



VAS VÁRMEGYEI SZAKKÉPZÉSI CENTRUM  
HORVÁTH BOLDIZSÁR  
KÖZGAZDASÁGI ÉS INFORMATIKAI  
TECHNIKUM

A 5 0613 12 03 számú Szoftverfejlesztő és –tesztelő vizsgaremek

**A szoftveralkalmazás dokumentációja**

Készítették:

TANDL ZSANETT

DANCSECS MÁTÉ

SOMOGYI BENJÁMIN

**SZOMBATHELY**

**2025**

# 1.Bevezetés

## 1.1 A program célja

Programunk céljai közé tartozik, hogy egy letisztult, mindenki számára könnyen kezelhető felületet biztosítson receptek kereséséhez és feltöltéséhez. Oldalunk segítséget biztosít abban, hogy a felhasználó az otthon megszokott ízek mellett más, számára ismeretlen konyhák ízvilágát is megismerje.

Napjainkban sokan küzdenek különböző ételallergiával amire az oldalunk segítséget nyújthat, hiszen az oldalunkon megjelenített recepteket kategorizáljuk és amennyiben a felhasználó igényt tart rá, akkor az oldalon nem jelennek meg olyan receptek, amelyek olyan hozzávalót tartalmaznak, amelyre ő érzékeny lehet.

Egy másik probléma, amire az oldalunk megoldást nyújt az az, hogy otthon, a rendelkezésre álló alapanyagok alapján az oldal recepteket ajánl. Ez hasznos, ha valamit gyorsan otthon kell főzni vagy pedig váratlanul vendégek érkeznek hozzánk.

## 1.2 Felhasználási terület – Célközönség

A szoftver célközönsége széleskörű, mivel minden olyan felhasználó számára hasznos lehet, aki érdeklődik a főzés iránt, szeret új recepteket kipróbálni, illetve személyre szabott étkezési lehetőségeket keres. Az oldal kifejezetten azoknak a felhasználóknak szól, akik:

- **Otthoni főzésre keresnek ötleteket:** Legyen szó gyors napi étkezésekről vagy különleges alkalmakra szánt fogásokról, a program segít megtalálni a megfelelő receptet.
- **Különböző ételallergiákkal rendelkeznek:** Az allergiás vagy érzékeny felhasználók számára lehetőség van a receptek szűrésére, hogy olyan étkezéseket találjanak, amelyek nem tartalmaznak számukra problémás alapanyagokat.
- **Új konyhák és ízek iránt érdeklődnek:** A program segíthet a felhasználóknak abban, hogy megismerjék más országok, kultúrák étkezési hagyományait és receptjeit.

## 2.A szoftver működése

Recepteket regisztrált felhasználók tölthetnek fel, ahol beállíthatják, hogy mennyi időbe telik az általuk feltöltött étel elkészítése, annak lépéseit, illetve az elkészítéséhez szükséges alapanyagokat. Az alapanyagokat adatbázisba gyűjtöttük ki és a receptek feltöltése során ezek közül lehet választani. A különféle hozzávalók különböző allergén osztályokba is sorolva vannak. Amennyiben a felhasználó érzékeny valamely allergén(ek)re – és ezt a regisztrációkor be is jelölte –, az oldal automatikusan nem ajánl olyan recepteket, amely az adott allergén osztály(ok)ba tartozik.

Az oldalra felkerülő receptek az admin joggal rendelkező felhasználók ellenőrzik, módosíthatják, elfogadhatják vagy elutasíthatják. Az oldalon nem jelennek meg receptek mindaddig, amíg azt egy admin jogú felhasználó el nem fogadja. Ehhez az oldal biztosít számukra egy felületet, ahol látják az összes elfogadásra váró receptet.

- **Receptek ellenőrzése és módosítása:** A receptek részleteit, mint például a nevét, leírását, hozzávalóit és azok mértékegységeit szükség esetén képesek megváltoztatni.
- **Receptek elutasítása vagy elfogadása:** Ha egy recept nem felel meg az oldal elvárásainak – nem az oldalra illő kép, leírás vagy név megadása esetén – akkor az admin felhasználók elutasíthatják az adott receptet, ezáltal pedig az nem jelenik meg a többi felhasználó számára, mind emellett az adatbázisból is törlésre kerül. Amennyiben a recept megfelel minden szempontból, akkor az admin elfogadhatja azt és az adott recept pedig megjelenik a többi felhasználó számára is.

A moderációs rendszer biztosítja, hogy az oldalon csak minőségi és megfelelő tartalom jelenjen meg, így a felhasználók megbízható recepteket találhatnak az oldalon.

## 3. Komponenseinek technikai leírása

### 3.1 A backend komponens működése

#### 3.1.1 Fő funkciója

Projektünkben a kliens oldali alkalmazás és az adatbázis közti kapcsolatot a backend szerver valósítja meg. Ez felelős azért, hogy a kliens csakis ellenőrzött módon kommunikálhasson az adatbázissal, ezáltal pedig az nem képes kárt okozni benne.

Ezen komponens definiálja, hogy az adatbázisból milyen adatok kérhetőek, illetve törölhetőek le, ezentúl pedig, hogy mely adatok tölthetőek fel és módosíthatóak.

#### 3.1.2 Technológiák alkalmazása

Projektünk kettő backend alkalmazást is tartalmaz, melyek szorosan együttműködnek egymással.

A kliens oldali alkalmazás elsősorban az ASP.NET alapú EntityFramework 6 keretrendszerben, a C# nyelv használatával megírt backend programmal kommunikál. Ez valósítja meg a teljes kapcsolatot az adatbázis és a kliens között. Ezen backend tartalmazza a megszorításokat és a kapcsolatokat az adatbázis egyedei között, illetve ez határozza meg a különböző táblák végpontjainak működését is.

Munkánk második backendje egy Python-ban megírt, a Flask keretrendszeren alapuló program, amely a képek feltöltését valósítja meg. Ezen alkalmazás feladata, hogy a kliens oldalon feltöltött képek neveit egy egyedi számmal lássa el, mely az éppen aktuális szerver idő lekérésével történik. – Lásd: 1.kép – Ez teszi lehetővé azt, hogy nem tud két felhasználó ugyanolyan névvel file-t feltölteni, ugyanis ezen szám miatt minden file egyedi. Mindezen felül a program feladata közé tartozik az is, hogy a képeket az erre a célra létrehozott frontend map-pában elhelyezze. A kép elérésének elhelyezése az adatbázisba pedig csakis mindazok után történik, miután a backend sikeresen végrehajtotta az összes feladatát.

```
#Egyedi nev generalasa
timestamp = int(time.time())
file_extension = os.path.splitext(original_filename)[1]
unique_filename = f"{timestamp}_{original_filename}"

file_path = os.path.join(app.config['UPLOAD_FOLDER'], unique_filename)
```

1. kép: Egyedi file név megvalósítása

### 3.1.3 Végpontok működése

#### 3.1.3a Általános bementi és válasz formátumok

A szerver oldali alkalmazásunk, mely az adatbázissal kommunikál, összesen 16 egyed lekérpezését, illetve a köztük lévő különböző – 1:1; 1:N; illetve N:M – kapcsolatokat tartalmazza. Minden végpont minden metódusa a kérések során egy HTTP státuszkódot, illetve egy JSON formátumú body választ küld. – Lásd.: 2. és 3. kép –

```
{  
  "Kid": 1,  
  "Nev": "Leves"  
}
```

2. kép : JSON formátumú válasz a Kategória tábla GET metódusánál

```
200
```

3. kép: Sikeres HTTP státuszkód

A különböző végpontok bemenetei egyediek, azonban a közös mindegyikben az, hogy egy JSON formátumú body-t várnak, eseteként pedig 1, de maximum 2 paramétert az elérési útban. – Lásd.: 4. és 5. kép –

```
{  
  "Nev": "string"  
}
```

4. kép: JSON formátumú body a Kategória tábla PUT metódusánál

```
'https://localhost:44350/api/Recept_Hozzavalo?Rid=1&HozzavaloNev=Hagyma'
```

5. kép: 2 paraméter fogadása a Recept\_Hozzávaló tábla PUT metódusánál

### 3.1.3b Metódusok általános működése

A következő API metódus leírások általánosságban érvényesek, azonban egyes eljárások működése eltérő lehet. Részletes leírás az egyes metódusokról a „**3.1.3c Metódusok részletes bemutatása**” című fejezetben olvasható.

#### GET metódusok

Minden végpont tartalmaz legalább kettő GET metódust, melyek az adott egyed tulajdonságainak lekérésére szolgálnak. Ezen eljárások az adott rekord minden adatát visszaküldik a kliens számára. Amennyiben egy egyed kapcsolatban áll egy másik táblában szereplő egyeddel, akkor nem csak annak azonosítóját küldi vissza, hanem a kliens oldali alkalmazás számára további fontos információkat az adott rekordról. – Lásd.: 6.kép –

```
{
  "R_id": 7,
  "H_id": 14,
  "Mennyiség": 20,
  "Mertekegység": "g",
  "ReceptNev": "Sajtburgerleves",
  "KategoriaNev": "Leves",
  "FelhasznaloNev": "Anna",
  "HozzavalaloNev": "Vaj"
},
```

6. kép: A Recept\_Hozzávaló tábla GET metódusának válasza

A második GET metódus egy adott rekord lekérdezésére szolgál, azonban, ha két rekord között N:M kapcsolat áll fenn, akkor a válasz a rekordok összes elemét tartalmazza. – Például, ha egy hozzávaló összes mértékegységét szeretnénk lekérni, akkor elég megadnunk csak a hozzávaló azonosítóját a Mértékegység\_Hozzávaló tábla GET/Id metódusának. Lásd: 7. kép –

```
{
  "H_id": 20,
  "MertekegységNev": "g",
  "HozzavaloNev": "Sajt"
},
{
  "H_id": 20,
  "MertekegységNev": "szelet",
  "HozzavaloNev": "Sajt"
}
```

7. kép: A sajthoz tartozó összes mértékegység.

A GET metódusok válaszainak státuszkódjai:

- Ok – 200 – : Sikeres lekérdezés esetén, bármely GET metódusnál.
- NoContent – 204 – : Sikeres lekérdezés esetén, amennyiben a válasz üres. Csak a paraméter nélküli GET metódusnál.
- NotFound – 404 – : Sikertelen lekérdezés esetén, amennyiben a keresett rekord nem található. Csak a paraméteres GET metódusnál.

### **POST metódusok**

Minden végpont tartalmaz egy POST metódust, amely az adatok az adatbázisba történő rögzítésére szolgál. Minden POST eljárás egyedi JSON formátumú bemenettel rendelkezik, amely legalább egy mezőt tartalmaz. Azon végpontok esetében, ahol név alapján nem lehet kettő vagy több azonos rekord, ott a body minden esetben a rekordok név értékét fogja kérni. – Lásd: 8.kép –

```
{
  "HozzavaloNev": "string",
  "ErzékenységNev": "string"
}
```

8. kép: Hozzávaló\_Érzékenység tábla POST metódusának bemeneti értéke

A POST metódusok válaszainak státuszkódjai:

- Ok – 200 – : Az adat sikeres mentése esetén
- InternalServerError – 500 – : Az adatot valamilyen hiba miatt nem lehet elmenteni az adatbázisban

### **PUT metódusok**

A végpontok többsége implementálja a PUT metódust is, amely egy adott rekord minden adatát – az azonosító kivételével – módosít. Ezen eljárások esetében minden mezőt kötelező kitölteni. Minden PUT metódus egy egyedi JSON formátumú bemenettel rendelkezik, amely legalább egy mezőt tartalmaz. – Lásd: 9.kép –

```
{
  "Mennyiség": 0,
  "MertekegysegNev": "string"
}
```

9. kép: *Recept\_Hozzávaló* PUT metódusának bemeneti értéke

A PUT metódusok válaszainak státuszkódjai:

- Ok – 200 – : Az adatok módosítása sikeresen megtörtént.
- NotFound – 404 – : A módosítani kívánt rekord nem található.
- InternalServerError – 500 – : A rekord valamilyen hiba miatt nem módosítható. – ilyen hiba pl.: valamely bemeneti érték nincs kitöltve –
- NotImplemented – 501 – : A végpont nem támogatja a rekordok módosítását.

### **PATCH metódus**

A PATCH metódust mindösszesen egy végpont – a felhasználó – tartalmazza. Ezen eljárás egy adott rekord adott mezőinek módosítására szolgál. A PATCH metódus hívásakor a mezők kitöltése nem kötelező. Amennyiben csak egy adott tulajdonság értékének megváltoztatására van szükség, elegendő csak azon attribútum kitöltése. – Lásd: 10. és 11. kép –

```
{
  "Fnev": "string",
  "Email": "string",
  "Jelszo": "string",
  "Jogosultsag": 0,
  "Erzekeny": "string",
  "ProfilkepURL": "string"
}
```

10. kép : *Felhasználó* PATCH metódusának összes bementi értéke

```
{
  "Jelszo": "string"
}
```

11. kép : *Felhasználó* PATCH metódusának bemenete, amennyiben csak a jelszót módosítjuk



A PATCH metódus válaszainak státuszkódjai:

- Ok – 200 – : A rekord tulajdonságának sikeres módosítása.
- NotFound – 404 – : A módosítani kívánt rekord nem található.
- InternalServerError – 500 - : A módosítani kívánt rekord valamilyen hiba miatt nem módosítható.

### ***DELETE metódusok***

A DELETE metódust a legtöbb végpont – kettő kivételével : Hozzávaló\_Érzékenységek és a Tagek – tartalmazza. Ezen eljárás feladata, hogy töröljön egy meghatározott rekordot az adatbázisból. Egy törölt rekord visszaállítása nem lehetséges. Minden DELETE metódus legalább egy, de maximum kettő paramétert fogad.

A DELETE metódus válaszainak státuszkódjai:

- Ok – 200 - : A rekord sikeresen törölve lett az adatbázisból
- NotFound – 404 - : A törölni kívánt rekord nem található
- NotImplemented – 501 - A végpont nem támogatja a rekordok törlését.

### ***3.1.3c Metódusok részletes bemutatása***

#### ***Értékelés végpont***

Elérése a **https://localhost:44350/api/Ertekeles** URL-en keresztül történik. Ezen végpon-  
ton keresztül érhető el az összes leadott értékelés a különböző receptekre.

#### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy az értékelések minden adatát visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**”, című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- Ertid: egész szám : Az értékelés azonosítója
- Csillag: egész szám : Hány csillagra értékelt egy adott receptet
- ReceptId : egész szám : Az értékelt recept azonosítója
- ReceptNev : szöveg : Az értékelt recept neve
- FelhasznaloId : egész szám : A felhasználó azonosítója, aki értékelt a receptet
- FelhasznaloNev : szöveg : A felhasználó neve, aki értékelt a receptet

Ezen GET metódus az összes recept minden értékelését visszaküldi.

A *paraméteres GET* eljárás feladata, hogy az egy adott recepthez tartozó összes értékelést visszaküldje. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### **POST metódus**

A POST metódus feladata, hogy rögzítse az adatbázisban az adott receptekre leadott értékeléseket. Egy felhasználó ugyanarra a receptre kétszer nem adhat le értékelést, legfeljebb módosíthatja az előzőt.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- Csillag : egész szám – 1 és 5 között - : Az értékelés értéke
- ReceptId : egész szám : Azon recept azonosítója, amelyre az értékelést leadták
- FelhasznaloId : egész szám : A felhasználó azonosítója, aki értékelte a receptet

### **PUT metódus**

A PUT metódus feladata, hogy amennyiben egy felhasználó értékelt már egy adott receptet, akkor képes legyen azon változtatni. Ezen eljárás csak létező értékelések értékét tudja megváltoztatni, új értékelés felvételére nem alkalmas. **FONTOS**, hogy a metódus csak az értékelés *Csillag* értékét tudja megváltoztatni, függetlenül attól, hogy más értékeket is fogad.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet PUT metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő *paraméter bemenettel* – melyet, az elérési útban kell elhelyezni.

Lásd: 12.kép - rendelkezik:

- Id: egész szám : Az értékelés azonosítója

`https://localhost:44350/api/Ertekeles/13`

12. kép : Értékelés PUT metódusának paramétere

A JSON formátumú bemenet tulajdonságai a body-ban:

- Csillag : egész szám – 1 és 5 között - : Az értékelés értéke
- ReceptId : egész szám : Azon recept azonosítója, amelyre az értékelést leadták
- FelhasznaloId : egész szám : A felhasználó azonosítója, aki értékelte a receptet

### ***DELETE metódus***

A DELETE metódus felelős egy adott értékelés törléséért. Az eljárás csak létező értékelést képes törölni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus egy paraméter bemenettel rendelkezik:

- **Id**: egész szám : Az értékelés azonosítója

Az értékelés törlése után annak visszaállítására nincs lehetőség.

### ***Érzékenység végpont***

Elérése a **https://localhost:44350/api/Erzekenyseg** URL-en keresztül történik. Ezen végponton keresztül érhető el az összes érzékenység típus, amelyre a felhasználó esetleg érzékeny lehet.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy az érzékenységek minden adatát visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**”, című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- **Eid** : egész szám : Az érzékenység azonosítója
- **Nev** : szöveg : Az érzékenység neve

Ezen GET metódus az összes érzékenység tulajdonságait visszaküldi a kliens számára.

A *paraméteres GET* eljárás feladata, hogy egy adott érzékenység tulajdonságait küldje. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódus***

A POST eljárás feladata, hogy új, az adatbázisban még nem létező érzékenységet vegyen fel. Egy érzékenység csak egyszer szerepelhet az adatbázisban.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- **Eid** : egész szám : Az érzékenység azonosítója
- **Nev** : szöveg : Az érzékenység neve

### ***PUT metódus***

A PUT eljárás feladata, hogy egy, már meglévő érzékenység nevét módosítsa. Ezen metódus csak meglévő rekordok nevét tudja módosítani, új rekord felvételére nem alkalmas. **FONTOS**, hogy a metódus csak az értékelés *Nev* értékét tudja megváltoztatni, függetlenül attól, hogy más értéket is fogad.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet PUT metódusokról szóló fejezetében leírtakkal.

