Technical Specification Document

1. Introduction

- **Project Title**: Smart Meal Finder
- **Document Version**: 5.1
- **Date**: 2025.12.16.
- **Prepared by**: Kuti Zoltán

2. Version history

#	Ver.	Date	Author	Description
00	0.0	2024.07.08.	T. Storcz	
01	1.0	2024.11.13.	Z. Kuti	Started template fill
02	2.0	2024.11.14.	Z. Kuti	Added System Architecture description and Data Flow Diagram
03	3.0	2024.11.17.	Z. Kuti	Added Non-Functional Requirements description and Technology Stack descpription
04	4.0	2024.11.18.	Z. Kuti	Filled Interface Specification
05	5.0	2025.09.16.	Z. Kuti	Updated System Architecture, Data Design, Interface Specifications (External,Internal)
06	5.1	2025.12.16.	Z. Kuti	Updated System Architecture, Data Design, Interface Specifications (Internal), Technology Stack

3. Purpose

- **Purpose Statement**: Briefly describe the purpose of the document and what it aims to achieve.
    - A dokumentum célja a Smart Meal Finder project technikai felépítésének meghatározása és ismertetése.
- **Scope**: Define the boundaries of the technical specifications covered in this document.
    - **Included**: Adatbázis, Webalkalmazás

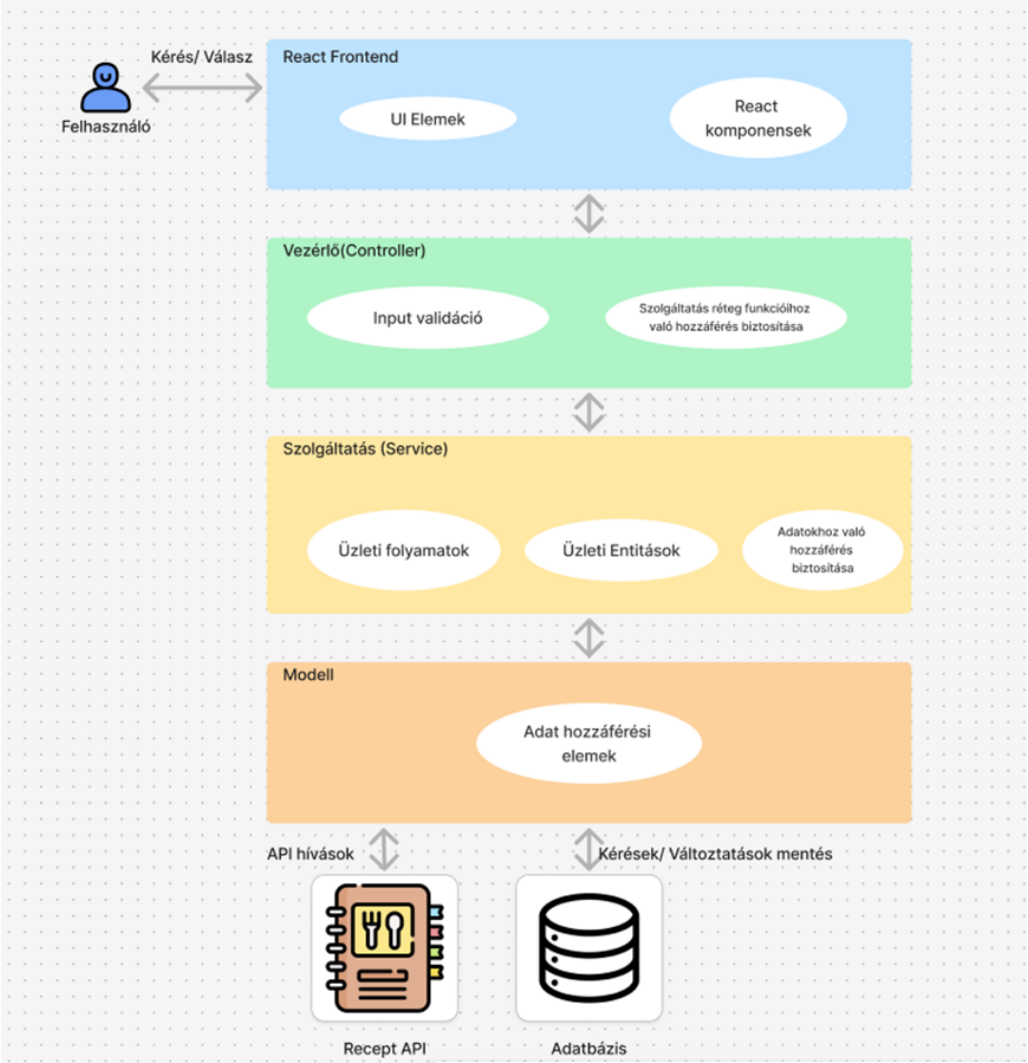
4. System Overview

- **System Description**: High-level description of the system and its objectives.
    - Segítse a felhasználókat az ételreceptek felfedezésében az összetevőik és a konkrét étrendi céljaik alapján.
- **Architecture Overview**: A summary of the system architecture, including major components and their interactions.
    - A Webalkalmazás egy kliens-szerver architektúrára épül. Maga az alkalmazás Model-View-Controller-Service architektúrát használ.
        - Model: Feladata az alkalmazás adatainak kezelése, a kommunikáció kezelése az adatbázis és az alkalmazás között, valamint a Service réteg számára adatokat szolgáltat

