

# **ENERGETIKAI TECHNIKUM ÉS KOLLÉGIUM**

## **Vizsgaremek**

### **Nutri applikáció**

**2025.**

**Szakma:**  
**SZOFTVERFEJLESZTŐ ÉS -TESZTELŐ**  
**5 0613 12 03**

**Készítették:**  
**Aczélos Csaba**  
**Balogh Csabád**  
**Rummler János**

# Tartalomjegyzék

I.	Bevezetés.....	4
II.	Témaválasztás indoklása .....	4
III.	Fejlesztői dokumentáció, technikai leírás.....	4
1.	Specifikáció .....	4
2.	Az alkalmazott fejlesztői eszközök .....	5
3.	Adatmodell leírása.....	6
4.	Algoritmusok.....	11
5.	Tesztelési dokumentáció.....	15
IV.	Felhasználói dokumentáció .....	19
1.	A program általános specifikációja, célja.....	19
2.	Rendszerkövetelmények.....	19
3.	A program telepítése.....	20
4.	A program használatának a részletes leírása .....	21
V.	Összegzés .....	22
VI.	Ábrajegyzék.....	23
VII.	Irodalomjegyzék, hivatkozásjegyzék .....	23

## 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  
Balogh Csanád  
Rummler János  
Név

## **I. 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.

## **II. 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.

## **III. Fejlesztői dokumentáció, technikai leírás**

### **1. 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 kérvényezni kell a support felületen keresztül, 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

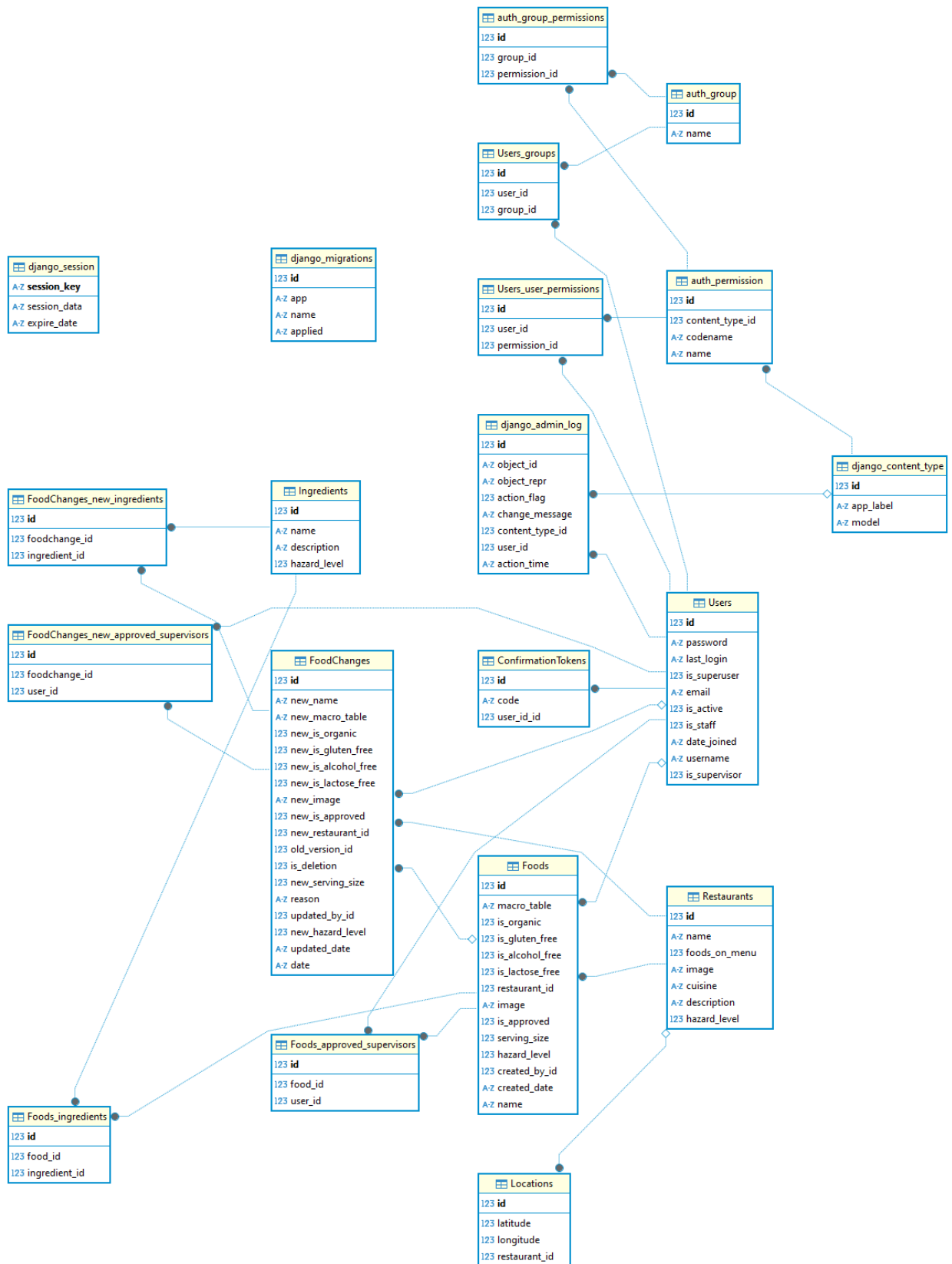
éttermek és összetevők előre feltöltött adatokkal kell, hogy dolgozzanak. További rugalmasságot engedő tervezet a továbbfejlesztési tervek között van. Az alkalmazás bővítése esetén az éttermek városa és pontos helye is nagyobb jelentőséget kapna.