### ***DELETE metódus***

A DELETE metódus felelős egy adott érzékenység törléséért. Az eljárás csak létező érzékenységet képes törölni. Az érzékenység törlését követően minden olyan is törlésre kerül az adatbázisból, amellyel az kapcsolatban állt – töröl minden olyan rekordot a Felhasználó\_Érzékenység, illetve Hozzávaló\_Érzékenység táblákból is, amely kapcsolatban állt az adott érzékenységgel.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus egy paraméter bemenettel rendelkezik:

- **Id** : egész szám : Az érzékenység azonosítója

Az érzékenység törlése után annak visszaállítására nincs lehetőség.

### ***Felhasználó végpont***

Elérése a **https://localhost:44350/api/Felhasznalo** URL-en keresztül történik. Ezen végponton keresztül érhető el az összes felhasználó minden adata.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy a felhasználók minden adatát visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**”, című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- **Id** : egész szám : A felhasználó azonosítója
- **Fnev** : szöveg : A felhasználó neve – Felhasználóneve
- **Email** : szöveg : A felhasználó e-mail címe
- **Jelszo\_Hash** : byte típusú tömb : A felhasználó jelszava titkosítva
- **Jelszo\_Salt** : byte típusú tömb : A felhasználó jelszava titkosítva
- **Jogosultság** : egész szám : A felhasználó jogosultsága – Ez a szám 1, ha a felhasználó adminisztrátori joggal rendelkezik, minden más esetben ez a szám 2.
- **ProfilkepURL** : szöveg : A felhasználó profilképének elérési útja

Ezen GET metódus az összes felhasználó minden tulajdonságát visszaküldi a kliens számára.

A *paraméteres GET* eljárás feladata, hogy egy adott felhasználó tulajdonságait küldje. Ezen eljárás a következő paramétereket fogadja:

- Id : egész szám : A felhasználó azonosítója
- Email : szöveg : A felhasználó e-mail címe
- Jelszo : szöveg : A felhasználó jelszava

A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### **POST metódusok**

A Felhasználó végpont kettő POST eljárással rendelkezik. Az első POST metódus feladata, hogy új, az adatbázisban még nem létező felhasználót vegyen fel. Az újonnan felvett felhasználók automatikusan az alap felhasználói jogokkal rendelkeznek csak – Jogosultságuk 2-es értékkel rendelkezik – Egy felhasználó csak egyszer szerepelhet az adatbázisban, ugyanazzal az e-mail címmel és / vagy felhasználónévvel kétszer nem regisztrálhatnak.

A sikeres válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**” című fejezet POST metódusokról szóló fejezetében leírtakkal. Amennyiben a felhasználó valami oknál fogva nem rögzíthető az adatbázisban a következő válaszüzenetek lehetségesek:

- Conflict – 409 – : „Az E-mail cím már foglalt” : Abban az esetben, ha az adott E-mail címmel már regisztrált korábban felhasználó
- Conflict – 409 – : „A felhasználónév már foglalt” : Abban az esetben, ha az adott felhasználónévvel már regisztrált korábban felhasználó
- InternalServerError – 500 – : A felhasználót valamilyen hiba miatt nem lehet elmenteni az adatbázisban

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- Fnev : szöveg : A felhasználó neve – Felhasználóneve
- Email : szöveg : A felhasználó e-mail címe
- Jelszo : szöveg : A felhasználó jelszava
- Jogosultság : egész szám : A felhasználó jogosultsága – Ez a szám 1, ha a felhasználó adminisztrátori joggal rendelkezik, minden más esetben ez a szám 2
- Erzekeny : szöveg : Az érzékenység neve, amennyiben a felhasználó érzékeny valamilyen allergénre – Ezen mező kitöltése nem kötelező
- ProfilkepURL : szöveg : A felhasználó profilképének elérési útja

A végpont második POST metódusa az autentikációt – a felhasználó azonosítását – valósítja meg. Ezen eljárás a <https://localhost:44350/api/Felhasznalo/authenticate> URL-en keresztül érhető el.

Az eljárás egy JSON formátumú body-ban a következő értékeket fogadja:

- Email : szöveg : A felhasználó e-mail címe
- Jelszo : szöveg : A felhasználó jelszava

A válasz egy http státuszkódot, illetve sikeres azonosítás esetén egy objektumot tartalmaz.

A http státuszkódok a következőek lehetnek:

- Ok – 200 – : A felhasználó sikeresen azonosítva lett – megfelelő e-mail cím – jelszó párost adott meg
- Unauthorized – 401 – : A felhasználó azonosítása sikertelen – az e-mail cím – jelszó páros nem megfelelő
- NotFound – 404 – : A megadott e-mail címmel még nem regisztrált felhasználó

### ***PUT metódus***

A PUT eljárás feladata, hogy egy, már meglévő felhasználó adatait módosítsa. Ezen metódus csak meglévő rekordok adatait tudja módosítani, új rekord felvételére nem alkalmas. A metódus a felhasználók ezen adatainak módosítására alkalmas:

- Fnev : szöveg : A felhasználó felhasználóneve
- Email : szöveg : A felhasználó e-mail címe
- Jelszo : szöveg : A felhasználó jelszava
- Jogosultsag : egész szám : A felhasználó jogosultsága
- ProfilkepURL : szöveg : A felhasználó profilképének elérési útja

Az eljárás JSON formátumú bemenete a kérés body-ban, megegyezik a fent leírtakkal.

A metódus paraméterként egy értéket fogad:

- Id : egész szám : A felhasználó azonosítója

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet PUT metódusokról szóló fejezetében leírtakkal.

### ***PATCH metódus***

A PATCH metódus feladata, hogy a felhasználók egyes adatát módosítsa. A PUT eljárástól eltérően, ezen metódushívásakor a JSON formátumú bemenet nem minden adata kötelezően kitöltendő. A módosítható tulajdonságok megegyeznek a PUT metódus által módosítható értékekkel.

### ***DELETE metódus***

A DELETE metódus felelős egy adott felhasználó törléséért. Az eljárás csak létező felhasználókat képes törölni. A felhasználói fiók törlését követően minden olyan is törlésre kerül az adatbázisból, amellyel az kapcsolatban állt – töröl minden olyan rekordot a

- Felhasználó\_Érzékenység
- Recept, Értékelés
- Lépés
- Multimédia
- Recept\_Hozzávaló
- Recept\_Tag

táblákból is, amely kapcsolatban állt az adott felhasználóval. –

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „ **3.1.3.b Metódusok általános működése** „ című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus egy paraméter bemenettel rendelkezik:

- Id: egész szám : A felhasználó azonosítója

Egy felhasználói fiók törlése után annak visszaállítására nincs lehetőség.

### ***Felhasználó\_Érzékenység végpont***

Elérése a **[https://localhost:44350/api/Felhasznalo\\_Erzekenysseg](https://localhost:44350/api/Felhasznalo_Erzekenysseg)** URL-en keresztül történik. Ezen végponton keresztül érhető el, hogy az egyes felhasználók mely allergénekre érzékenyek.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy az összes felhasználó minden érzékenységét visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „ **3.1.3.b Metódusok általános működése** „ című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- F\_id : egész szám : A felhasználó azonosítója
- E\_id : egész szám : Az érzékenység azonosítója
- FelhasznaloNev : szöveg : A felhasználó felhasználóneve
- ErzenysegNev : szöveg : Az érzékenység megnevezése

A *paraméteres GET* eljárás feladata, hogy egy adott felhasználó összes érzékenységét küldje. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódus***

A POST eljárás feladata, hogy az adatbázisban rögzítse az egyes felhasználók érzékenységeit. Egy felhasználóhoz ugyanaz az érzékenység többször nem tartozhat.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- FelhasznaloId : egész szám : Az felhasználó azonosítója
- ErzekenysegNev : szöveg : Az érzékenység neve

### ***PUT metódus***

Ezen végpont esetében a PUT eljárás **nincs implementálva**. Hívása a következő http státuszkódot eredményezi:

- NotImplemented – 501 – : A végpont nem tartalmazza a PUT metódust.

### ***DELETE metódus***

A DELETE metódus felelős egy adott érzékenység törléséért adott felhasználó esetében. Az eljárás csak létező rekordot képes törölni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus kettő paraméter bemenettel rendelkezik:

- fid: egész szám : A felhasználó azonosítója
- ErzNev : szöveg : Az érzékenység neve

Az érzékenység törlése után annak visszaállítására nincs lehetőség.

### ***Hozzávaló végpont***

Elérése a **https://localhost:44350/api/Hozzavalo** URL-en keresztül történik. Ezen végpon-  
ton keresztül érhető el, az adatbázisban tárolt hozzávalók minden adata.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy az összes hozzávaló minden tulajdonságát visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**”, című fejezet GET metódusokról szóló fejezetében leírtakkal.



A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- Hid : egész szám : A hozzávaló azonosítója
- Nev : szöveg : A hozzávaló megnevezése
- ReceptNevek : tömb : Az összes recept nevét tartalmazó tömb, amelyben szerepel az adott hozzávaló
- ErzékenységNev : szöveg : Az érzékenység megnevezése

A *paraméteres GET* eljárás feladata, hogy egy adott hozzávaló összes tulajdonságát küldje. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódus***

A POST eljárás feladata, hogy az adatbázisban új, még nem létező hozzávalót rögzítsen. Egy hozzávaló csak egyszer szerepelhet az adatbázisban.

A sikeres válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**„ című fejezet POST metódusokról szóló fejezetében leírtakkal. Amennyiben a hozzávaló valami oknál fogva nem rögzíthető az adatbázisban a következő válaszüzenetek lehetségesek:

- Conflict – 409 – : „A hozzávaló már létezik” : Abban az esetben, ha a hozzávaló már szerepel az adatbázisban
- InternalServerError – 500 – : Abban az esetben, ha a hozzávaló valamilyen oknál fogva nem rögzíthető az adatbázisban

### ***PUT metódus***

Ezen végpont esetében a PUT eljárás **nincs implementálva**. Hívása a következő http státuszkódot eredményezi:

- NotImplemented – 501 – : A végpont nem tartalmazza a PUT metódust.

### ***DELETE metódus***

A DELETE metódus felelős egy adott hozzávaló törléséért. Az eljárás csak létező rekordot képes törölni. Egy hozzávaló törlését követően minden rekord törlésre kerül az adatbázisból, amellyel az kapcsolatban állt – töröl minden olyan rekordot a

- Mértékegység\_Hozzávaló
- Hozzávaló\_Érzékenység
- Recept\_Hozzávaló

táblákból is, amelyel az adott hozzávaló kapcsolatban állt.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „ **3.1.3.b Metódusok általános működése** „ című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus egy paraméter bemenettel rendelkezik:

- Id : egész szám : A hozzávaló azonosítója

Az hozzávaló törlése után annak visszaállítására nincs lehetőség.

### ***Hozzávaló\_Érzékenység végpont***

Elérése a **https://localhost:44350/api/Hozzavalo\_Erzenyseg** URL-en keresztül történik.

Ezen végponton keresztül érhető el, hogy a különböző hozzávalók allergén osztályokba tartoznak.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy visszaküldje minden hozzávaló allergén osztályait a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „ **3.1.3.b Metódusok általános működése** „ című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- Eid : egész szám : Az allergén azonosítója
- Híd : egész szám : A hozzávaló azonosítója
- HozzavaloNev : szöveg : A hozzávaló megnevezése
- ErzenysegNev : szöveg : Az allergén megnevezése

Ezen GET metódus minden hozzávaló-érékenység párost visszaküld.

A *paraméteres GET* eljárás feladata, hogy az egy adott allergén osztályhoz tartozó összes hozzávalót visszaküldje. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódus***

A POST metódus feladata, hogy rögzítsen az adatbázisban egy adott hozzávalóhoz tartozó allergén osztályt.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- HozzvaloNev : szöveg : A hozzávaló megnevezése
- ErzekenysegNev : szöveg : A hozzávaló megnevezése

### ***PUT metódus***

Ezen végpont esetében a PUT eljárás **nincs implementálva**. Hívása a következő http státuszkódot eredményezi:

- NotImplemented – 501 – : A végpont nem tartalmazza a PUT metódust.

### ***DELETE metódus***

Ezen végpont esetében a DELETE eljárás **nincs implementálva**. Hívása a következő http státuszkódot eredményezi:

- NotImplemented – 501 – : A végpont nem tartalmazza a PUT metódust.

### ***Ízlés végpont***

Elérése a **https://localhost:44350/api/Izles** URL-en keresztül történik. Ezen végponton keresztül érhető el, hogy a felhasználók milyen ízlésvilággal rendelkeznek.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy minden felhasználó „ízlését” visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**”, című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- T\_id : egész szám : A választott tag azonosítója
- F\_id : egész szám : A felhasználó azonosítója

Ezen GET metódus minden felhasználó ízlését visszaküldi a kliens számára.

A *paraméteres GET* eljárás feladata, hogy egy adott felhasználó ízlését küldje. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódus***

A POST eljárás feladata, hogy rögzítse az adatbázisban a regisztrált felhasználó által választott tage(ke)t a felhasználóhoz párosítva az(oka)t.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „ **3.1.3.b Metódusok általános működése** „ című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- TagNev : szöveg : A tag neve

A metódus ezen kívül paraméterként még a következőt várja:

- F\_id : egész szám : A felhasználó azonosítója

### ***PUT metódus***

Ezen végpont esetében a PUT eljárás **nincs implementálva**. Hívása a következő http státuszkódot eredményezi:

- NotImplemented – 501 – : A végpont nem tartalmazza a PUT metódust.

### ***DELETE metódus***

A DELETE metódus felelős egy adott felhasználó-tag rekord törléséért. Az eljárás csak létező rekordot képes törölni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „ **3.1.3.b Metódusok általános működése** „ című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus kettő paraméter bemenettel rendelkezik:

- T\_id: egész szám : A tag azonosítója
- F\_id : egész szám : A felhasználó azonosítója

Egy rekord törlése után annak visszaállítására nincs lehetőség.

### ***Kategória végpont***

Elérése a **https://localhost:44350/api/Kategória** URL-en keresztül történik. Ezen végpon-  
ton keresztül érhető el minden étel kategória.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy minden kategóriát visszaküldjön a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartal-  
maz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános mű-  
ködése** „ című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- Kid: egész szám : A kategória azonosítója
- Nev : szöveg : A kategória megnevezése

Ezen GET metódus minden kategóriát visszaküld a kliens számára.

A *paraméteres GET* eljárás feladata, hogy egy adott kategóriát küldjön annak azonosítója alapján. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódus***

A POST eljárás feladata, hogy rögzítse az adatbázisban a kategóriákat. Minden kategória csak egyszer szerepelhet az adatbázisban.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése** „ című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- Nev : szöveg : A kategória neve

### ***PUT metódus***

A PUT metódus feladata, hogy egy adott kategória nevét meglehessen megváltoztatni. Ezen eljárás csak létező kategória *Nev* tulajdonságát tudja megváltoztatni, új kategória felvételére nem alkalmas.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése** „ című fejezet PUT metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő paraméter bemenettel rendelkezik:

- id : egész szám : A kategória azonosítója

A JSON formátumú bemenet tulajdonsága a body-ban:

- Nev : szöveg : A kategória megnevezése

### ***DELETE metódus***

A DELETE metódus felelős egy adott kategória törléséért. Az eljárás csak létező kategóriát képes törölni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „ **3.1.3.b Metódusok általános működése** „ című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus egy paraméter bemenettel rendelkezik:

- id : egész szám : A kategória azonosítója

Egy rekord törlése után annak visszaállítására nincs lehetőség.

### ***Lépés végpont***

Elérése a **https://localhost:44350/api/Lepes** URL-en keresztül történik. Ezen végponton keresztül érhető el minden recept elkészítésének lépései

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy az összes recept minden lépését visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „ **3.1.3.b Metódusok általános működése** „ című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- Lid: egész szám : A lépés azonosítója
- Sorszam : egész szám : A lépés sorszáma egy adott recepten belül
- Leiras : szöveg : A lépés leírása
- ReceptId : egész szám : Azon recept azonosítója, amelyhez tartozik az adott lépés
- ReceptNev : szöveg : Azon recept megnevezése, amelyhez tartozik az adott lépés

Ezen GET metódus minden lépést visszaküld a kliens számára.

A *paraméteres GET* eljárás feladata, hogy egy adott recepthez tartozó minden lépést visszaküldjön a recept azonosítója alapján. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódus***

A POST eljárás feladata, hogy rögzítse az adatbázisban a receptekhez tartozó minden lépést.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „ **3.1.3.b Metódusok általános működése** „ című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- Sorszam : egész szám : A lépés sorszáma az adott recepten belül
- Leiras : szöveg : A lépés leírása
- R\_id : egész szám : A recept azonosítója, amelyhez a recept tartozik

### ***PUT metódus***

A PUT metódus feladata, hogy egy adott recepthoz tartozó lépések leírásait meg lehessen változtatni. Ezen eljárás csak létező lépés *Leiras* és *Sorszam* tulajdonságait tudja megváltoztatni, új lépés felvételére nem alkalmas.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**” című fejezet PUT metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő paraméter bemenettel rendelkezik:

- *id* : egész szám : A recept azonosítója, amelyhez tartozik az adott lépés

A JSON formátumú bemenet tulajdonsága a *body*-ban:

- *Sorszam* : egész szám : A lépés sorszáma az adott recepten belül
- *Leiras* : szöveg : A lépés leírása

### ***DELETE metódus***

A DELETE metódus felelős egy adott lépés törléséért. Az eljárás csak létező lépést képes törölni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**” című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus egy paraméter bemenettel rendelkezik:

- *id* : egész szám : A lépés azonosítója

Egy rekord törlése után annak visszaállítására nincs lehetőség.

### ***Mértékegység végpont***

Elérése a **https://localhost:44350/api/Mertekegység** URL-en keresztül történik. Ezen végponton keresztül érhető el minden mértékegység, melyek hozzárendelhetők a különböző hozzávalókhoz.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy az összes mértékegységet visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú *body*-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**” című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- *Mid* : egész szám : A mértékegység azonosítója
- *MertekegységNev* : szöveg : A mértékegység megnevezése

Ezen GET metódus minden mértékegységet visszaküld a kliens számára.

A *paraméteres GET* eljárás feladata, hogy egy adott mértékegységet küldjön annak azonosítója alapján. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* módszer válaszával.