- Service (Szolgáltatás): Itt található az üzleti logika. A Vezérlő rétegtől jövő kéréseket végzi el a Model rétegtől kapott adatokon.
- Controller: Egy közvetítő szerepkört tölt be a Service és a View réteg között. Kezeli a Viewtól érkező felhasználó kéréseket és frissítéseket küld a View rétegnek, amelyek a Model réteg változásait reprezentálják. Itt végezhető például az input validáció.
- View: Megjeleníti az adatokat, valamint felhasználói inputokat küld a Controller rétegnek. Közvetlenül nem kommunikál a Model réteggel.

5. System Architecture

- **High-Level Architecture Diagram**: Visual representation of the system architecture.

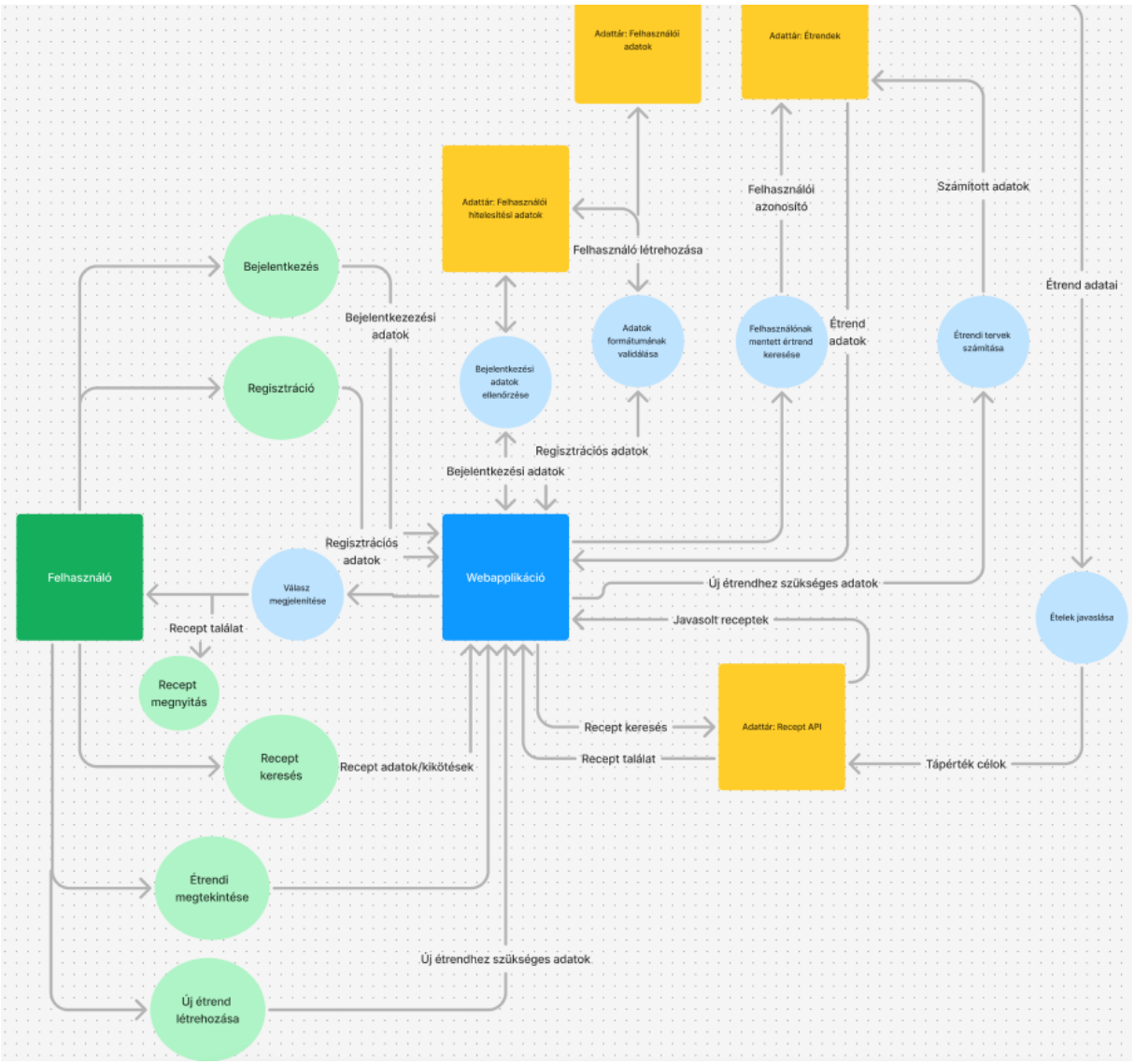


Component Descriptions: Detailed descriptions of each major component.