## 2. Az alkalmazott fejlesztői eszközök

- A következő eszközöket használtuk a **Backend** implementálásához:
  - Python 3.12
  - Visual Studio Code
  - DB Browser for SQLite
  - Django
  - Django-hoz 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 (képfeldolgozás, függőségi problémák léphetnek fel nélküle)
    - factory\_boy (nagyobb tesztfolyamatok könnyítésére)
    - faker (a tesztadatok generálására)
    - faker-food (ételekkel való kiterjesztése a faker modulnak)
- A **Frontend** implementálásához:
  - 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.

### 3. 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:

- a. **Food (Ételek):** Egy felhasználó több ételt is jóváhagyhat (approved\_supervisors).
- b. **Food (Ételek):** Egy felhasználó létrehozhat ételt (created\_by).
- c. **FoodChange (Ételváltozások):** Egy felhasználó frissítheti az ételeket (updated\_by).
- d. **FoodChange (Ételváltozások):** Jóváhagyhat ételváltozásokat (new\_approved\_supervisors).
- e. **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:

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

#### 4. Location (Helyszínek):

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

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

#### 5. Ingredient (Hozzávalók):

Az alapanyagokat és azok veszélyességi szintjét tárolja. Kapcsolatok:

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

#### 6. Food (Ételek):

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

- a. **Restaurant (Étterem):** Minden étel egy adott étteremhez tartozik.
- b. **Ingredient (Hozzávalók):** Egy étel több hozzávalóból állhat.
- c. **User (Felhasználók):** Egy felhasználó készítheti vagy jóváhagyhatja az ételt.
- d. **FoodChange (Ételváltozások):** Az ételek változásai itt kerülnek naplózásra.

#### 7. FoodChange (Ételváltozások):

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

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

#### 8. ConfirmationToken (Megerősítő tokenek):



A felhasználói fiókok hitelesítésére szolgáló tokeneket tárolja. Kapcsolatok:

- a. **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\_permissions
- A User model az alapvető Django adatmodellről 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.
- Django-ban rendkívül egyszerűen lehet modellek közti kapcsolatot megadni, a keretrendszer boilerplate sorok nélkül felismeri.
- A kapcsolótáblákat ManyToManyField generálja, ugyanúgy 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 bájtta 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 megőrténni.
- A `hazard_level` igazából számokat tárol, viszont ez a mező Djangóban 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 függvény 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