### ***POST módszer***

A POST eljárás feladata, hogy rögzítsen az adatbázisban egy adott mértékegységet. Minden mértékegység csak egyszer szerepelhet az adatbázisban.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- *MertekegysegNev* : szöveg : A mértékegység megnevezése

### ***PUT módszer***

A PUT módszer feladata, hogy egy adott mértékegység megnevezését meg lehessen változtatni. Ezen eljárás csak létező mértékegység *MertekegysegNev* tulajdonságát tudja megváltoztatni, új mértékegység felvételére nem alkalmas.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet PUT metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő paraméter bemenettel rendelkezik:

- *id* : egész szám : A mértékegység azonosítója

A JSON formátumú bemenet tulajdonsága a body-ban:

- *MertekegysegNev* : szöveg : A mértékegység megnevezése

### ***DELETE módszer***

A DELETE módszer felelős egy adott mértékegység törléséért. Az eljárás csak létező mértékegységet képes törölni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen módszer egy paraméter bemenettel rendelkezik:

- *id* : egész szám : A mértékegység azonosítója

Egy rekord törlése után annak visszaállítására nincs lehetőség.



### ***Mértékegység\_Hozzávaló végpont***

Elérése a **https://localhost:44350/api/Mertekegység\_Hozzavalo** URL-en keresztül történik. Ezen végponton keresztül érhető el, hogy mely hozzávalók milyen mértékegységben adhatóak meg.

#### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy az összes hozzávalóhoz tartozó minden mértékegységet visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**„ című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- **H\_id** : egész szám : A hozzávaló azonosítója
- **MertekegységNev** : szöveg : A mértékegység megnevezése
- **HozzavaloNev** : szöveg : A hozzávaló megnevezése

Ezen GET metódus az összes hozzávaló minden mértékegységét visszaküldi.

A *paraméteres GET* eljárás feladata, hogy az egy adott hozzávalóhoz tartozó összes mértékegységet visszaküldje. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

#### ***POST metódus***

A POST metódus feladata, hogy rögzítse az adatbázisban egy adott hozzávalóhoz tartozó minden mértékegységet. Ezen metódus egy időben több mértékegységet is képes hozzávalóhoz rendelni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**„ című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- **HozzavaloNev** : szöveg : A hozzávaló megnevezése
- **MertekegységNev** : szöveg : A mértékegységek megnevezése – Ezen mezőben vesszővel elválasztva több mértékegység is megadható. Ebben az esetben minden mértékegység hozzálesz rendelve a megadott hozzávalóhoz. Lásd : 13.kép –

```
{  
  "HozzavaloNev": "Paprika",  
  "MertekegységNev": "kg,dkg,g,db"  
}
```

13. kép : Mértékegység\_Hozzávaló POST metódusának JSON bemenetének lehetséges megadási módja

### ***PUT metódus***

Ezen végpont esetében a PUT eljárás **nincs implementálva**. Hívása a következő http státuskódot eredményezi:

- NotImplemented – 501 – : A végpont nem tartalmazza a PUT metódust.

### ***DELETE metódus***

A DELETE metódus felelős egy adott hozzávalóhoz rendelt mértékegység törléséért. Az eljárás csak létező rekordot képes törölni.

A válaszüzenet egy http státuskódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus kettő paraméter bemenettel rendelkezik:

- Hid : egész szám : A hozzávaló azonosítója
- Mid : egész szám : A mértékegység azonosítója

A rekord törlése után annak visszaállítására nincs lehetőség.

### ***Multimédia végpont***

Elérése a **https://localhost:44350/api/Multimedia** URL-en keresztül történik. Ezen végponton keresztül érhető el minden kép file, amely a receptekhez tartoznak.

### ***GET metódusok***

A *paraméter nélküli* GET eljárás feladata, hogy az összes kép file minden tulajdonságát visszaküldje a kliens számára. A válaszüzenet egy http státuskódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuskódjai megegyeznek a „**3.1.3.b Metódusok általános működése**”, című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- Vid : egész szám : A kép azonosítója
- URL : szöveg : A file egyedi neve
- ReceptId : egész szám : A recept azonosítója, amelyhez a kép tartozik
- ReceptNev : szöveg : A recept megnevezése, amelyhez a kép tartozik

Ezen GET metódus minden kép file-t visszaküld a kliens számára.

A *paraméteres* GET eljárás feladata, hogy egy adott recepthez tartozó képet küldjön a recept azonosítója alapján. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli* GET metódus válaszával.

### ***POST metódus***

A POST eljárás feladata, hogy rögzítse az adatbázisban egy adott recepthez tartozó kép file-t.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- URL : szöveg : A kép egyedi neve
- R\_id : egész szám : A recept azonosítója, amelyhez a kép tartozik

### ***PUT metódus***

Ezen végpont esetében a PUT eljárás **nincs implementálva**. Hívása a következő http státuszkódot eredményezi:

- NotImplemented – 501 – : A végpont nem tartalmazza a PUT metódust.

### ***DELETE metódus***

A DELETE metódus felelős egy adott kép törléséért. Az eljárás csak létező képt képes törölni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus egy paraméter bemenettel rendelkezik:

- id : egész szám : A kép azonosítója

Egy rekord törlése után annak visszaállítására nincs lehetőség.

### ***Recept végpont***

Elérése a **https://localhost:44350/api/Recept** URL-en keresztül történik. Ezen végponton keresztül érhető el minden recept.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy az összes recept minden tulajdonságát visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**” című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- **Rid**: egész szám : A recept azonosítója
- **Nev** : szöveg : A recept megnevezése
- **Leiras** : szöveg : A recept leírása
- **KategoriaNev** : szöveg : A recept kategóriája
- **FelhasznaloNev** : szöveg : A felhasználó felhasználóneve, aki feltöltötte az adott receptet
- **Allapot** : egész szám : Az adott receptet elfogadásra került-e egy admin által vagy sem – Egy recept állapota 1, amennyiben a recept elfogadásra került. Amennyiben az állapot 0, a recept még felülvizsgálatra vár.
- **Eperc** : egész szám : A recept elkészítésének ideje percben mérve
- **Szakmai** : logikai : A recept egy séfként dolgozó személy által lett-e összeállítva
- **Nehezseg** : szöveg : A recept elkészítésének nehézsége
- **HozzavaloDb** : egész szám : Számított érték, a recept mennyi különböző hozzávalót tartalmaz.

Ezen GET metódus minden receptet visszaküld a kliens számára.

A *paraméteres GET* eljárás feladata, hogy egy adott receptet küldjön annak azonosítója alapján. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódusok***

A recept végpont kettő POST eljárással rendelkezik. Az első metódus feladata, hogy rögzítsen az adatbázisban egy adott receptet.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**” című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- Nev : szöveg : A recept megnevezése
- Leiras : szöveg : A recept leírása
- KategoriaNev : szöveg : A recept kategóriája
- FelhasznaloNev : szöveg : A felhasználó felhasználóneve, aki feltöltötte az adott receptet
- Eperc : egész szám : A recept elkészítésének ideje percben mérve
- Szakmai : logikai : A recept egy séfként dolgozó személy által lett-e összeállítva
- Nehezseg : szöveg : A recept elkészítésének nehézsége

A végpont második POST metódusa a mértékegység átváltást valósítja meg. Ezen eljárás a <https://localhost:44350/atvaltas> URL-en keresztül érhető el.

A metódus egy JSON formátumú body-ban a következő értékeket fogadja:

- mibol : szöveg : Az eredeti mértékegység neve
- mibe : szöveg : A mértékegység, amibe átváltunk
- mennyiseg : egész szám : Az átváltandó mennyiség

A válasz egy http státuszkódot, illetve sikeres átváltás esetén egy decimális számot tartalmaz. A http státuszkódok a következők lehetnek:

- Ok – 200 – : Az átváltás sikeresen megtörtén
- NotFound – 404 – : A megadott mértékegység nem található

### ***PUT metódus***

A PUT metódus feladata, hogy egy adott recept tulajdonságait megváltoztassa. Ezen eljárás csak létező recept tulajdonságait tudja megváltoztatni, új recept felvételére nem alkalmas. A metódus a következő mezők értékének megváltoztatására alkalmas:

- Kategória
- Név
- Szakmai
- Eperc
- Állapot
- Nehézség

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**”, című fejezet PUT metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő paraméter bemenettel rendelkezik:

- id : egész szám : A recept azonosítója

A JSON formátumú bemenet tulajdonsága a body-ban:

- Rid: egész szám : A recept azonosítója
- Nev : szöveg : A recept megnevezése
- Leiras : szöveg : A recept leírása
- Kategorianev : szöveg : A recept kategóriája
- FelhasznaloNev : szöveg : A felhasználó felhasználóneve, aki feltöltötte az adott receptet
- Allapot : egész szám : Az adott receptet elfogadásra került-e egy admin által vagy sem – Egy recept állapota 1, amennyiben a recept elfogadásra került. Amennyiben az állapot 0, a recept még felülvizsgálatra vár.
- Eperc : egész szám : A recept elkészítésének ideje percben mérve
- Szakmai : logikai : A recept egy séfként dolgozó személy által lett-e összeállítva
- Nehezseg : szöveg : A recept elkészítésének nehézsége

### ***DELETE metódus***

A DELETE metódus felelős egy adott recept törléséért. Az eljárás csak létező receptet képes törölni. Egy recept törlése töröl minden olyan rekordot a következő táblákból:

- Értékelések
- Lépések
- Multimédiák
- Recept\_Hozzávalók
- Recept\_Tagek

amellyel az adott recept kapcsolatban állt.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**” című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus egy paraméter bemenettel rendelkezik:

- id : egész szám : A recept azonosítója

Egy rekord törlése után annak visszaállítására nincs lehetőség.

### ***Recept\_Hozzávaló végpont***

Elérése a **https://localhost:44350/api/Recept\_Hozzavalo** URL-en keresztül történik. Ezen végponton keresztül érhető el az összes recept minden hozzávalója.

#### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy az összes recept minden hozzávalóját visszaküldje a kliens számára. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**” című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- **R\_id**: egész szám : A recept azonosítója
- **H\_id** : egész szám : A hozzávaló azonosítója
- **Mennyiség** : egész szám : Az adott hozzávalóból mennyire van szükség
- **Mertekegyseg** : szöveg : A hozzávaló milyen mértékegységben van meghatározva
- **ReceptNev** : szöveg : A recept megnevezése, amelyhez a hozzávaló tartozik
- **KategoriaNev** : szöveg : A recept milyen kategóriába tartozik
- **FelhasznaloNev** : szöveg : A felhasználó neve, aki feltöltötte a receptet
- **HozzavaloNev** : szöveg : A hozzávaló megnevezése

Ezen GET metódus az összes recept minden hozzávalóját visszaküldi.

A *paraméteres GET* eljárás feladata, hogy egy adott recepthez tartozó összes hozzávalót visszaküldje. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

#### ***POST metódus***

A POST metódus feladata, hogy rögzítse az adatbázisban az adott recepthez tartozó hozzávalókat.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**” című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- **Mennyiség** : egész szám : Az adott hozzávalóból mennyire van szükség
- **MertekegysegNeve** : szöveg : A hozzávaló milyen mértékegységben van meghatározva
- **HozzavaloNev** : szöveg : A hozzávaló megnevezése
- **R\_id** : egész szám : A recept azonosítója

### ***PUT metódus***

A PUT metódus feladata, hogy egy recept adott hozzávalójának *menyiség*, illetve *mértékegység* tulajdonságait megváltoztassa. Ezen eljárás csak létező rekordok értékét tudja megváltoztatni, új rekord felvételére nem alkalmas.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**” című fejezet PUT metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő kettő *paraméter bemenettel* rendelkezik:

- Rid : egész szám : A recept azonosítója
- HozzavaloNev : szöveg : A hozzávaló megnevezése

A JSON formátumú bemenet tulajdonságai a body-ban:

- Mennyiség : egész szám : Az adott hozzávalóból mennyire van szükség
- MertekegysegNeve : szöveg : A hozzávaló milyen mértékegységben van meghatározva

### ***DELETE metódus***

A DELETE metódus felelős egy meghatározott recept adott hozzávalójának törléséért. Az eljárás csak létező rekordot képes törölni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**” című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus kettő paraméter bemenettel rendelkezik:

- Rid : egész szám : A recept azonosítója
- HozzavaloNev : szöveg : A hozzávaló megnevezése

A hozzávaló törlését követően annak visszaállítására nincs lehetőség.

### ***Recept\_Tag végpont***

Elérése a **https://localhost:44350/api/Recept\_Tag** URL-en keresztül történik. Ezen végponton keresztül érhető el az összes recepthez tartozó minden tag.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy visszaküldje a kliens számára az összes recepthez tartozó minden taget. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**” című fejezet GET metódusokról szóló fejezetében leírtakkal.



A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- R\_id: egész szám : A recept azonosítója
- T\_id : egész szám : A tag azonosítója
- ReceptNev : szöveg : A recept megnevezése, amelyhez a tag tartozik
- TagNev : szöveg : A tag megnevezése

Ezen GET metódus az összes recept minden tag-jét visszaküldi.

A *paraméteres GET* eljárás feladata, hogy egy adott recepthez tartozó összes taget visszaküldje. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódus***

A POST metódus feladata, hogy rögzítse az adatbázisban az adott recepthez tartozó tageket.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „ **3.1.3.b Metódusok általános működése** „ című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- R\_id : egész szám : A recept azonosítója
- TagNev : szöveg : A tag megnevezése

### ***PUT metódus***

A PUT metódus feladata, hogy egy recept adott tag-jét megváltoztassa. Ezen eljárás csak létező rekordok értékét tudja megváltoztatni, új rekord felvételére nem alkalmas.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „ **3.1.3.b Metódusok általános működése** „ című fejezet PUT metódusokról szóló fejezetében leírtakkal.

Ezen eljárás egy *paraméter bemenettel* rendelkezik:

- id : egész szám : A recept azonosítója
- HozzavaloNev : szöveg : A hozzávaló megnevezése

A JSON formátumú bemenet tulajdonságai a body-ban:

- TagNev : szöveg: Az adott tag megnevezése, amelyet cserélni szeretnénk
- Uj\_TagNev : szöveg : A tag megnevezése, amelyre cserélni szeretnénk

### ***DELETE metódus***

A DELETE metódus felelős egy meghatározott recept adott tag-jének törléséért. Az eljárás csak létező rekordot képes törölni.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „ **3.1.3.b Metódusok általános működése** „ című fejezet DELETE metódusokról szóló fejezetében leírtakkal.

Ezen metódus kettő paraméter bemenettel rendelkezik:

- id : egész szám : A tag azonosítója
- Rid : egész szám : A recept azonosítója

A tag törlését követően annak visszaállítására nincs lehetőség.

### ***Tag végpont***

Elérése a **https://localhost:44350/api/Recept\_Tag** URL-en keresztül történik. Ezen végponton keresztül érhető el az összes recepthez tartozó minden tag.

### ***GET metódusok***

A *paraméter nélküli GET* eljárás feladata, hogy visszaküldje a kliens számára az adatbázisban szereplő összes tag-et. A válaszüzenet egy http státuszkódot, illetve egy JSON formátumú body-t tartalmaz. A válasz lehetséges státuszkódjai megegyeznek a „**3.1.3.b Metódusok általános működése**„ című fejezet GET metódusokról szóló fejezetében leírtakkal.

A szerver válaszában küldött JSON objektum a következő tulajdonság értékeket tartalmazza:

- Tid : egész szám : A tag azonosítója
- Nev : szöveg : A tag megnevezése

Ezen GET metódus az összes tag-et visszaküldi.

A *paraméteres GET* eljárás feladata, hogy egy adott tag tulajdonságait küldje vissza. A válasz felépítése teljes mértékben megegyezik a *paraméter nélküli GET* metódus válaszával.

### ***POST metódus***

A POST metódus feladata, hogy rögzítsen az adatbázisban egy új, az adatbázisban még nem létező tag-et.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**„ című fejezet POST metódusokról szóló fejezetében leírtakkal.

Ezen eljárás a következő JSON formátumú bemenetet vár a kérés body-ban:

- Nev : szöveg : A tag megnevezése

### ***PUT metódus***

A PUT metódus feladata, hogy egy tag *Nev* tulajdonságát megváltoztassa. Ezen eljárás csak létező rekordok értékét tudja megváltoztatni, új rekord felvételére nem alkalmas.

A válaszüzenet egy http státuszkódot tartalmaz, amely megegyezik a „**3.1.3.b Metódusok általános működése**„ című fejezet PUT metódusokról szóló fejezetében leírtakkal.

Ezen eljárás egy *paraméter bemenettel* rendelkezik:

- id : egész szám : A tag azonosítója

A JSON formátumú bemenet tulajdonságai a body-ban:

- Nev : szöveg: A tag megnevezése

### ***DELETE metódus***

Ezen végpont esetében a DELETE eljárás **nincs implementálva**. Hívása a következő http státuszkódot eredményezi:

- NotImplemented – 501 – : A végpont nem tartalmazza a DELETE metódust.

## **3.2 A frontend komponens működése**

### **3.2.1 Fő funkciója**

A projektünk frontend része React komponensekből épül fel, illetve Vite keretrendszert használunk a megvalósításához. Az alkalmazásunkat stuktúrált felépítés jellemzi ez lehetővé teszi a könnyű navigációt és a funkcionalitást. A komponensek közötti kommunikáció React hook-okon – mint a useState, useEffect, useNavigate - történik és a react-router-dom könyvtáron alapul, míg az adatok tárolása jelentős részben az adatbázisban részben a böngésző localStorage-ában történik – itt tároljuk el az éppen bejelentkezett felhasználó egyes adatait. A frontend oldal vizuális megjelenését egyedi CSS fájlok alakítják, ezek minden komponenshez külön van definiálva. Célunk az volt, hogy a lehető legletisztultabb, legfelhasználóbarátabbá tegyük az oldalunkat.

A backend API-val való kommunikáció a Fetch API-n keresztül valósul meg, amely lehetővé teszi az adatok dinamikus lekérdezését és módosítását.

### **3.2.2 Komponensek részletes leírása és rövid bemutatása**

#### **3.2.2a Kezdőképernyő szekció**

A LoadingScreen komponens egy vizuális betöltési képernyő.

#### **3.2.2b Kezdőoldal**

A projektünk kezdőoldala a Home komponens, egy rövid betöltő képernyő után ez fogadja a felhasználókat. Az oldal ezen részén megjelennek más komponensek is, amikre a továbbiakban bővebben is ki fogok térni - ilyen például a Header, MainBanner, BlogSection, Footer. Ez a komponensünk egyszerű, de hatékony struktúrája biztosítja, hogy a felhasználók egy jól felépített, átlátható oldalt kapjanak.