- **Nézet (View) réteg:** Feladata az adatok felhasználóbarát megjelenítése és felület biztosítása a kommunikációhoz az alkalmazással
  - **UI Elemek:** Ezek olyan vezérlők, amelyek segítik a kommunikációt a Felhasználóval. A felhasználó ezeken keresztül vihet be kéréseket. Ide tartoznak azok az elemek is, amelyek a kérésekre érkező válaszokat jelenítik meg.
  - **React komponensek:** A React segít a felhasználói felület kialakításában.
- **Vezérlő (Contoller) réteg:** Feladata a kapott adatokat a Szolgáltatás és a Nézet réteg számára megfelelő módra alakítani és hozzáférést biztosítani a Szolgáltatás réteg béli üzleti folyamatokhoz.
  - **Input validáció:** A felhasználói bevitelből eredő adatok formátumát ellenőrzi, és csak akkor továbbít kérést a Szolgáltatás rétegnek, ha azok megfelelőek.
  - **Szolgáltatás réteg funkcióihoz való hozzáférés biztosítása:** Feladata a rétegbe érkező kérések továbbítása a Szolgáltatási réteg felé.
- **Szolgáltatás (Service) réteg:** Ez a réteg az alkalmazás lelke. Itt találhatóak az alkalmazás funkciói.
  - **Üzleti folyamatok:** Ezek a folyamatok, amelyek a program funkcionalitását adják. Minden változtatás, keresés, szűrés, számítás itt hajtódik végre.
  - **Üzleti entitások:** Ezek azok az elemek, amelyeken a műveletek elvégződnek (pl.: User). Az itt elvégzett változtatásokat továbbküldjük a Model rétegnek, ami az adatbázissal kommunikál.
- **Modell réteg**: Feladata az adatforrások elérésének biztosítása.
  - **Adat hozzáférési elem**: Az adatbázisban található adatokhoz biztosít elérést. Feladatai közé tartozik a leképezés, kapcsolat kezelés, konkurens adatelérés kezelése.