## 4. 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 algoritmust, függvényt igényel.

### Django REST Framework Nézetek Kulcsfontosságú Algoritmusai

#### Felhasználókezelési Algoritmusok

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

#### Étel Kezelési Algoritmusok

1. Étel Létrehozás Automatikus Képletöltéssel
  - a. A **FoodCreateView** osztály új ételek létrehozásáért felelős:
  - b. Feldolgozza és érvényesíti a beérkező adatokat
  - c. Létrehozza az étel rekordot az adatbázisban
  - d. Ha nem töltöttek fel képet, automatikusan keres egyet az étel neve és az étterem neve alapján
  - e. Kiszámítja a veszélyességi szintet az összetevők alapján
  - f. Sikeres létrehozás esetén visszaadja az étel adatait, vagy hibaüzenetet küld
  - g. Ezen kívül a `perform_create` metódus:
    - i. Beállítja a jóváhagyási státuszt és a létrehozót
    - ii. Ha a felhasználó supervisor, hozzáadja a jóváhagyók listájához

2. **Étel Módosítási Javaslati Rendszer**
  - a. A **CreateFoodChange** osztály az ételek módosítási javaslatait kezeli:
  - b. Ellenőrzi, hogy az étel létezik-e
  - c. Kivonja és érvényesíti az új adatokat, beleértve az összetevőket
  - d. Az értékeket megfelelő típusokra konvertálja (segédfunkciók használatával)
  - e. Létrehoz egy étel változtatási rekordot a régi és új értékekkel
  - f. Beállítja az összetevőket és kiszámítja az új veszélyességi szintet
  - g. Sikeres vagy hibaüzenetet ad vissza
3. **Étel Törlési Javaslati Rendszer**
  - a. A **CreateFoodRemoval** osztály az ételek törlési javaslataiért felelős:
  - b. Azonosító alapján lekérdezi az ételt
  - c. Ellenőrzi, hogy van-e már meglévő törlési javaslat
  - d. Létrehoz egy étel változtatási rekordot törlési jelzővel
  - e. Átmásolja a meglévő étel adatait a javaslatba
  - f. Ha supervisor kéri a törlést, hozzáadja kezdeti jóváhagyását
  - g. Sikeres vagy hibaüzenetet ad vissza

## **Jóváhagyási Munkafolyamat Algoritmusok**

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

## 1. ApproveProposal osztály

```
class ApproveProposal(generics.UpdateAPIView):
    queryset = FoodChange.objects.all()
    serializer_class = FoodChangeSerializer
    permission_classes = [IsAuthenticated]

    def patch(self, request, *args, **kwargs):
        # Retrieve the FoodChange instance
        food_change = get_object_or_404(FoodChange, id=kwargs.get('pk'))

        # Check if the user is a supervisor
        if not request.user.is_supervisor:
            logger.warning(
                f"User {request.user.username} attempted to approve change but is not a supervisor")
            return Response({"error": "Only supervisors can approve food changes."}, status=status.HTTP_403_FORBIDDEN)

        # Add the supervisor to the new_approved_supervisors
        food_change.new_approved_supervisors.add(request.user)
        logger.info(
            f"Supervisor {request.user.username} approved food change #{food_change.id} for {food_change.new_name}")

        # Get the current count of approvals
        approval_count = food_change.new_approved_supervisors.count()

        # Check if the FoodChange has enough approvals to be marked as approved
        # Lower from 20 to a more reasonable number (adjust as needed)
        required_approvals = 2
        logger.info(
            f"Food change #{food_change.id} has {approval_count}/{required_approvals} approvals")

        if approval_count >= required_approvals:
            logger.info(
                f"Food change #{food_change.id} has reached approval threshold!")
            try:
                # Simply mark as approved; the signal will handle applying the changes
                food_change.new_is_approved = True
                food_change.save()
                logger.info(
                    f"Marked food change #{food_change.id} as approved")
            except Exception as e:
                # Log any errors during the approval process
                logger.error(
                    f"Error approving food change #{food_change.id}: {e}")
                logger.error(traceback.format_exc())
                return Response(
                    {"error": f"Error approving food change: {str(e)}"},
                    status=status.HTTP_500_INTERNAL_SERVER_ERROR
                )

        return Response({"message": "Food change approval recorded successfully."}, status=status.HTTP_200_OK)
```