A Home komponens felépítése lehetővé teszi, hogy a fejlesztők könnyedén testre szabják, vagy új szekciókkal bővítsék. Jelenleg nincs benne állapotkezelés vagy API-hívás így a működése teljesen a benne használt komponensektől függ. Ez a letisztult megközelítés jól támogatja a modularitást.

#### **3.2.2c Fejléc**

A Header komponens a projektünk navigációs sávjaként szolgál, így ez minden oldalunkon megjelenik, ezzel biztosítjuk az egységességet az oldalaink között.

Ez a komponens kulcsfontosságú a felhasználói élmény szempontjából, mivel lehetővé teszi a gyors navigációt a kezdőlap, a receptfeltöltés, a hűtőalapú keresés, az összes recept megtekintése, az adminisztrátori felület, illetve a felhasználói profil között. A komponens dina-

mikusan kezeli a felhasználói autentikációt és a jogosultságokat, így például az adminisztrátori menüpont csak azok számára jelenik meg, akik megfelelő jogosultsággal rendelkeznek.

A Header elemei közé tartozik - valami, ami a kezdőlapra navigál -, egy hamburger menü mobil nézetekhez, valamint egy profilkép, amely a felhasználói menüt aktiválja. A profilmenü dinamikusan változik attól függően, hogy a felhasználó be van-e jelentkezve. Bejelentkezett állapotban a felhasználói oldal megtekintését, illetve a kijelentkezést kínálja, míg kijelentkezett állapotban a bejelentkezési és regisztrációs opciókat jeleníti meg. Abban az esetben, ha nincs bejelentkezve felhasználó vagy pedig a bejelentkezett felhasználó nem állított be magának profilképet akkor egy alapértelmezett kép kerül elhelyezésre.

### **3.2.2d Lábléc**

A Footer komponens az alkalmazás lábléce, amely statikus tartalmat jelenít meg minden oldalon. Ez a komponens minimalista megközelítést követ, kizárólag alapvető információkat közöl. A lábléc egyszerűségével szeretnénk volna biztosítani, hogy ne vonja el a figyelmet a fő tartalomtól, de közben tartsa fent a professzionális megjelenést.

Ehhez az oldalhoz tartozó stílus megjelenítéseket a Footer.css fájl definiálja.

A komponens jelenlegi állapotában megfelel az oldalunk számára, viszont a jövőben érdemes lehet további elemekkel is bővíteni, mint például közösségi média linkekkel vagy hírlevél feliratkozási lehetőséggel.

### **3.2.2e Receptajánló szekció**

A BlogSection komponens a kezdőlap receptajánló szekciója, amely a felhasználó ízlésprofiljához igazított recepteket jelenít meg. Ez a komponens különösen fontos a személyre szabott élmény biztosításában, mivel az API-tól lekért adatok alapján szűri a recepteket, figyelembe véve a felhasználó által preferált „tageket”.

A komponens működése több lépésben zajlik.

1. Először a useEffect hook segítségével lekéri a felhasználó ízlésprofilját - **/api/Ízles/:userId** - és az összes receptet - **/api/Recept**.
2. Minden recepthez külön lekéri a hozzá tartozó tageket - **/api/Recept\_Tag/:recipeId** - és az adott recepthez tartozó képet - **/api/Multimedia/:recipeId** -,
3. majd az adatok alapján szűri a recepteket.

Ha a felhasználó ízlésprofilja alapján nem található elegendő releváns recept, a komponens véletlenszerű receptekkel egészíti ki a listát, hogy mindig legalább három receptet jelenítsen

meg. A recepteket egy rácsos elrendezésben mutatja, ahol minden elem kattintható, és a `useNavigate` hook segítségével a `/recipeDetails/:recipeId` útvonalra irányít.

A komponens állapotkezelése a `useState` hook-okkal történik, amelyek a felhasználói tageket – `userTags` –, az összes receptet – `recipes` – és a szűrt recepteket – `filteredRecipes` – tárolják.

### **3.2.2f Regisztráció szekció**

A Registration komponens a felhasználói regisztrációt kezeli, lehetővé téve új fiókok létrehozását a weboldalon.

A komponens négy fő állapotot kezel, a felhasználónevet, e-mail címet, jelszót és a jelszó megerősítését. Ezeken kívüli állapotok nyomon követik a regisztrációs kritériumokat, például a felhasználónév hosszát, az e-mail formátumát és a jelszó erősségét. A jelszó erősségét egy vizuális sáv - progress-bar - jeleníti meg, amelynek színe a kritériumok teljesülésétől függően változik.

Az oldal lehetővé teszi profilkép feltöltését egy külön API végponton - `http://localhost:5000/upload` - keresztül, ez a feltölteni kívánt kép előnézetét is megjeleníti. Elküldésekor a `handleSubmit` függvény ellenőrzi a megadott adatok érvényességét, valamint azt, hogy a felhasználónév és e-mail cím nem foglalt-e. Ha minden ellenőrzés sikeres, a komponens egy POST kérést indít az `/api/Felhasznalo` végpont felé, majd a felhasználót az `/allergen` útvonalra navigálja az ételérzékenységek megadásához.

A stílusokat a `Reg.css` fájl biztosítja, amely reszponzív dizájnért is felelős.

### **3.2.2g Allergia komponens**

Az Allergy komponens lehetővé teszi a felhasználók számára, hogy megadják ételérzékenységeiket, amelyeket az alkalmazás a receptszűrés során figyelembe vesz. Ez a komponens a regisztrációs folyamat része, de a profilbeállításokon keresztül később is módosítható. A felület kártyákat használ az allergének megjelenítéséhez.

A komponens egy előre definiált allergénlistát – `allergensList` – és a hozzájuk tartozó ikonokat használja. Inicializáláskor az `/api/Felhasznalo_Erzekenyseg/:userId` végpontból lekéri a felhasználó meglévő érzékenységeit, és ennek megfelelően előre bejelöli a megfelelő allergéneket. Az allergének kiválasztása jelölőnégyzetekkel történik, amelyek állapotát a `checkedStates` objektum tárolja.

"Mentés" gomb aktiválásakor a komponens:

1. Összegyűjti a kiválasztott allergéneket.
2. Ellenőrzi a felhasználó azonosítóját.

3. Menti az allergéneket az **/api/Felhasznalo\_Erzenyseg** végponton keresztül POST kérésekkel, és törli a már nem releváns érzékenységeket DELETE kérésekkel.
4. Navigál a megfelelő útvonalra, /taste-profile új regisztráció esetén, vagy /profile módosítás esetén, a localStorage változás kulcsa alapján.

A stílusokat az allergy.css biztosítja, ez felel a kártya alapú elrendezésért, illetve a responszív dizájnért.

### **3.2.2h Ízlés szekció**

A TasteProfile komponens lehetővé teszi a felhasználók számára, hogy megadják ízlésükkel kapcsolatos preferenciáikat tagek kiválasztásával, amelyeket az alkalmazás később a receptajánlások személyre szabására használ.

A komponens inicializálásakor a useEffect hook segítségével lekéri az összes elérhető taget az **/api/Tag** végponton keresztül. A felhasználók a tagekre kattintva választhatják ki preferenciáikat, amelyeket a selectedTags állapot tárol. A kiválasztás vizuális visszajelzést nyújt, mivel a kiválasztott tagek eltérő stílust kapnak a TasteProfile.css fájlban definiált osztályok alapján. A preferenciák mentése egy "Küldés" gomb megnyomásával történik, amely minden kiválasztott taget külön-külön elküld az **/api/Izles?F\_id=:userId** végpontnak egy POST kéréssel. A mentés után a komponens a bejelentkező oldalra navigál.

### **3.2.2i Felhasználóprofil szekció**

A Profile komponens a felhasználói profil kezelését biztosítja, megjelenítve a felhasználó adatait, feltöltött receptjeit és értékeléseit.

A komponens inicializáláskor a localStorage-ból olvassa ki a felhasználói adatokat. Az **/api/Ertekeles** végpontból lekéri a felhasználó értékeléseit, és az **/api/Recept/:recipeId** és **/api/Multimedia/:recipeId** végpontok segítségével betölti az értékelt recepteket és azoknak a képeit. A felhasználó által feltöltött recepteket az **/api/Recept** végpontból szűri a felhasználónév alapján.

A profil felület három fő szekciót tartalmaz: a felhasználói, az értékeléseket - csillagok szerint szűrhető kártyák - és a feltöltött recepteket. A beállítások módosítására egy modális ablak – SettingsModal - szolgál, amelyet egy fogaskerék ikon aktivál. Erről a komponensről a következő pontban lesz szó.

A stílusokat a Profile.css fájl biztosítja, amely a dizájnért, illetve a responszív elrendezésért felelős.

### 3.2.2j Beállítás szekció

A SettingsModal komponens egy modális ablak, amely lehetővé teszi a felhasználók számára jelszavuk és profilképeik módosítását, valamint az ételérzékenységek frissítését. Ez a komponens szorosan kapcsolódik a Profile komponenshez.

A komponens két fő állapotot kezel a newPassword az új jelszót, míg a newProfilePicture a feltöltött kép nevét tárolja. A jelszó módosításakor a komponens érvényesíti a jelszó erősségét. A profilképek feltöltése a `http://localhost:5000/upload` végponton keresztül történik, míg a módosítások mentése PATCH kérésekkel frissíti a felhasználói adatokat az `/api/Felhasznalo/:userId` végponton.

A stílusokat a SettingsModal.css fájl biztosítja.

### 3.2.2k Összes recept megtekintése szekció

A ViewAll komponens az alkalmazás receptböngésző felülete, amely lehetővé teszi a felhasználók számára, hogy az összes receptet megtekintsék, szűrjenek és lapozzanak közöttük. A komponens inicializálásakor számos API végpont kerül meghívásra, hogy betöltsük

- a recepteket - `/api/Recept`,
- tageket - `/api/Recept_Tag`, `/api/Tag`,
- kategóriákat - `/api/Kategoria`,
- értékeléseket - `/api/Ertekeles`,
- hozzávalókat - `/api/Recept_Hozzavalo`,
- a felhasználó ételérzékenységeit - `/api/Felhasznalo_Erzenyseg/:userId`,  
`/api/Hozzavalo_Erzenyseg/:sensitivityId`,
- a receptekhez tartozó képeket pedig az `/api/Multimedia/:recipeId` végpont biztosítja.

A szűrőfelület egy keresőmezőt, két legördülő menüt - tag és kategória - és egy ételérzékenységi kapcsolót tartalmaz, amely lehetővé teszi a felhasználók számára, hogy preferenciáik alapján testreszabják a találatokat. A szűrési logika lehetővé teszi a keresést név, szűrési tagek, kategóriák és ételérzékenységek alapján. A lapozást egy külön Pagination komponens kezeli, amely dinamikusan frissül a szűrt receptek száma alapján.

A komponens vizuális elemei közé tartoznak a receptkártyák, amelyek gyors információkat - elkészítési idő, hozzávalók száma, leírás - és értékeléseket jelenítenek meg csillagokkal. Ezekre kattintva a felhasználót a `/recipeDetails/:recipeId` oldalra navigálja és ott megtudja részletesebben tekinteni az adott receptet.



### **3.2.2l Mi van a hűtődben? szekció**

A WhatsInYourFridge komponens az alkalmazás egyik leginnovatívabb funkciója, amely lehetővé teszi a felhasználók számára, hogy a hűtőjükben lévő alapanyagok alapján recepteket keressenek. Ez a komponens különösen hasznos azok számára, akik inspirációt keresnek a meglévő hozzávalóik felhasználásához.

A komponens egy olyan űrlapot kínál, ahol a felhasználók megadhatják a hozzávalók nevét, mennyiségét és mértékegységét. Az autocomplete funkció az **/api/Hozzavalo** végponton keresztül lekért adatok alapján javaslatokat tesz, míg a mértékegységeket az **/api/Mertekegység\_Hozzavalo/:ingredientId** végpont biztosítja. A hozzáadott hozzávalók egy listában jelennek meg, a felhasználók egy szűrési százalékot is megadhatnak, amely meghatározza, hogy a receptek hány százalékban feleljen meg a megadott hozzávalóknak.

A keresés során a komponens az **/api/Recept** és **/api/Recept\_Hozzavalo/:recipeId** végpontokat használja a receptek és hozzávalók lekérdezéséhez, és az **/atvaltas** végpont segítségével konvertálja a mértékegységeket, ha szükséges.

A talált receptek színkódolással jelennek meg: zöld, ha minden hozzávaló elérhető, sárga, ha legfeljebb két hozzávaló hiányzik, és piros, ha több hiányzik. A receptkártyák értékeléseket is tartalmaznak, amelyeket az **/api/Ertekeles** végponton kérünk le.

A komponens stílusa a WhatsInYourFridge.css és Upload.css fájlokból származik, ezek biztosítják a kinézetet, illetve a reszponzív megjelenést.

A WhatsInYourFridge komponens erőssége a komplex szűrési és konverziós logikában rejlik.

### **3.2.2m Válaszra váró receptek szekció**

A PendingRecipes komponens az adminisztrátorok számára készült, és a jóváhagyásra váró recepteket jeleníti meg. Ez a komponens lehetővé teszi az adminisztrátorok számára, hogy áttekinthessék a felhasználók által feltöltött recepteket, mielőtt azok nyilvánosan elérhetővé válnak.

A komponens inicializálásakor az **/api/Recept** végponton keresztül lekéri az összes receptet, majd szűri azokat az ``Allapot == 0`` feltétel alapján, amely a jóváhagyásra váró állapotot jelzi. Minden recepthez külön lekéri a hozzá tartozó képet az **/api/Multimedia/:recipeId** végponton keresztül, és a recepteket egy rácsos elrendezésben jeleníti meg. A receptkártyák kattinthatóak, és az **/adminRecipeDetails/:recipeId** útvonalra navigálnak, átadva a recept adatait a state-ben.

A stílusokat a ViewAll.css és BlogSection.css fájlok biztosítják.

### **3.2.2n Recept részletek szekció**

A RecipeDetails komponens egy adott recept részletes nézetét biztosítja, lehetővé téve a felhasználók számára, hogy megtekintsék a recept hozzávalóit, elkészítési lépéseit, tageket, valamint értékeléseket adjanak vagy módosítsanak. Ez a komponens kulcsfontosságú az alkalmazás receptkezelési funkcionalitásában mivel interaktív élményt nyújt.

A komponens a recept adatait a react-router-dom useLocation és useParams hook-jaiból nyeri, lehetővé téve, hogy az adatokat vagy az útvonal paramétereiből, vagy a navigáció során átadott state-ből szerezze be.

Az inicializálás során a useEffect hook segítségével lekérjük

- a recept lépéseit - **/api/Lepes/:recipeId**,
- hozzávalóit - **/api/Recept\_Hozzavalo/:recipeId**,
- értékeléseit - **/api/Ertekeles/:recipeId**
- és a tageket - **/api/Recept\_Tag/:recipeId**.

Az értékelési rendszer lehetővé teszi, hogy a felhasználók csillagokkal pontozzák a receptet, és a komponens automatikusan kezeli, hogy új értékelést adjanak - **POST /api/Ertekeles** - vagy meglévő értékelést szeretnének módosítani - **PUT /api/Ertekeles/:ratingId**.

A stílusokat a RecipeDetails.css fájl biztosítja, amely reszponzív elrendezést és interaktív elemeket támogat.

### **3.2.2o Admin recept részletek szekció**

Az AdminRecipeDetails komponens az adminisztrátorok számára biztosít felületet a jóváhagyásra váró receptek szerkesztésére és jóváhagyására vagy elutasítására.

A komponens a recept adatait a useParams útvonal paraméterből nyeri, és több API végpontot hív meg inicializáláskor:

- a lépéseket - **/api/Lepes/:recipeId** -,
- a hozzávalókat - **/api/Recept\_Hozzavalo/:recipeId** -,
- a kategóriákat - **/api/Kategoria** -,
- a mértékegységeket - **/api/Mertekegyseg** -,
- és a tageket tölti be - **/api/Recept\_Tag/:recipeId**.

A recept adatai szerkeszthetők, beleértve a nevet, leírást, kategóriát, nehézségi szintet, hozzávalók mennyiségét, illetve mértékegységét, lépéseket és tageket.

Az "Elfogadás" gomb a receptet jóváhagyott állapotba helyezi - Allapot: 1 egy PUT kéréssel az **/api/Recept/:recipeId** végponton keresztül -, miközben frissíti a hozzávalókat és lépéseket külön kérésekkel. Az "Elutasítás" gomb törli a receptet - DELETE **/api/Recept/:recipeId** - és a hozzá tartozó képet - <http://127.0.0.1:5000/delete/:filename>.

### **3.2.2p Feltöltés szekció**

Az Upload komponens a receptfeltöltési folyamatot kezeli, lehetővé téve a felhasználók számára, hogy új recepteket hozzanak létre és mentsenek az alkalmazás adatbázisába.

A komponens inicializálásakor több API hívást indít a szükséges adatok betöltésére. Az **/api/Tag** végpontból lekéri az elérhető tageket, az **/api/Kategoria** végpontból a kategóriákat, az **/api/Hozzavalo** végpontból pedig az összes hozzávalót az autocomplete funkció támogatásához. A felhasználói adatokat a localStorage-ból olvassa ki – mint például a felhasználó azonosítója. A komponens számos állapotot kezel a useState hook-okkal, például a recept címét, leírását, hozzávalókat, lépéseket, tageket, képet.

A receptfeltöltési űrlap ikonokkal támogatva a vizuális eligazodást és a hozzávalók hozzáadása során egy autocomplete funkció segít a meglévő hozzávalók kiválasztásában a felhasználót.

A lépések és tagek listázása dinamikus, lehetővé téve a felhasználók számára, hogy könnyen bővítsék vagy szerkesszék azokat. A képfeltöltés egy külön API végponton - <http://127.0.0.1:5000/upload> - keresztül történik - ez a projektünk backendjének a python része -, amely a feltöltött kép elérési útját adja vissza.

A recept mentése során a komponens több egymást követő API hívást indít:

1. először a recept alapadatait menti az **/api/Recept** végponton keresztül,
2. majd a hozzávalókat - **/api/Recept\_Hozzavalo**,
3. lépéseket - **/api/Lepes**,
4. tageket - **/api/Recept\_Tag**
5. és a képet - **/api/Multimedia**

Ha a feltöltés sikeres, az uploadSuccess modális ablak jelenik meg.

