszínnel
   1. A RestaurantWithLocationView osztály éttermet hoz létre helyszínnel együtt egy tranzakcióban:
   2. Érvényesíti az étterem adatait
   3. Létrehozza az étterem rekordot
   4. Hozzárendeli a helyszín adatokat
   5. Sikeres létrehozás esetén visszaadja az adatokat, vagy hibaüzenetet küld
2. Csoportos Étterem Feldolgozás
   1. A BatchSaveRestaurantsView osztály csoportos étterem adatok feldolgozását végzi előnézeti funkcióval:
   2. Ellenőrzi, hogy a bemenet lista-e
   3. Mintát vesz az első 5 étteremből
   4. Ellenőrzi, hogy az egyes éttermek léteznek-e már
   5. Előnézetet ad arról, mi történne mentés esetén
   6. A tényleges mentéshez egy kezelési parancsra irányít

**API Végpont Funkciók**

1. Étel Változtatási Javaslat Létrehozása
   1. A propose\_food\_change függvény étel változtatási javaslatot hoz létre a jelenlegi felhasználóval:
   2. Érvényesíti az adatokat a szerializáló segítségével
   3. Mentéskor beállítja az updated\_by mezőt a jelenlegi felhasználóra
   4. Sikeres vagy hibaüzenetet ad vissza

## Tesztelési dokumentáció

* Regisztráció, bejelentkezés, és kijelentkezés tesztelése:
  + - Amennyiben a felhasználó rákattint a kijelentkezés gombra, instant kijelentkezik. Az oldal erre reagálva a jogosultsági hibát jelzi a felhasználónak, ha olyan oldalon van, amihez bejelentkezés szükséges.
    - Ez más létező webapplikációkon is megtörténhet, de megváltoztatni elég egyszerű, szimplán vissza kell navigálni a főoldalra, ha nem található valid felhasználó / token
* Étel létrehozásának tesztelése:
  + - A weboldal hibát dob, ha egy szám-alapú beviteli mező nagyobb, mint az előjeles 64 bites egész számok maximuma.
    - Minden ételnek kell minimum egy összetevő, ezért ha nincs hozzáadva egy sem, ilyenkor is dob hibát az oldal.
    - Ha negatív számot ír be a felhasználó, szimplán átkonvertálja a frontend pozitívvá a request elküldése előtt
* **Ételek létrehozásának tesztelése képekben:**
* **Bolondbiztosság a regisztrációkor:**



* A tesztelés során kiderült hibák:
  + Amikor az e-mail megerősítő rendszer került tesztelésre, kiderült, hogy a ConfirmEmail.tsx URL konfigurációjában: code volt a megerősítő kód neve, az oldalon pedig egy token nevű paramétert várt.
  + A logóra kattintva visszadob a főmenüre, ez viszont nem történt meg a „Nutri” szóra való kattintás után.
* Mi a feketedobozos tesztelésre fókuszáltunk, mert ez dönti el, hogy az alkalmazás használható-e, és kívülről megtörhető-e.
* A django jelentősen megkönnyíti a unit és integrációs tesztek írását, szóval a fehérdobozos tesztelés sem lenne nehéz.
* Egyéb megjegyzés: a felhasználó nem tud elérni magasabb rangú oldalakat sem, akkor se ha átlagfelhasználói profillal van belépve, és beírja a helyes URL-t a böngésző címsorába

# Felhasználói dokumentáció

## A program általános specifikációja, célja

Ez a rész a program fontosabb jellemzőit és funkcióit tartalmazza. A cél, hogy a leendő felhasználó ezt a fejezetet elolvasva el tudja dönteni, hogy a program megfelelő-e a számára.

Ajánlott terjedelem: ½ -1 oldal.

## Rendszerkövetelmények

Hardver követelmények

Ebben a részben kell leírni a minimális és ajánlott hardver konfigurációt, amely a program futtatásához szükséges. Pontos paramétereket kell megadni, még akkor is, ha a program amúgy minden gépen lefut.

Szoftver követelmények

Le kell írni, hogy mely operációs rendszere(ke)n fut a program, és milyen egyéb szoftver komponensek szükségesek a működéshez (pl. .NET, DirectX, esetleg adatbázis-szerver és adatbázis-állományok, stb.)

Ajánlott terjedelem: ½ -1 oldal, felsorolás szerűen leírva

## A program telepítése

• Képekkel illusztrált, részletes leírás a program telepítésének a menetéről.

• A leírás alapján a felhasználónak hiba nélkül telepíteni kell tudni a programot. A leírásnak ki kell térnie a telepítés során kiválasztható opciókra is.

• Ha esetleg nincs telepítőprogram, akkor kellő részletességgel le kell írni, hogy mely fájlokat, pontosan hova kell felmásolni, és hogy lehet a programot futtatni.

Ajánlott terjedelem: 1-3 oldal, ábrákkal együtt.

## A program használatának a részletes leírása

Mindenre kiterjedő, részletes leírás a program használatáról.

Alapszabályok:

• Amit leprogramoztál, azt a dokumentációban írd is le, ne legyenek eltitkolt funkciók.

• Minden pontosan, „szájbarágósan” legyen leírva. A dokumentáció alapján a teljesen kezdő, vagy laikus felhasználóknak is használniuk kell tudni a programot.

• A stílus legyen pontos és közérthető, vedd figyelembe, hogy a felhasználói dokumentáció nem szakembereknek készül.

• Ugyanakkor kerüld a laza stílust: rövidítések, smilie-k, szleng kizárva.

• Alkalmazz ábrákat, screenshot-okat , de a ne legyen túlzott a képek aránya a szöveghez képest.

Ajánlott terjedelem: 1-5 oldal

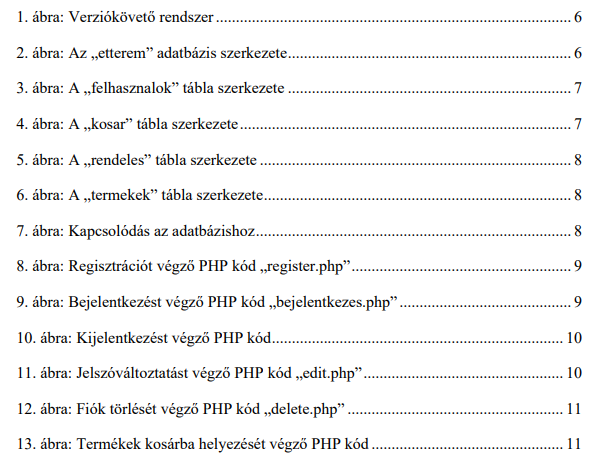
# Összegzés

Elérte-e a kitűzött célt, milyen további fejlesztési lehetőségeket lát.

* Olyan ötletek, amelyeket meg akartál valósítani, de nem sikerült, vagy nem fért bele az időbe
* Olyan ötletek, amelyeket még érdemes a jövőben megvalósítani

# Ábrajegyzék

Dokumentációban szereplő ábrák felsorolása



# Irodalomjegyzék, hivatkozásjegyzék

Felhasznált szakirodalom és webes hivatkozások felsorolása!!!!!

pl:  