## Adatfeldolgozási Algoritmusok

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

## **Helyszínkezelési Algoritmusok**

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

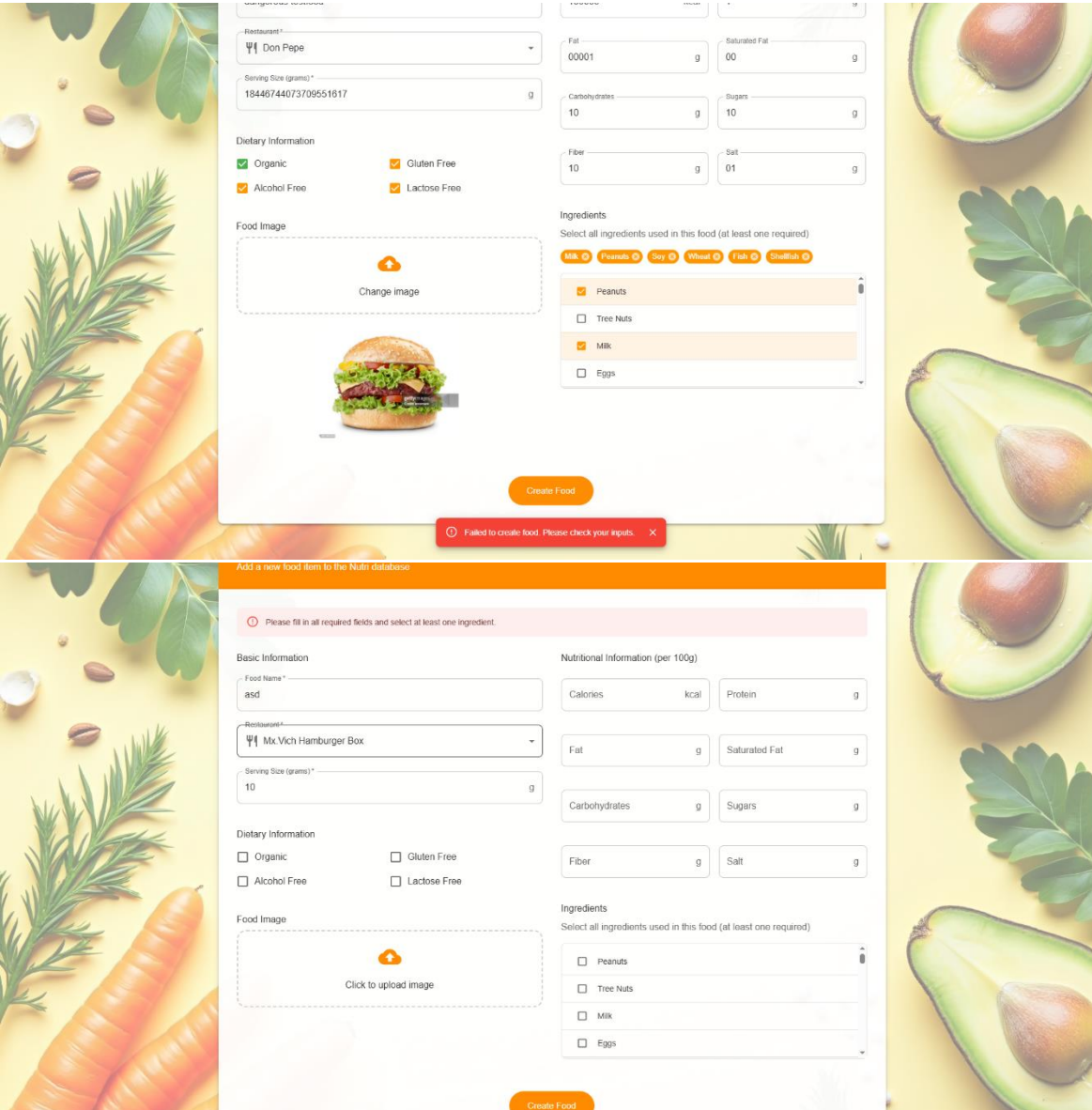
## **API Végpont Funkciók**

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

## 5. 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

## 2. Ételek létrehozásának tesztelése képekben



**Top Screenshot: Successful Food Creation**

Restaurant: **Don Pepe**

Serving Size (grams): **18446744073709551817** g

**Dietary Information:**

- ☒ Organic
- ☒ Gluten Free
- ☒ Alcohol Free
- ☒ Lactose Free

**Food Image:**

Change image

**Ingredients:**

Select all ingredients used in this food (at least one required)

**Ingredients:**

- ☒ Peanuts
- ☐ Tree Nuts
- ☒ Milk
- ☐ Eggs

**Bottom Screenshot: Failed Food Creation**

Add a new food item to the Nutri database

Please fill in all required fields and select at least one ingredient.