A komponens stílusa az Upload.css fájlban van definiálva, ez felel az oldal reszponzivitásáért is.

### **3.2.2q Lapozás szekció**

A Pagination komponens a nagy adathalmazok (például receptek) lapozását kezeli, lehetővé téve a felhasználók számára, hogy könnyen navigáljanak az oldalak között. Ez a komponens különösen fontos a ViewAll komponensben, ahol sok recept megjelenítése szükséges.

A komponens három prop-ot fogad: a currentPage az aktuális oldalt, a totalPages az összes oldal számát, és az onPageChange egy függvény, amely az oldalváltást kezeli. A komponens dinamikusan generál gombokat az oldalhoz, és külön gombokat biztosít az „Előző” és „Következő” navigációhoz, amelyek le vannak tiltva, ha nem érvényesek.

A stílusokat a Pagination.css fájl biztosítja, amely kiemeli az aktuális oldalt és reszponzív elrendezésért felelős.

### **3.2.2r Moduláris ablak a felhasználónak való visszajelzésre**

A Success komponens egy moduláris ablak, amely egy végrehajtott művelet sikerességéről nyújt tájékoztatást a felhasználóknak. Ez a komponens testre szabható címmel, leírással, gombszöveggel és átirányítási célponttal működik. A modális ablak akkor jelenik meg, amikor egy művelet - például receptfeltöltés - sikeresen befejeződik, és célja, hogy egyértelmű visszajelzést nyújtson a felhasználónak.

A komponens öt prop-ot fogad:

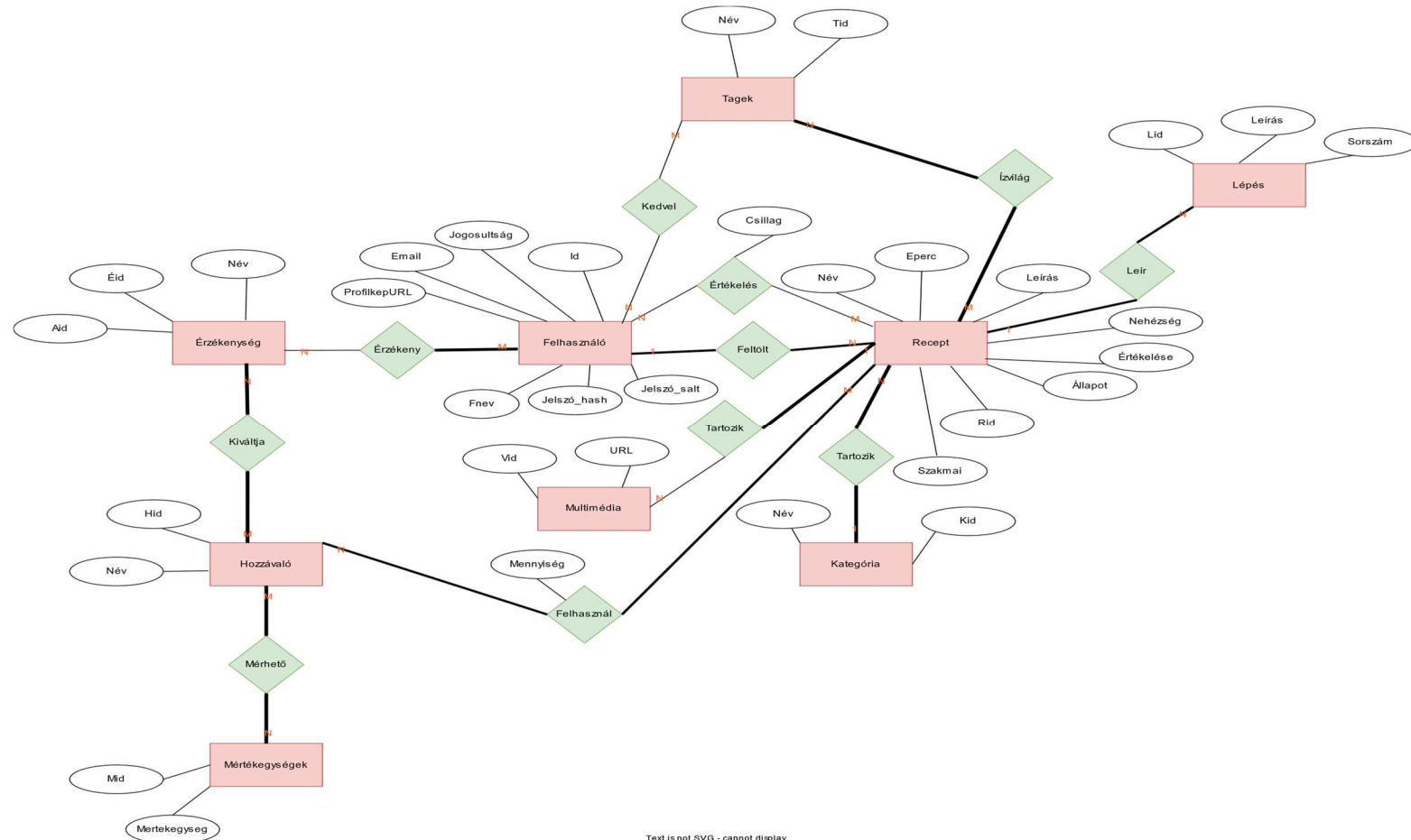
- isOpen: Logikai érték, amely meghatározza, hogy a modális látható-e.
- onClose: Függvény, amely a modális bezárását kezeli.
- title: A modális címe - például "Recept sikeresen feltöltve!".
- description: A modális részletes leírása - például a moderációs folyamat ismertetése.
- buttonText: A bezáró gomb szövege - például "Megértettem".
- relocate: Az URL, ahová a modális tartalomra kattintva a felhasználó átirányítódik.

Ha az isOpen értéke false, a komponens elrejteti magát. A modális tartalma egy testreszabható cím, egy SVG egér ikon - mouse.svg -, egy leíró szöveg és egy gomb. Az overlay-re kattintva a modális bezárul, míg a tartalmi területre kattintás a relocate prop-ban megadott URL-re navigál - például /upload egy újabb receptfeltöltéshez. A gomb az onClose függvényt hívja meg, és ezzel bezárja a modálist.

A SettingsModal.css és Upload.css fájlok biztosítják a stílusokat, reszponzív elrendezést.

## 4. Az adatbázis felépítése

### 4.1 Adatbázis ER modell ábra



Text is not SVG - cannot display

### 4.1.1 Adatbázis adatmodel leírása

A „recept” adatbázis tartalmazza minden minden egyed leképezését. Az adatbázis 9 egyed-típust tartalmaz, melyek tárolják az oldal működéséhez szükséges információkat. Az adatbázis 16 különálló táblát tartalmaz, melyek közül 7 „kapcsolótábla”, ezáltal megvalósítva a több-több kapcsolatokat.

Minden egyed rendelkezik egy egyedi azonosítóval, – elsődleges kulccsal – ezáltal minden rekord megkülönböztethető egymástól. A kapcsolótáblák azonosítója egy összetett kulcs, amely a táblák idegenkulcsaiból áll össze.

Az adatmodel a 3. normálforma elvárásainak felel meg, tehát egy – nem kulcs – tulajdonság sem függ olyan mezőtől, amely nem kulcs tulajdonságú, mindemellett pedig 2. és 1. normálformában is van.

### 4.1.2 Az adatbázis tábláinak bemutatása

#### 4.1.2.a Értékelések

Az adatmodel ezen egyede tárolja az összes recepthez tartozó értékeléseket. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Ertid</i>	int(10)	Elsődleges kulcs	Az értékelés azonosítója
<i>Csillag</i>	int(2)	-	Egy adott értékelés értéke
<i>R_id</i>	int(10)	Idegenkulcs	A recept azonosítója
<i>F_id</i>	int(10)	Idegenkulcs	A felhasználó azonosítója

Egy adott felhasználó egy receptre csak egy értékelést adhat le. Azonban minden felhasználó több receptet is értékelhet.

Egy adott rekord az *Ertid* tulajdonság alapján egyértelműen azonosítható. Ezen tulajdonság „auto increment”, ennél fogva az adatbázis automatikusan állít elő értéket számára. Új rekord felvételekor ezen mező kitöltése nem kötelező.

A rekordok egyik tulajdonsága sem lehet null, ezért minden mező értéke – A kulcs tulajdonság kivételével – kötelezően kitöltendő.

#### 4.1.2.b Érzékenységek

Az adatmodel ezen egyede tárolja az összes érzékenység osztályt. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Eid</i>	int(10)	Elsődleges kulcs	Az érzékenység azonosítója
<i>Nev</i>	varchar(50)	-	Egy adott érzékenység neve

Az adatbázis 14 különböző érzékenység osztályt határoz meg.

Egy adott rekord az *Ertid* tulajdonság alapján egyértelműen azonosítható. Ezen tulajdonság „auto increment”, ennél fogva az adatbázis automatikusan állít elő értéket számára. Új rekord felvételekor ezen mező kitöltése nem kötelező.

A rekordok egyik tulajdonsága sem lehet null, ezért minden mező értéke – A kulcs tulajdonság kivételével – kötelezően kitöltendő.

#### 4.1.2.c Felhasználók

Az adatmodel ezen egyede tárolja az összes regisztrált felhasználót. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Id</i>	int(10)	Elsődleges kulcs	A felhasználó azonosítója
<i>Fnev</i>	varchar(60)	-	A felhasználó felhasználóneve
<i>Email</i>	varchar(200)	-	A felhasználó e-mail címe
<i>Jelszo_Salt</i>	blob	-	A felhasználó jelszava titkosítva
<i>Jelszo_Hash</i>	blob	-	A felhasználó jelszava titkosítva
<i>Jogosultsag</i>	int(3)	-	A felhasználó jogosultsága
<i>ProfilkepURL</i>	varchar(500)	-	A felhasználó profilképének elérési útja

Egy adott rekord az *Id* tulajdonság alapján egyértelműen azonosítható. Ezen tulajdonság „auto increment”, ennél fogva az adatbázis automatikusan állít elő értéket számára. Új rekord felvételekor ezen mező kitöltése nem kötelező.

A rekord *ProfilkepURL* tulajdonsága lehet null, ennél fogva új rekord felvételekor ezen tulajdonság kitöltése nem kötelező. Minden más attribútum azonban kötelező.

A jogosultság mező tárolja, hogy egy adott felhasználó milyen jogkörrel rendelkezik. Egy felhasználó vagy admin jogú – ekkor a jogosultság mező értéke 1-, vagy átlag felhasználó. – ekkor a mező értéke 2

#### 4.1.2.d Felhasználó\_Érzékenység

Az adatmodel ezen egyede tárolja a felhasználók különböző érzékenységeit. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>F_id</i>	int(10)	Összetett kulcs	A felhasználó azonosítója
<i>E_id</i>	int(10)		Egy adott érzékenység azonosítója

Egy felhasználó akár több érzékenységgel is rendelkezhet, viszont vannak felhasználók, akik egyel sem. Ezen tábla csak azokat a felhasználókat tartalmazza, akik legalább egy étel allergiával rendelkeznek.

A két mező együttesen egy összetett kulcsot alkotnak, így egy rekord azonosítása csak mindkét mező segítségével történhet.

#### 4.1.2.e Hozzávalók

Az adatmodel ezen egyede tárolja az összes hozzávalót. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Hid</i>	int(10)	Elsődleges kulcs	A hozzávaló azonosítója
<i>Nev</i>	varchar(50)	-	Egy adott hozzávaló neve

Egy adott hozzávaló az adatbázisban csak egyszer szerepelhet.

Egy adott rekord a *Hid* tulajdonság alapján egyértelműen azonosítható. Ezen tulajdonság „auto increment”, ennél fogva az adatbázis automatikusan állít elő értéket számára. Új rekord felvételekor ezen mező kitöltése nem kötelező.

#### 4.1.2.f Hozzávaló\_Érzékenységek

Az adatmodel ezen egyede tárolja, hogy a különböző hozzávalók mely érzékenységeket váltják ki. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>H_id</i>	int(10)	Összetett kulcs	A hozzávaló azonosítója
<i>E_id</i>	int(10)		Egy adott érzékenység azonosítója

Egy hozzávaló akár több érzékenységet is kiválthat, viszont vannak hozzávalók, amelyek egyet sem. Ezen tábla csak azokat a hozzávalókat tartalmazza, melyek legalább egy allergiát kiváltanak.

A két mező együttesen egy összetett kulcsot alkotnak, így egy rekord azonosítása csak mindkét mező segítségével történhet.



#### 4.1.2.g Ízlések

Az adatmodel ezen egyede tárolja, hogy a különböző felhasználók milyen típusú ételeket kedvelik. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>F_id</i>	int(10)	Összetett kulcs	A felhasználó azonosítója
<i>T_id</i>	int(10)		Egy adott tag azonosítója

A felhasználó ízlésprofiljának kialakításához a tageket használtuk. A felhasználók több ízvilágot is választhatnak, amelyeket kedvelnek.

A két mező együttesen egy összetett kulcsot alkotnak, így egy rekord azonosítása csak mindkét mező segítségével történhet.

#### 4.1.2.h Kategóriák

Az adatmodel ezen egyede tárolja az összes étel kategóriát. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Kid</i>	int(10)	Elsődleges kulcs	A kategória azonosítója
<i>Nev</i>	varchar(30)	-	Egy adott kategória neve

A receptek különböző kategóriákba soroljuk. Ezen adatmodel tartalmazza az összes kategóriát, amely megadható egy recept feltöltésénél.

Egy adott rekord a *Kid* tulajdonság alapján egyértelműen azonosítható. Ezen tulajdonság „auto increment”, ennél fogva az adatbázis automatikusan állít elő értéket számára. Új rekord felvételekor ezen mező kitöltése nem kötelező.

Egy kategória csak egyszer szerepelhet az adatbázisban.

#### 4.1.2.i Lépések

Az adatmodel ezen egyede tárolja az összes recept elkészítésének lépéseit. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Lid</i>	int(10)	Elsődleges kulcs	A lépés azonosítója
<i>Sorszam</i>	int(2)	-	Egy adott lépés sorszáma adott recepten belül
<i>Leiras</i>	varchar(500)	-	Egy adott lépés leírása
<i>R_id</i>	int(10)	Idegenkulcs	A recept azonosítója, melyhez a lépés tartozik

Minden recepthez tartoznak lépések annak elkészítéséhez. Egy recepthez legalább egy lépés tartozik.

Egy adott rekord a *Lid* tulajdonság alapján egyértelműen azonosítható. Ezen tulajdonság „auto increment”, ennél fogva az adatbázis automatikusan állít elő értéket számára. Új rekord felvételekor ezen mező kitöltése nem kötelező.

A rekordok egyik tulajdonsága sem lehet *null*, ezért minden mező értéke – A kulcs tulajdonság kivételével – kötelezően kitöltendő.

A sorszám érték mindig csak egy adott recepten belül érvényes. A mező értéke minden recept esetében 1-től kezdődik.

A leírás tulajdonság hossza maximum 500 karakter. Ennél hosszabb szöveg tárolására nem alkalmas.

#### 4.1.2.j Mértékegységek

Az adatmodel ezen egyede tárolja az összes megadható mértékegységet. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Mid</i>	int(10)	Elsődleges kulcs	A mértékegység azonosítója
<i>Mertekegy-segNev</i>	varchar(20)	-	Egy adott mértékegység neve

Az adatbázis csak a *metrikus rendszer* mértékegységeit tartalmazza.

#### 4.1.2.k Mértékegység\_Hozzávaló

Az adatmodel ezen egyede tárolja, hogy az egyes hozzávalók mely mértékegységekben adhatóak meg. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>M_id</i>	int(10)	Összetett kulcs	A mértékegység azonosítója
<i>H_id</i>	int(10)		Egy adott hozzávaló azonosítója

A két mező együttesen egy összetett kulcsot alkotnak, így egy rekord azonosítása csak mindkét mező segítségével történhet.

#### 4.1.2.l Multimédiák

Az adatmodel ezen egyede tárolja az összes kép file elérését a receptekhez. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Vid</i>	int(10)	Elsődleges kulcs	A kép azonosítója
<i>URL</i>	varchar(200)	-	Egy adott kép elérési útja
<i>R_id</i>	int(10)	Idegenkulcs	A recept azonosítója

Minden recepthez tartozik legalább egy kép. Az adatmodel mindösszesen a kép elérési útját tárolja, magát a kép file-t viszont nem.

Egy adott rekord a *Vid* tulajdonság alapján egyértelműen azonosítható. Ezen tulajdonság „auto increment”, ennél fogva az adatbázis automatikusan állít elő értéket számára. Új rekord felvételekor ezen mező kitöltése nem kötelező.

#### 4.1.2.m Receptek

Az adatmodel ezen egyede tárol minden feltöltött receptet. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Rid</i>	int(10)	Elsődleges kulcs	A recept azonosítója
<i>Nev</i>	varchar(100)	-	A recept neve
<i>Leiras</i>	varchar(500)	-	A recept leírása
<i>Allapot</i>	int(1)	-	A recept állapota
<i>Szakmai</i>	tinyint(1)	-	A recept „séf” által lett-e összeállítva
<i>Eperc</i>	int(11)	-	A recept elkészítésének ideje
<i>K_id</i>	int(10)	Idegenkulcs	A kategória azonosítója
<i>F_id</i>	int(10)	Idegenkulcs	A felhasználó azonosítója
<i>Neheszseg</i>	varchar(60)	-	A recept elkészítésének nehézsége

Egy adott rekord a *Rid* tulajdonság alapján egyértelműen azonosítható. Ezen tulajdonság „auto increment”, ennél fogva az adatbázis automatikusan állít elő értéket számára. Új rekord felvételekor ezen mező kitöltése nem kötelező.

A rekordok egyik tulajdonsága sem lehet *null*, ezért minden mező értéke – A kulcs tulajdonság kivételével – kötelezően kitöltendő.

A leírás tulajdonság hossza maximum 500 karakter. Ennél hosszabb szöveg tárolására nem alkalmas.

Az állapot tulajdonság feladata, hogy egy adott recept elfogadásra került-e egy adminisztrátor által vagy sem. Egy recept mindaddig nem jelenhet meg az oldalon amíg ezen mező nem 1-es értékkel rendelkezik.

A szakmai tulajdonság értéke jelzi, hogy az adott recept egy olyan személy által lett-e összeállítva, aki egy elismert séf vagy szakács.