**Basic Information**

Food Name: **asd**

Restaurant: **Mx Vch Hamburger Box**

Serving Size (grams): **10** g

**Dietary Information:**

- ☐ Organic
- ☐ Gluten Free
- ☐ Alcohol Free
- ☐ Lactose Free

**Food Image:**

Click to upload image

**Nutritional Information (per 100g)**

Calories: **00001** kcal

Protein: **00** g

Fat: **10** g

Saturated Fat: **10** g

Carbohydrates: **10** g

Sugars: **10** g

Fiber: **10** g

Salt: **01** g

**Ingredients:**

Select all ingredients used in this food (at least one required)

**Ingredients:**

- ☐ Peanuts
- ☐ Tree Nuts
- ☐ Milk
- ☐ Eggs



### 3. Bolondbiztosság regisztrációkor

The image displays three sequential screenshots of a web application's sign-up form, illustrating different validation error messages. The form is titled "Sign Up" and includes the subtitle "Create an account to use Nutri". The background features a vibrant, flat-lay photograph of fresh ingredients: carrots, avocados, and various herbs on a light yellow surface.

The form fields and their associated error messages are as follows:

- Username:** The first screenshot shows an error message: "Username error. Ensure this field has no more than 50 characters." The input field contains a long string of 'a's.
- Email Address:** The second screenshot shows an error message: "Email error. Enter a valid email address." The input field contains the text "aaaa".
- Password:** The third screenshot shows an error message: "Password must be at least 8 characters long." The input field contains a single character.

Each screenshot also shows the "Register" button, a link to "Sign In" for existing users, and a "Sign In to Existing Account" button. The navigation bar at the top of each screenshot includes links for "Nutri", "Home", "Restaurants", "Foods", "Login", and "Register".

❖ **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

## IV. Felhasználói dokumentáció

### 1. A program célja

A Nutri applikáció célja, hogy segítse a felhasználókat a számukra legmegfelelőbb ételek és éttermek kiválasztásában, táplálkozási szokásaik és földrajzi elhelyezkedésük alapján. Az alkalmazás átláthatóan jeleníti meg az ételek összetevőit, makrotápanyag-tartalmát, allergén-információkat, és veszélyességi szintjeit. Lehetőséget nyújt a felhasználóknak arra is, hogy új ételeket töltsenek fel, javaslatokat tegyenek, vagy hozzájáruljanak a már meglévő adatbázis bővítéséhez. A rendszer célja az átláthatóság, a közösségi adatrögzítés támogatása, valamint a felhasználók táplálkozási döntéseinek elősegítése.

#### Hardver követelmények

- A Nutri applikáció egy webalapú rendszer, így külön telepítésre nincs szükség.
- Futtatásához egy modern számítógépre, laptopra vagy okoseszközre van szükség, amely képes korszerű webböngészőket használni.
- Minimum 2 GB RAM és 1 GHz-es processzor javasolt a zökkenőmentes működéshez.

#### Szoftver követelmények

- **Böngészők:** Google Chrome, Mozilla Firefox, Microsoft Edge, Safari (legfrissebb verzió ajánlott).
- **Operációs rendszer:** Windows 10/11, macOS, Linux disztribúciók, Android vagy iOS.
- **Internetkapcsolat:** stabil online kapcsolat szükséges az adatbázis eléréséhez.
- **E-mail fiók:** a regisztráció megerősítéséhez aktív e-mail fiók szükséges (pl. Gmail, Freemail, Outlook stb.).