A nehézség tulajdonság egy szöveges mező, amely három értékkel rendelkezhet:

- Könnyű : A recept elkészítése egyszerű
- Közepes : A recept elkészítéséhez némi tapasztalatra van szükség
- Nehéz : A recept elkészítése kezdők számára nem ajánlott

#### 4.1.2.n Recept\_Hozzávaló

Az adatmodel ezen egyede tárolja, hogy az adott receptek elkészítéséhez mely hozzávalók szükségesek. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Mennyiség</i>	double	-	Egy adott hozzávalóból mennyire van szükség
<i>R_id</i>	int(10)	Összetett kulcs	A recept azonosítója
<i>H_id</i>	int(10)		Egy adott hozzávaló azonosítója
<i>M_id</i>	int(11)	Idegenkulcs	Az adott hozzávaló mely mértékegységben lett megadva

Minden recept tartalmaz legalább egy hozzávalót. Ezt a *H\_id* tulajdonság határozza meg.

Két mező – *R\_id* és *H\_id* – együttesen egy összetett kulcsot alkotnak, így egy rekord azonosítása csak mindkét mező segítségével történhet.

A mennyiség tulajdonság egy decimális számot tartalmazó mező, amely az adott hozzávalóból szükséges mennyiséget tárolja.

Az *M\_id* tulajdonság határozza meg, hogy egy meghatározott recept esetében az adott hozzávaló mely mértékegységben lett megadva.

#### 4.1.2.o Recept\_Tag

Az adatmodel ezen egyede tárolja a receptekhez tartozó tageket. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>R_id</i>	int(10)	Összetett kulcs	A recept azonosítója
<i>T_id</i>	int(10)		Egy adott tag azonosítója

Minden recepthez tartozik legalább egy tag, amely a recept ízvilágát hívatott jelezni.

A két mező együttesen egy összetett kulcsot alkotnak, így egy rekord azonosítása csak mindkét mező segítségével történhet.

#### 4.1.2.p Tagek

Az adatmodel ezen egyede tárolja az összes taget. Szerekezeti felépítése a következő:

Mező név	Típus	Index	Leírás
<i>Tid</i>	int(10)	Elsődleges kulcs	A tag azonosítója
<i>Nev</i>	varchar(50)	-	Egy adott tag neve

A tagek feladata az ízvilágok meghatározása.

Egy adott rekord a *Rid* tulajdonság alapján egyértelműen azonosítható. Ezen tulajdonság „auto increment”, ennél fogva az adatbázis automatikusan állít elő értéket számára. Új rekord felvételekor ezen mező kitöltése nem kötelező.

A rekordok egyik tulajdonsága sem lehet *null*, ezért minden mező értéke – A kulcs tulajdonság kivételével – kötelezően kitöltendő.

## 5. Működésének műszaki feltételei

### 5.1 Kötelezően telepítendő programok

Projektünk elindításához szükséges, hogy a számítógépen telepítve legyen az adatbázist és annak adatainak importálást lehetővé tevő **webszerver**. Ennek megvalósítására mi a Xampp-ot használtuk, azonban minden más erre a célra kialakított program is használható, mint például:

- dbForge Studio
- MAMP
- Laragon

Az oldalon a képek feltöltésére használt backend szerver **python**-ban lett megvalósítva, így annak futtatásához szükséges, hogy a gépen telepítve legyen a python. A projekt ennek megléte nélkül is elindítható, és a funkciók többsége hiba nélkül működik is, azonban a kép file-ok feltöltéséhez elengedhetetlen.

A frontend Vite-ban lett elkészítve, amely egy **NodeJS**-en alapuló keretrendszer. Ennél fogva a számítógépen telepítve kell, hogy legyen a NodeJS. Ennek hiányában a projekt nem indítható el.

A Backend projekt futtatásához legalább a **Visual Studio 2019**–es verziója szükséges, azonban ettől újabb verzióval is kompatibilis.

## 6. Használatának bemutatása

### ***Kezdő oldal***

Az oldal betöltődését követően a felhasználó a kezdőoldalra kerül. Az oldal alján található szekcióban olyan receptek jelennek meg, amelyek az adott felhasználó ízlésprofiljának felelnek meg. – *Lásd: 14.kép*

### ***Fejléc***

Az oldalon való navigálásra a fejléc használható, mellyel az oldal bármely részére képes elnavigálni a felhasználó. Ezen fejléc minden oldalon elérhető, ezzel megkönnyítve az oldalak közti váltást.

### ***Regisztráció és bejelentkezés***

Az oldal bizonyos funkciói felhasználói fiókhoz vannak kötve. Így ezen funkciók csak bejelentkezett felhasználók számára elérhetőek. Ilyen funkció a recept feltöltés, a receptek értékelése, valamint az érzékenységek beállítása.

Amennyiben nincs bejelentkezve egyetlen felhasználó sem, a profil ikonra kattintva a regisztráció, valamint a bejelentkezés opciók jelennek meg.

Egy fiók regisztrálásához egy felhasználónévre, illetve egy e-mail címre van szükség. Mindemellett pedig egy olyan jelszóra, amely megfelel az oldalon feltüntetett kritériumoknak. – *Lásd: 15.kép*

A profilképre kattintva lehetőség van saját profilkép beállítására.

Sikeres regisztrációt követően az allergénes oldalra kerül, ahol a felhasználó kiválaszthatja, hogy rendelkezik-e valamely ételallergiával. A mentés gombra kattintva mentheti azokat. – *Lásd: 16.kép, 17.kép*

Ezek után az ízlésprofil kialakítására van lehetőség. A felhasználó kiválaszthat olyan tageket, amely ízvilágú ételeket kedveli. – *Lásd: 18.kép*

A bejelentkezéshez pedig a felhasználónak egy érvényes e-mail cím – jelszó párosra van szüksége. Sikeres bejelentkezést követően a felhasználó a kezdő oldalra kerül. – *Lásd: 19.kép*

### ***Receptek feltöltése***

A fejlécben található „Recept feltöltés” gombra kattintva juthat el a felhasználó a receptek feltöltésére szolgáló oldalra. Itt több különböző adatot adhat meg egy-egy receptről a felhasználó, – melyek többségének kitöltése kötelező – köztük pedig saját képet is tölthet fel az elkészített receptről. – *Lásd: 20.kép, 21.kép* - A kép feltöltése nem kötelező, ebben az esetben az oldal az alapértelmezett képet foglya használni, mikor a recept megjelenik a többi felhasználó számára. Egy feltöltött recept nem jelenik meg az oldalon azonnal, hanem előbb



egy adminisztrátornak ellenőriznie és jóváhagynia kell a receptet, mielőtt az látható a többi felhasználó számára is.

### ***Mi van a hűtődben?***

A felhasználó a fejlécből könnyedén eljuthat a „Mi van a hűtődben?” részére az oldalunknak. Ezen az oldalon megadhat olyan alapanyagokat és azok mennyiségét, amely a rendelkezésre áll, vagy amelyet szeretne felhasználni. Ezután a keresés gombra kattintva az oldal ajánl olyan recepteket, amelyek tartalmazzák ezen hozzávalókat.

Az oldalon beállítható, hogy a receptek hozzávalóinak mekkora százalékkal kell rendelkezzen a felhasználó, hogy az oldal ajánlasként megjelenítse. Ezen százalék alapértelmezettként 80-ra van beállítva, azonban 10 és 100 százalék között a felhasználó szabadon állíthatja be ezen értéket. – *Lásd: 22.kép, 23.kép*

### ***Összes recept***

A kezdő oldal, illetve a fejléc segítségével is könnyedén eljuthat a felhasználó az oldal azon részére, ahol a rendelkezésreálló összes receptet megtalálhatja.

Ezen az oldalon a felhasználó kereshet név alapján egy-egy adott receptet, vagy beállíthat különböző szűrő feltételeket is. Ilyen szűrők a különböző tag-ek, és kategóriák.

Emellett, ha a felhasználó rendelkezik valamilyen allergiával – és regisztrációkor vagy a profil beállításoknál megadta azt -, akkor az oldalon megjelenik egy extra gomb, amellyel szabályozhatja, hogy jelenjenek-e meg olyan receptek, amelyek számára érzékenységet válthatnak ki vagy sem. Alapértelmezettként az oldalon nem jelennek meg azok a receptek, amelyek a felhasználó számára érzékenységet váltanak ki.

A különböző receptek „kártyákként” jelennek meg, melyen a recept neve, képe, elkészítési ideje, hozzávalóinak darabszáma, leírásának első pár sora, illetve az értékelése – amennyiben rendelkezik értékeléssel – látható. – *Lásd: 24.kép, 25.kép* - Ezekre kattintva nyithatja meg a felhasználó az adott recept részletes leírását.

### ***Receptek részletei***

Ezen oldal szolgál arra, hogy a felhasználók megtekinthessék egy-egy adott recept részletes leírását.

Ezen az oldalon jelennek meg a receptek tagjai, – Minden tag rendelkezik legalább egyel, de akár többel is -, kategóriája, elkészítési ideje, nehézsége. Mindemellett megjelenik minden hozzávaló, amelyre szükség van a recept elkészítéséhez, és azok mennyisége.

A receptekhez tartoznak az elkészítéshez szükséges lépések leírásai is.

Egy recept értékelésére is ezen az oldalon van lehetőség. Amennyiben az adott felhasználó már értékelt egy receptet, utána lehetősége van annak módosítására is. – *Lásd: 26.kép, 27.kép, 28.kép*

### ***Profil oldal***

A felhasználói profil oldalra a fejlécben található profilképpel ellátott gombra kattintva van lehetőség.

Ezen az oldalon találhatóak meg azok a receptek, amelyeket az adott felhasználó töltött fel. Ezen felül tud keresni azon receptek között is, amelyeket értékelt. – függetlenül attól, hogy milyen értékelést adott le –

Az oldalon található fogaskerék ikonra kattintva a felhasználó megváltoztathatja jelszavát, illetve profilképét, emellett pedig érzékenysége módosítására is van lehetőség. – *Lásd: 29.kép, 30.kép*

### ***Döntésre váró receptek***

A fejléc ezen menüpontja csak az adminisztrátor jogú felhasználók számára jelenik meg.

Ezen az oldalon a felhasználók által feltöltött receptek jelennek meg, amelyeket még egy admin jogú felhasználó sem hagyott jóvá. A receptek „kártya”-ként jelennek meg. Ezekre kattintva léphet át a felhasználó az adott recept ellenőrzésére szolgáló oldalra. – *Lásd: 31.kép*

### ***Döntésre váró receptek részletei***

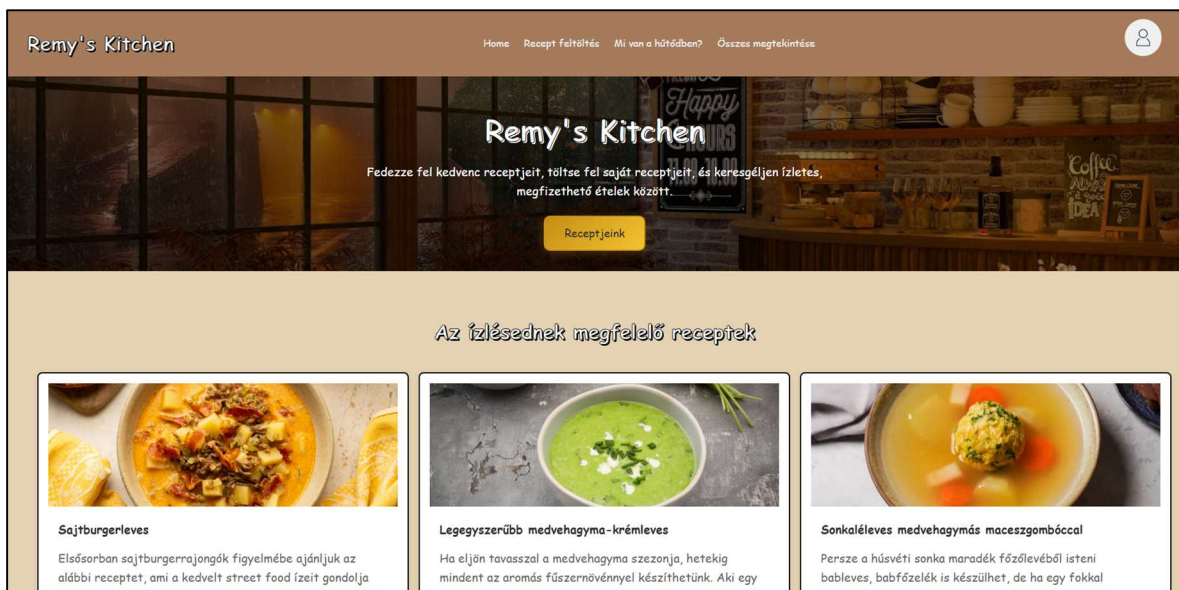
Ezen oldalon az adminisztrátor jogú felhasználók szabadon elbírálhatják, hogy egy adott recept felkerülhet-e az oldalra a többi felhasználó számára vagy sem. Az oldalon az adminok módosíthatják a recept nevét és leírását, illetve a recept kategóriáját, valamint nehézségi szintjét.

Ezen felül módosíthatják a megadott hozzávalókhöz tartozó mennyiséget és mértékegységet is. – Ezen az oldalon az összes mértékegység megjelenik, nem csak azok, amelyek egy adott hozzávalóhoz vannak rendelve az adatbázisban. Így az admin szabadon beállíthat bármilyen mértékegységet bármely hozzávalóhoz. –

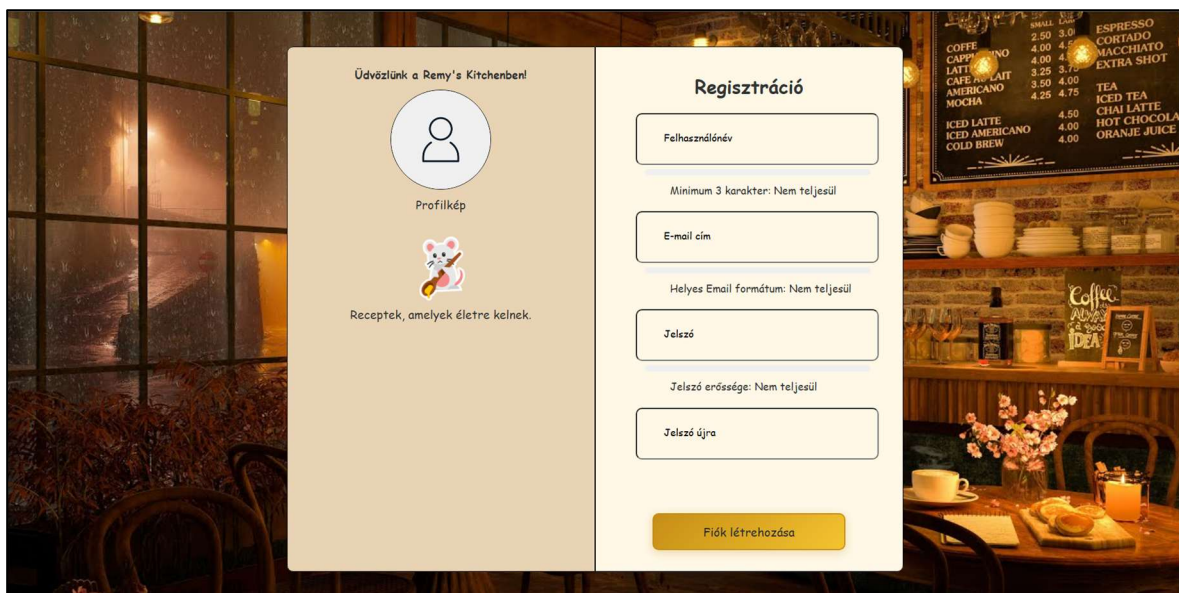
Az adminisztrátorok módosíthatják a recept lépéseinek leírását is, valamint képesek egy-egy tag törlésére is.

Amennyiben a recept megfelel minden kritériumnak, az elfogadás gombra kattintva a recept állapota megváltozik és onnantól megjelenik az oldalon más felhasználók számára is. Ellenkező esetben pedig a recept és annak minden tulajdonsága törlésre kerül. – *Lásd: 32.kép, 33.kép, 34.kép*

## Monitorképek



14. kép: Kezdő oldal








15. kép: Regisztráció

**Válasszon allergént**

<p>Glutén</p>  <p><input type="checkbox"/> 1.</p>	<p>Rákfélék</p>  <p><input type="checkbox"/> 2.</p>	<p>Tojás</p>  <p><input type="checkbox"/> 3.</p>
<p>Hal</p>  <p><input type="checkbox"/> 4.</p>	<p>Földimogyoró</p>  <p><input type="checkbox"/> 5.</p>	<p>Szójabab</p>  <p><input type="checkbox"/> 6.</p>
<p>Tej</p> 	<p>Diófélék</p> 	<p>Zeller</p> 

16. kép: Allergénes oldal

<input type="checkbox"/> 7.	<input type="checkbox"/> 8.	<input type="checkbox"/> 9.
<p>Mustár</p>  <p><input type="checkbox"/> 10.</p>	<p>Szezám</p>  <p><input type="checkbox"/> 11.</p>	<p>Kén-dioxid</p>  <p><input type="checkbox"/> 12.</p>
<p>Csillagfűt</p>  <p><input type="checkbox"/> 13.</p>	<p>Puhatestűek</p>  <p><input type="checkbox"/> 14.</p>	
<p>Mentés</p>		

17. kép: Allergénes oldal

### Milyen ízvilágú ételeket kedvelsz?

Magyaros	Vegetáriánus	Gyors
Diétás	Célpés	Édes
Sós	Csokis	Gluténmentes
Laktózmentes	Cukormentes	Alkoholos
Tésztás	Ázsiai ízvilágú	Európai ízvilágú
Amerikai ízvilágú	Egészséges	Vegán

Küldés

18. kép: Ízlés profil kialakítása

### Bejelentkezés

Belépés

Még nincs fiókod?

Fiók létrehozása

© 2024 Remy's Kitchen. All rights reserved.  
Contact us: info@remykt.com

19. kép Bejelentkezés

Remy's Kitchen
Home Recept feltöltés Mi van a hűtődben? Összes megtekintése Döntésre váró receptek

### Recept Feltöltése

**Recept neve:**

**Leírás:**

Válassz kategóriát

**Kategória**

0
Elkészítési idő

Könnyű
Nehézség

**Kép feltöltése:**

**Hozzávalók:**

0
Válassz mértékegységet

20. kép: Recept feltöltése

Hozzávalók:
0
Válassz mértékegységet

Hozzáadás

**Lépések:**

Hozzáadás

**Tagek:**

Válassz Taget
Hozzáadás

Mentés

21. kép: Recept feltöltése

Remy's Kitchen
Home Recept feltöltés Mi van a hűtődben? Összes megtekintése Döntésre váró receptek

### Mi van a hűtődben?

80
Hozzávaló
Mennyiség
Válassz mértékegységet
Hozzáad

**Alapanyagok:**

Sajnos nem találtunk megfelelő receptet

22. kép: Mi van a hűtődben?

62



Remy's Kitchen
Home Recept feltöltés Mi van a hűtődben? Összes megtekintése Döntésre váró receptek

5 - 5 csipet
Keresés

### Legegyszerűbb medvehagyma-krémleves

Ha eljön tavasszal a medvehagyma szezonja, hetekig mindent az aromás fűszernövényrel készíthetünk. Aki egy igazán gyors és komfortos receptet keres a zöldséggel, az alábbi krémlevesben találhatja megváltását, ami ráadásul lisztes sűrítés nélkül is isteni állagot kap. Sajtos pirítással a legfinomabb.

★★★★☆

**Hozzávalók:**

- Vöröshagyma: 1 db

### Sonkaléves medvehagymás maceszgombóccal

Persze a húsvéti sonka maradék főzolevéből isteni bableves, babfőzelék is készülhet, de ha egy fokkal egyszerűbb ételt főznénk belőle, válasszuk az alábbi receptet. Szerény zöldségekkel adhatunk gazdag aromát a füstös lének, amit medvehagymás maceszgombócok tesznek majd laktatóvá.

**Hozzávalók:**

- Sárgarépa: 2 db
- Frisz fehérrépa: 2 dkg
- Karalábé: 1 db
- Vöröshagyma: 1 db

### Tejszínes vegyes hagymakrémleves

Finom, könnyen elkészíthető hagymakrémleves.

**Hozzávalók:**

- Vöröshagyma: 2 db
- Fokhagyma: 4 gerezd
- Hagyma: 3 db
- Liszt: 40 g
- Habtejszín: 200 ml
- Víz: 1600 ml
- Só: 3 csipet
- Bors: 2 csipet
- Olivaj: 5 evőkanál

23. kép: Mi van a hűtődben? ajánlások

Remy's Kitchen
Home Recept feltöltés Mi van a hűtődben? Összes megtekintése Döntésre váró receptek

Összes recept

Keresés a receptek között...
Minden tag
Minden kategória

### Sajtburgerleves

50 Perc 17 Hozzávaló

Elsősorban sajtburgerrajongók figyelmébe ajánljuk az alábbi receptet, ami a kedvelt street...

★★★★★

### Legegyszerűbb medvehagyma-krémleves

25 Perc 7 Hozzávaló

Ha eljön tavasszal a medvehagyma szezonja, hetekig mindent az aromás...

★★★★★

### Sonkaléves medvehagymás maceszgombóccal

50 Perc 10 Hozzávaló

Persze a húsvéti sonka maradék főzolevéből isteni bableves, babfőzelék is készülhet, de ha...

★★★★★

### Kucsmagomba leves podagrafűvel

35 Perc 11 Hozzávaló

Egy üde gyertyánosban jártam, ahol együtt nőtt a méltatlanul nem ismert podagrafű és a...

★★★★★

### Rösti pizza

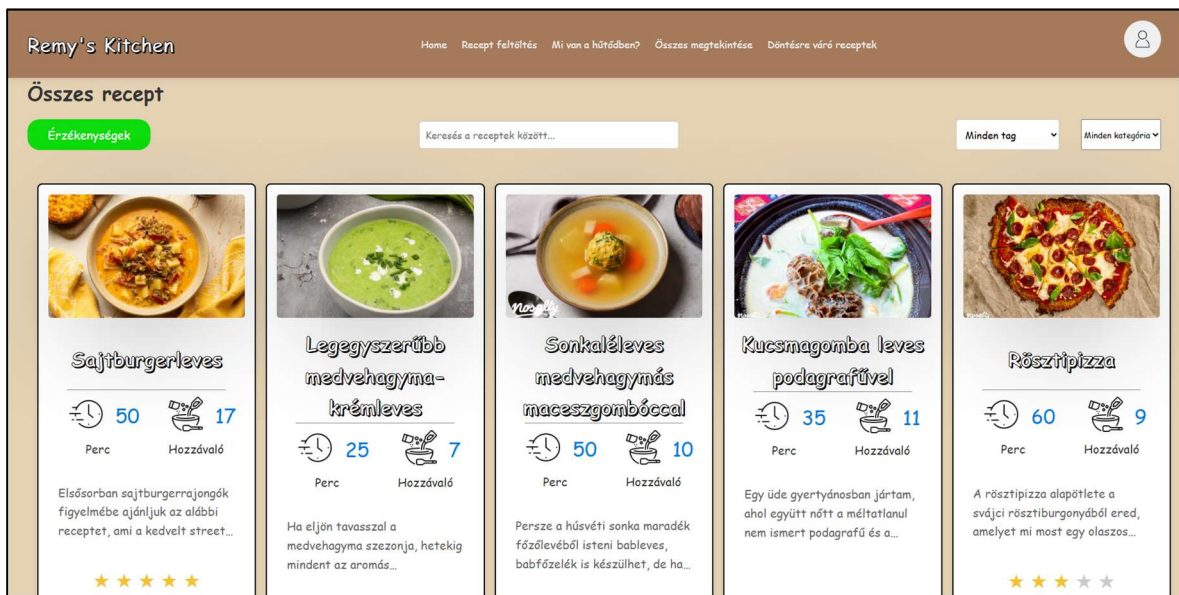
60 Perc 9 Hozzávaló

A rösti pizza alapötlete a svájci röstitburgonyából ered, amelyet mi most egy olaszos...

★★★★★

24. kép: Összes recept

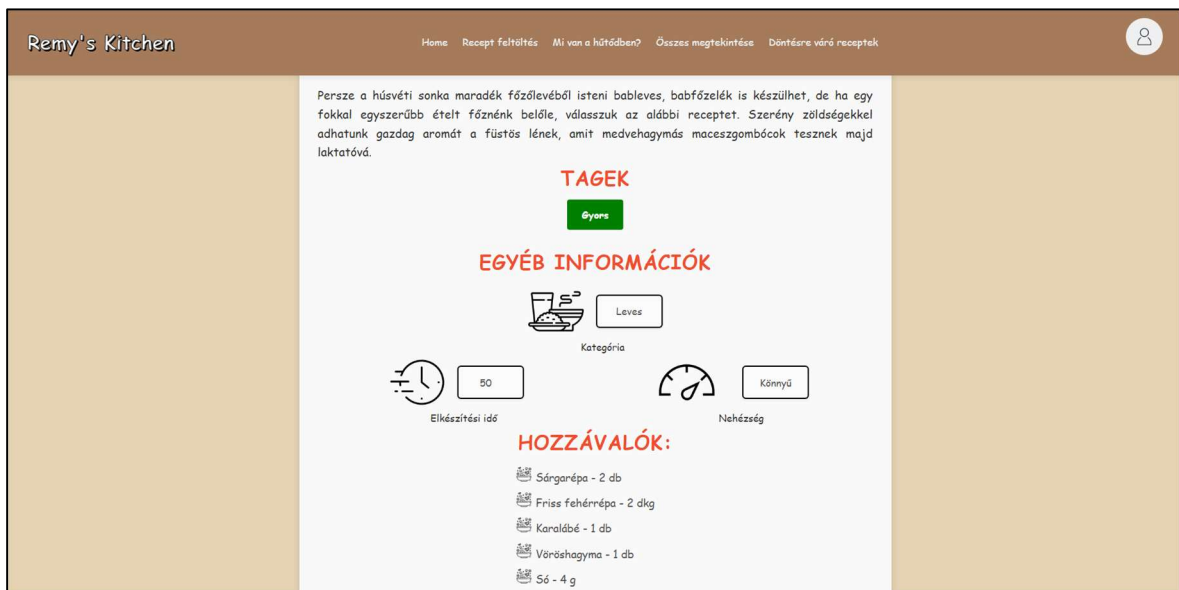




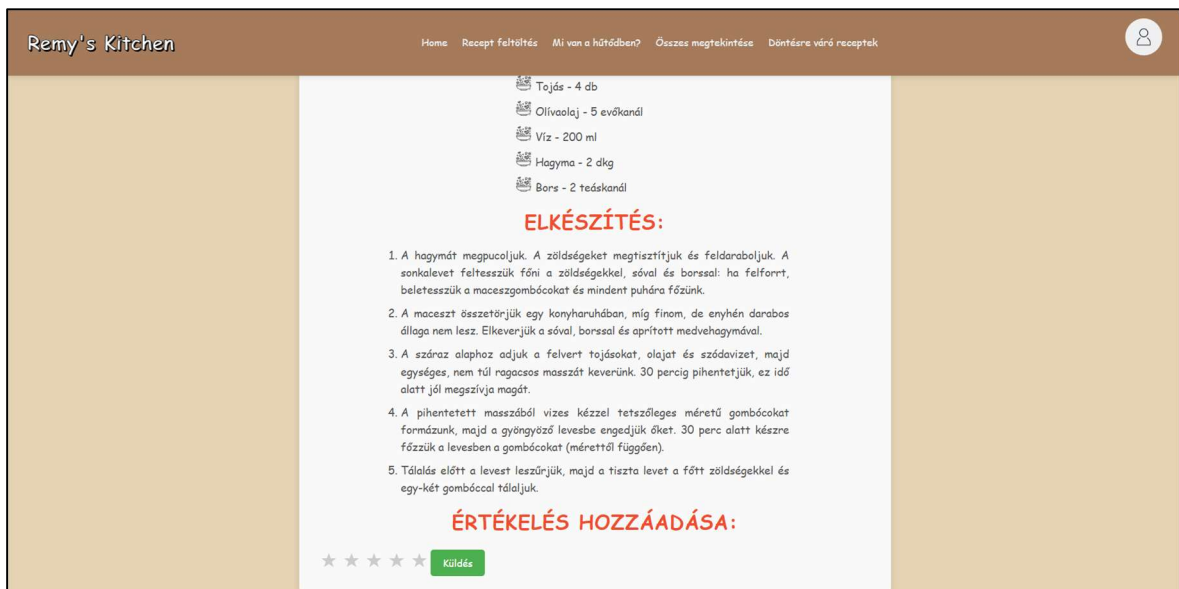
25. kép: Összes recept allegén gombbal



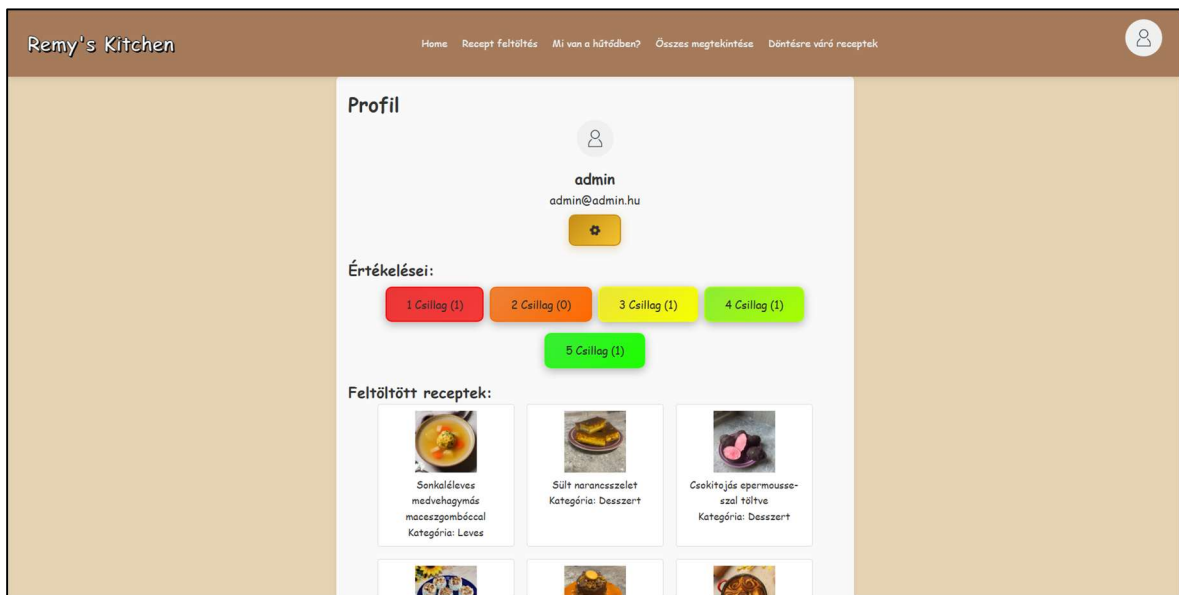
26. kép: Receptek részletei



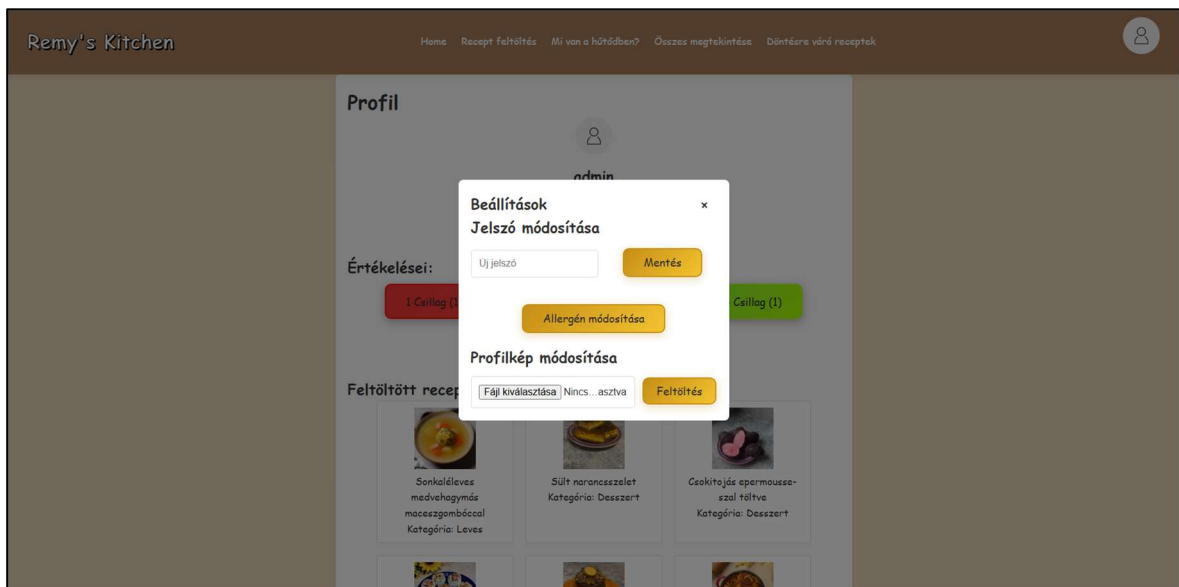
27. kép: Receptek részletei



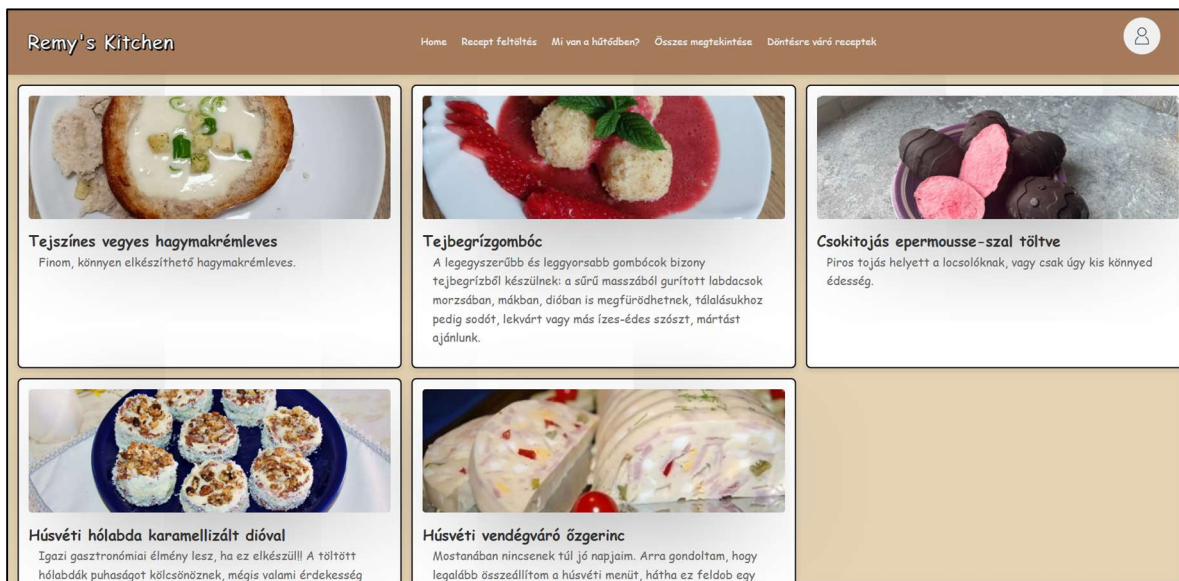
28. kép: Receptek részletei



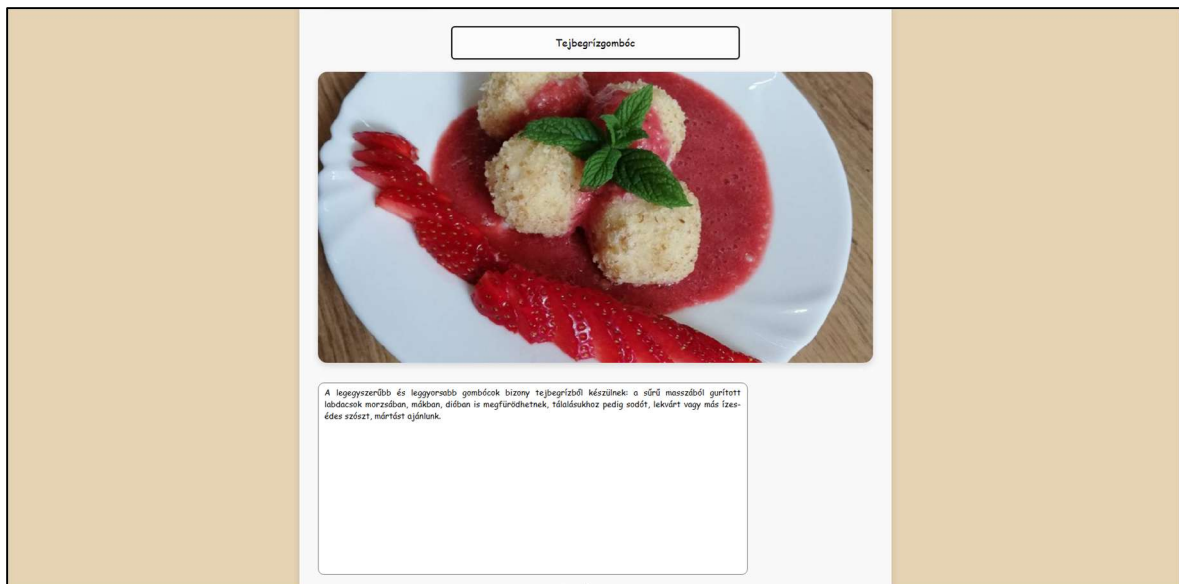
29. kép: Profil oldal




30. kép: Profil beállítások



31. kép: Döntésre váró receptek




32. kép: Döntésre váró receptek részletei




Deszert

Kategória



30











Elkészítési idő



Könnyű

Nehézség

### HOZZÁVALÓK:

 Tej	-	500	ml
 Cukor	-	60	g
 Vaníliásukor	-	1	csomag
 Só	-	1	csipet
 Vaj	-	50	g
 Zsemlemorzsó	-	50	g
 Dió	-	50	g
 Fahéj	-	2	teáskanál
 Málna	-	300	g
 Balzsamecet	-	1	evőkanál