## 2. A program telepítése

A Nutri alkalmazás **nem igényel telepítést**, mivel egy webes szolgáltatásról van szó. Az alábbi lépések elvégzésével a felhasználó azonnal elkezdheti használni a programot:

- **Nyisd meg a webböngészőt.**
- Írd be a Nutri alkalmazás elérhetőségét a címsorba (mivel nincs hostolva - `https://localhost:8000/`)
- A főoldalon kattints a **„Register”** gombra. (Jobb felső sarok)
- Töltsd ki a regisztrációs űrlapot: e-mail cím, jelszó, felhasználónév.
- Kattints a „Register” gombra. (középen)
- Ellenőrizd az e-mail fiókot: egy megerősítő e-mailt kapsz, amelyben található egy aktiváló link.
- Kattints a linkre az aktiváláshoz – inntől kezdve be tudsz jelentkezni.

### 3. A program használatának leírása

Ajánlott:

1. **Regisztrálj**
2. **Aktiváld** a fiókot
3. **Jelentkezz be** a jelszóddal és e-mail fiókkal
4. Innentől **megnézhetsz bármilyen éttermet**, és láthatod az ételeik tápanyagtartalmát.
5. Kezdeményezhetsz étel **létrehozást és törlést** is.

A program kezelői felülete nagyon intuitív, B1-es angoltudással nagyjából mindent érthet az ember, és aki már használt bármilyen webes felületet, könnyen tudná használni.

Ha nem regisztrálsz, nem tudsz ételeket / változtatásokat létrehozni.

Ha találsz olyan éttermet, ami nincs fent a listán, írd a supportnak.

Bejelentkezés nélkül bármelyik éttermet meg lehet nézni, a tápértéktartalmukat láthatod.

Új alapanyagokat az üzemeltető feladata hozzáadni, de ezért is lehet írni a supportnak.

Ételeket tudsz létrehozni a már létező éttermeknél.

A létrehozásnál az oldalon látott összes szempontot ki kell tölteni a hibakód mentes létrehozásért.

Ez módosítási javaslatra is vonatkozik. Ott viszont meg kell adni egy okot is a módosításra.

A törlést is lehet kezdeményezni.

A létrehozást, módosítást, és törlést csak a „supervisor” (felügyelő) tagok tudják megszavazni. Mindkettőhöz 10 szavazat szükséges.

A jelszavaknak legalább 8 karakterből kell állniuk, és tartalmazniuk kell nagybetűt, kisbetűt és számot a megfelelő biztonság érdekében.

A Supervisor rangot kérelmezéskor indoklással kell ellátni, hogy miért szeretné a felhasználó ezt a magasabb jogosultsági szintet.

Az ételkeresés funkció használható az étel neve, összetevők vagy tápanyagtartalom alapján is, a keresőmező segítségével.

## Összegzés

A Nutri projekt MVP (Minimum Viable Product) állapotban van: a legalapvetőbb funkciók működnek, a rendszer használatra kész. A további fejlesztések során a többnyelvűség és a nemzetközi ételadatbázisok támogatása kerülhet fókuszba. A jövőbeni bővítési lehetőségek közé tartozik a mobilalkalmazás fejlesztése, gépi tanulási alapú veszélyességi előrejelzések bevezetése, valamint a felhasználói szokások alapján történő ételajánlás. Továbbfejlesztési lehetőségek:

- Több ország és nyelv támogatása (Ehhez új táblákra van szükség).
- Elévülés implementációja (1 hónapnál öregebb módosítási javaslatok automatikus törlése)
- Fájlszerver létrehozása kizárólag a képeknek
- Load Balancerek alkalmazása a skálázhatóságért
- Összes vallási korlátozás támogatása
- Életkor és egészségügyi állapot alapján is lehetne ajánlásokat bevezetni

Összességében a projekt egy korszerű, valós problémára reflektáló megoldást kínál, amely a későbbiekben akár kereskedelmi irányba is továbbvihető, megfelelő fejlesztések és támogatás mellett.

## **V. Ábrajegyzék**

1. ApproveProposal osztály.....	13
2. Ételek létrehozásának tesztelése képekben.....	16
3. Bolondbiztosság regisztrációkor.....	17

## **VI. Irodalomjegyzék, hivatkozásjegyzék**

- <https://docs.djangoproject.com/en/5.2/>
- <https://react.dev/reference/react>
- <https://chatgpt.com/>
- <https://claude.ai/>
- <https://www.youtube.com/watch?v=PtQiiknWUcI>
- <https://www.youtube.com/watch?v=xjMP0hspNLE>
- <https://www.youtube.com/watch?v=s2skans2dP4>