	ELKÉSZÍTÉS:
1.	A tejből, sárból, cukorból és bizadardóid aórdí tejbegrizt főzünk, majd hozzáadjuk a vaj felét és elkeverjük. Egy tálba öntjük, felszínre frissen tartó fóliát nyomunk (hogy ne hűvösödjön), majd hozzájuk tesszük kihűlni.
2.	A diót durvára daroljuk. Egy serpenyőben felolvasztjuk a vaj másik felét, majd beleadjuk a zsemlemorzsát, diót és fahéjat. Közepes lángon aranybarnára pirítjuk az egészet.
3.	A kihűlt tejbegriztből vízzel kézzel gombócokat formázunk. Ha van, hozzáadjuk ehhez fagyaltszedő kakaót. A gombócokat megforgatjuk a diós morzsában, majd a málnaszósszal tállaluk.
4.	A málnát, cukrot és balsemeccet egy kis fazékban összefőzünk nagyjából 5-7 perc alatt, majd ezzel tállaljuk a gombócokat.
	TAGEK
Elfogadás	Elutasítás

## Bejelentkezés

Az oldalunk lehetőséget nyújt bejelentkezésre, illetve regisztrációra is – néhány funkció csak bejelentkezést követően érhető el –. A bejelentkezéshez pedig egy regisztrált E-mail cím és jelszó párosra van szükség. Amennyiben a felhasználó nem rendelkezik fiókkal, abban az esetben regisztrálnia kell egyet.

–KÉP a bejelentkező és regisztráló felületről–

Admin jogú felhasználói fiók bejelentkezési adatai:

E-mail: [admin@admin.hu](mailto:admin@admin.hu)

Jelszó: Admin123!

Átlag felhasználói fiók bejelentkezési adatai:

E-mail: [pelda@pelda.hu](mailto:pelda@pelda.hu)

Jelszó: Pelda123!

Ezen felhasználók egyike sem rendelkezik érzékenységgel, sem pedig ízlés profillal. Az érzékenységek változtatására a profil oldalon van lehetősége.

## 7. A program tesztelése

### 7.1 Backend unit – egységtesztek

A backend projekt tesztelésére unit tesztet alkalmaztunk, amely a rendszer működését és stabilitását biztosítja. A tesztelés célja, hogy az egyes komponensek önálló működését ellenőrizze.

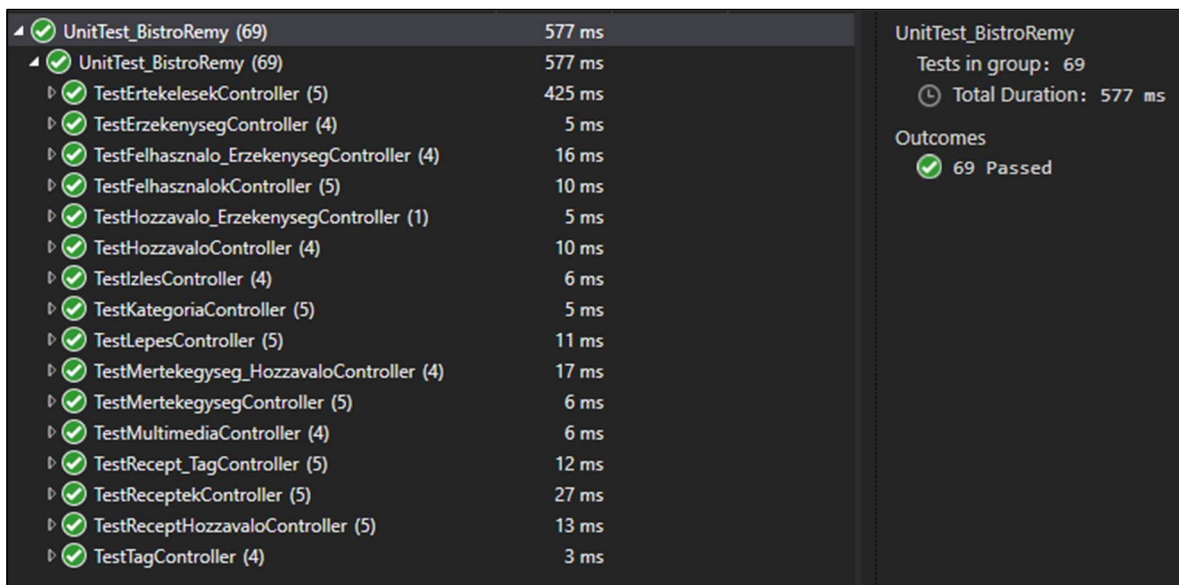
A tesztelés ezen formája nem az éles adatbázison dolgozik, hanem egy úgynevezett mockolt, csak a memóriában létező adatbázist használ. Ez biztosítja azt, hogy tesztelés során, az adatbázisban lévő adatok nem vesznek el, a tesztelés pedig kockázatmentesen folytatható le. A teszt demo adatokat használ, amelyek a projekten belül egy külön file-ban találhatóak. – DemosClass.css – A tesztelési folyamat egyszerűsítése érdekében az ebben a file-ban található metódusok csupán egy paramétert fogadnak, amelyek a demo adat sorszámát várja. Ennek a függvények segítségével sok, egymástól eltérő adat könnyedén előállítható teszt adatbázishoz. Egy ilyen metódus a következőképp néz ki:

```
public Izles IzlesDemo(int instances)
{
    PasswordManager.CreatePasswordHash("Demo", out byte[] hash, out byte[] salt);
    var felhasznalo= new Felhasznalo
    {
        Id = 1 + instances,
        Fnev = "Demo" + instances,
        Email = "DemoEmailt",
        Jelszo_Hash = hash,
        Jelszo_Salt = salt,
        Jogosultsag = 1,
        ProfilkepURL = "Demo.jpg"
    };
    var tag = new Tagek
    {
        Tid = 1 + instances,
        Nev = "Demo" + instances
    };
    return new Izles
    {
        T_id = tag.Tid,
        F_id = felhasznalo.Id
    };
}
```

35. kép : Ízlés demo adatokat előállító függvény

Az egységteszt egy automatizált teszt, amely a Visual Studio-n keresztül indítható el.

Az egyes tesztek, a komponens egyes részeit külön-külön tesztelik. A backend minden metódusára alkalmazunk tesztet, így a projekt összesen 69 tesztet tartalmaz. A tesztek mindegyike az elvártnak megfelelően fut le – Lásd: 36. kép –, ezáltal a metódusok működése is helyes.



36. kép: Unit tesztek sikeres kimenetele



## 7.2 Frontend tesztelés

A weboldal tesztelését a cypress segítségével végeztük el, melyet frontend alkalmazások tesztelésére fejlesztettek ki. A teszteket közvetlenül a böngészőben futtatja, így egy gyors és valós idejű visszajelzést ad.

A tesztelés célja a teljes felhasználói felület működésének ellenőrzése, hogy azok az elvártnak megfelelően és hibamentesen működjenek. A tesztelés az oldal minden komponensére kiterjed és a legtöbb felhasználói interakciót ellenőrzi. Vizsgálja, hogy a különböző komponensek és azok elemei megjelennek-e és ha igen, akkor annak meghatározott helyén jelennek-e meg. Ezenfelül ellenőrzi a különböző navigációs elemeket, hogy azok a megfelelő helyre mutatnak-e.

A tesztelés során a tesztkörnyezet nem valós, hanem előre beállított adatokkal dolgozik. – A tesztek során megjelenő receptek és azok adatai nem valósok, kizárólag a teszt környezetben léteznek. –

## 8. Továbbfejlesztési lehetőségek

A rendszer jelenleg már számos kulcsfontosságú elemet tartalmaz, amik a felhasználók számára fontos lehet egy oldal esetében, mint például a felhasználói fiókok kezelése vagy a receptértékelés. Azonban a jövőben megszeretnénk valósítani további fejlesztéseket a szoftverünkön annak érdekében, hogy még fejlettebb teljesítményt és felhasználóorientáltabb élményt nyújthasson, a továbbiakban pár ilyen lehetőséget említenénk meg.

### 8.1 „Mi van a hűtődben?” bővítési lehetőségei

Szeretnénk az oldalunk „Mi van a hűtődben?” oldalát bővíteni néhány funkcióval, a továbbiakban ezeket fogjuk felsorolni.

#### 8.1.1 Kalóriaszámláló és tápérték-elemző funkció

A rendszerünkbe integrálnánk egy olyan komponenst, amely a recept összetevői alapján automatikusan kiszámolja az étel kalória- és tápanyagtartalmát. Ezt a funkciót nem csak a „Mi van a hűtődben?” kiegészítéseként szeretnénk, hanem esetleg az oldalon való keresés modulba is beépíteni, hogy tényleg mindenki megtalálja a számára legjobb ételt. Az oldalunk azon részén, ahol az összes receptet feltüntetjük ott ezt úgy oldanánk meg hogy minden receptnek számolnánk egy átlag kalóriát, míg a „Mi van a hűtődben?” résznél a felhasználó által felvitt alapanyagok alapján számítanánk ki. Ennek a megvalósítása így nézne ki:

1. A felhasználó felviheti, hogy milyen alapanyagok találhatók otthon.
2. A rendszer ezek alapján receptötleteket kínál, és kiszámolja, hogy ha az adott receptet elkészíti, az hány kalóriát tartalmaz.
3. Külső API – pl.: USDA, Edamam – segítségével valósítanánk meg a pontos tápanyagadatok lekérdezése.

#### 8.1.2 Vonalkód alapú ételismerés

A felhasználói élmény miatt fontosnak tartjuk, hogy olyan funkcióval bővítsük az oldalunkat, ami megkönnyíti az oldal használatát a felhasználók számára. Ezért is gondoljuk úgy, hogy megkönnyítenénk az alapanyag felvitelt, ezzel a fejlesztéssel az előző pontban már bemutatott funkciót tökéletesítenénk. Ez a következőképpen működne:

1. A felhasználó mobilkészülékének kameráját vagy számítógép webkameráját használva beolvashat egy termék csomagolásán található vonalkódot.
2. A rendszer automatikusan felismeri az adott ételismert.
3. Hozzáadja azt a „hűtőm tartalma” listához.
4. Ez felhasználható a bevásárlólista, kalóriaszámláló és receptajánlások során.

## 8.2 További felhasználói funkciók

### 8.2.1 Bevásárlólista generálás

A felhasználó a kiválasztott recept alapján automatikusan létrehozhat egy bevásárlólistát, amely tartalmazza az összes szükséges alapanyagot. Ezt a listát ki is nyomtathatja vagy akár le is töltheti. A későbbiekben, ha lesznek olyan erőforrásaink/partnereink, szeretnénk egy olyan funkciót is beleépíteni, hogy az oldal a bevásárlólista generálásakor a felhasználó számára jelezze, hogy honnan és milyen márkájú alapanyagot ajánljunk neki – ez akkor is hasznos, ha a felhasználó diétázik vagy valamilyen étel érzékenységgel él.

### 8.2.2 Heti menü/étkezési naptár

A felhasználói profilhoz hozzáadnánk egy naptár funkciót ahová a lementett receptjei alapján egy menüt, étkezési naptárat állíthat össze. Így a felhasználó előre megtervezheti, hogy melyik nap milyen ételt szeretne elkészíteni és akár az egész hetes kalóriabevitelét számon tudja tartani.

### 8.2.3 Push értesítések és e-mail rendszer

Fontos célunk az, hogy a felhasználó úgy érezze, hogy egy közösség tagja ezért is szeretnénk megvalósítani, hogy értesítéseket tudjunk küldeni a felhasználók számára, ha igényt tartanak rá. Így a felhasználó értesítést kap, ha valaki értékelte vagy hozzászolt az általa feltöltött recepthez vagy, ha az admin elfogadta vagy elutasította az általa feltölteni kívánt receptet, esetleg új recepteket ajánlanánk a felhasználóknak és emlékeztetnénk őket, hogy tervezze meg a heti menü sorát.

## 8.3 Technikai fejlesztések

### 8.3.1 Többnyelvűség – lokalizáció

A felület fordítható lenne több nyelvre, mint például angol, német vagy francia. Ez lehetővé tenné, hogy nemzetközi közönség is könnyedén használhassa az alkalmazásunkat.

### 8.3.2 Mobilalkalmazás / Offline mód

Egy natív mobilalkalmazás – Android/iOS – segítségével szeretnénk elérni, hogy az adatbázisunk egy része elérhető legyen internetkapcsolat nélkül is. Ezzel megkönnyítenénk a vó-nalkód-beolvasást, illetve a bevásárlás közbeni használatot.

### 8.3.3 Alkalmazás struktúra fejlesztése

- Cache-elés a gyorsabb működéshez.
- Kétlépcsős azonosítás – 2FA.
- Skálázhatóság: nagyobb felhasználószám kezelése felhőalapú infrastruktúrával – mint például: Azure, AWS.

# Tartalom

1.Bevezetés .....	1
1.1 A program célja .....	1
1.2 Felhasználási terület – Célközönség.....	1
2.A szoftver működése .....	2
3. Komponenseinek technikai leírása .....	3
3.1 A backend komponens működése .....	3
3.1.1 Fő funkciója.....	3
3.1.2 Technológiák alkalmazása.....	3
3.1.3 Végpontok működése .....	4
3.2 A frontend komponens működése .....	35
3.2.1 Fő funkciója.....	35
3.2.2 Komponensek részletes leírása és rövid bemutatása .....	35
4. Az adatbázis felépítése .....	44
4.1 Adatbázis ER modell ábra .....	44
4.1.1 Adatbázis adatmodel leírása .....	45
4.1.2 Az adatbázis tábláinak bemutatása .....	45
4.1.2.a Értékelések.....	45
4.1.2.b Érzékenységek .....	46
4.1.2.c Felhasználók .....	46
4.1.2.d Felhasználó_Érzékenység.....	47
4.1.2.e Hozzávalók .....	47
4.1.2.f Hozzávaló_Érzékenységek.....	47
4.1.2.g Ízlések.....	48
4.1.2.h Kategóriák .....	48
4.1.2.i Lépések.....	49
4.1.2.j Mértékegységek.....	49

4.1.2.k Mértékegység_Hozzávaló .....	50
4.1.2.l Multimédiák .....	50
4.1.2.m Receptek .....	51
4.1.2.n Recept_Hozzávaló .....	52
4.1.2.o Recept_Tag.....	52
4.1.2.p Tagek .....	53
5. Működésének műszaki feltételei .....	54
5.1 Kötelezően telepítendő programok.....	54
6. Használatának bemutatása.....	55
Kezdő oldal.....	55
Fejléc .....	55
Regisztráció és bejelentkezés .....	55
Receptek feltöltése.....	55
Mi van a hűtődben? .....	56
Összes recept .....	56
Receptek részletei .....	57
Profil oldal .....	57
Döntésre váró receptek .....	57
Döntésre váró receptek részletei.....	57
Monitorképek .....	59
Bejelentkezés .....	69
7. A program tesztelése .....	70
7.1 Backend unit – egységtesztek.....	70
7.2 Frontend tesztelés .....	72
8. Továbbfejlesztési lehetőségek .....	73
8.1 „Mi van a hűtődben?” bővítési lehetőségei .....	73
8.1.1 Kalóriaszámláló és tápérték-elemző funkció.....	73

8.1.2 Vonalkód alapú élelmiszer-felismerés .....	73
8.2 További felhasználói funkciók .....	74
8.2.1 Bevásárlólista generálás .....	74
8.2.2 Heti menü/étkezési naptár .....	74
8.2.3 Push értesítések és e-mail rendszer .....	74
8.3 Technikai fejlesztések.....	74
8.3.1 Többnyelvűség – lokalizáció .....	74
8.3.2 Mobilalkalmazás / Offline mód.....	74
8.3.3 Alkalmazás struktúra fejlesztése .....	74
Források.....	77

## Források

Az adatbázisban szereplő minden recept és annak képe a <https://www.nosalty.hu/kereses/recept> magyar receptes oldalról származik.

Az oldalon használt ikonok a <https://www.freepik.com/> illetve, a <https://www.flaticon.com/icons> oldalakról származnak.

Az oldalunk logójául szolgáló kép a [https://iconscout.com/free-icon/cute-mouse-bring-spoon-9633873\\_7822369](https://iconscout.com/free-icon/cute-mouse-bring-spoon-9633873_7822369) oldalon található.

Az oldal fejlécében, illetve a bejelentkezési felületen található kép a <https://wallpaperaccess.com/cozy-cafe> oldalon elérhető

Az oldalon a receptek alapértelmezett képe a <https://dribbble.com/whomohit> oldalon található.