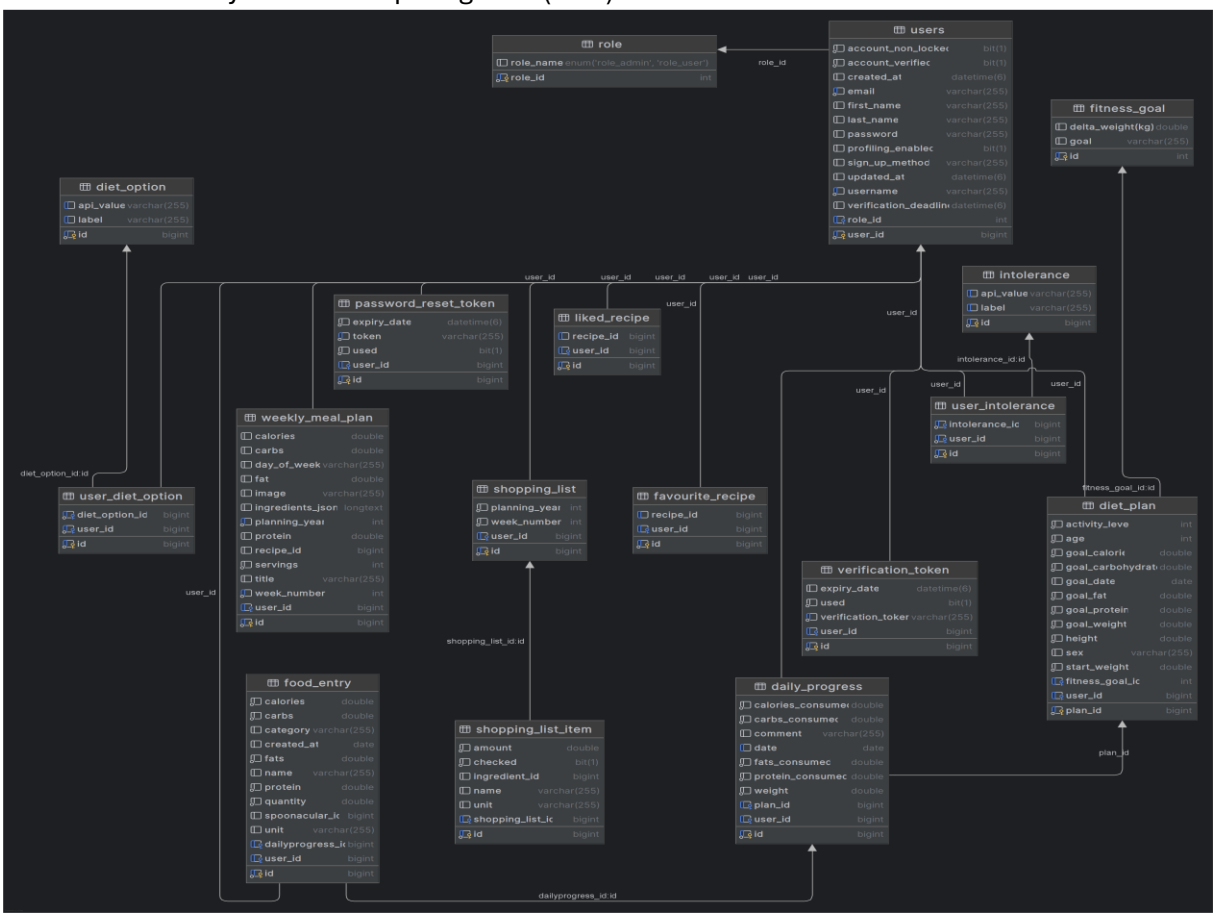
- **Component integration:** Specify the interfaces and communication protocols used for integration of components.
    - **Frontend és Backend:** A kommunikáció a Frontend és az alkalmazás között HTTP protokollon keresztül történik.
      Néhány végpont:
        - /: A weblap kezdőoldalára visz.
        - /login: A bejelentkezési oldalt tölti be. Helyes adatok megadása esetén a felhasználó adatait betölti az applikációba.
        - /register: A regisztráló felületet tölti be. Itt kitölthető egy form, ami alapján az alkalmazás elmenti a megfelelő adatokat az adatbázisba és létrehoz egy profilt. Sikeres regisztráció esetén a bejelentkezési oldalra irányít.
        - /plan: Az étrendtervező oldalra visz minket. Ha nincs még adat mentve, akkor létrehozhatunk egy tervet. Ha van már terv, akkor az aktuális céljainkat és a javasolt recepteket láthatjuk.
        - /profile: A bejelentkezett felhasználó adatait jeleníti meg, valamint módosításra ad lehetőséget.
        - /whats-in-my-fridge: Hzozzátevőalapú receptkeresést tesz lehetővé
        - /admin/users: Ha a felhasználónak van admin joga, akkor megjeleníti az összes regisztrált felhasználót egy táblában.
        - /admin/users/{id}: Magának az adiott felhasználónak az adatait módosíthatjuk, ha admin jogunk van.
    - **Backend és Adatbázis**: Az alkalmazás és az adatbázis közötti kommunikációra Spring JPA-t használunk. Ez az API lehetővé teszi Objektum-relációs leképezést (Object Relationship Mapping, ORM), ami az alkalmazás által használt objektumokat képezi le relációs adatbázisbeli entitásokra, illetve az elvégzett műveleteket fordítja SQL kódra, valamint hozzáférést biztosít az adatbázisban található funkciókhoz, procedure-ökhoz, nézetekhez.
    - **Hibakezelés**: A hibákat a felhasználó részére hibaüzenetben jelezzük. Ez lehet HTTP státuszkóddal (pl.: 404 – Not Found, 500 – Internal Server Error), vagy hibaüzenettel, amit a felhasználónak megjelenítünk. A hibákat egy adatbázis táblában log-oljuk.

- **Data Flow**: Description of how data will flow through the system.



6. Data Design

- **Data Model**: Entity-Relationship Diagrams (ERD) or other data models.



- **Database Schema**: Description of the database schema, including tables, fields, and relationships.
    - Itt található az Adatbázis szerkezete. Az ábrán nincs megjelenítve, de az összes tábla tartalmaz created at/by, illetve modified at/by mezőket.
    - **users:** A felhasználónak az adatait tartalmazza
        - user_id: A felhasználó egyedi azonosítója, típusa: number, elsődleges kulcs
        - first_name: A felhasználó keresztneve, típusa: varchar2
        - last_name: A felhasználó vezetékneve, típusa: varchar2
        - username: A felhasználót azonosító név, típusa: varchar2
        - password: A bejelentkezéshez használható titkosított jelszót tartalmazza, típusa varchar2
        - email: A felhasználó email-címét tartalamazza, amit a regisztrációkor megad, típusa: varchar2 Megkötés, hogy tartalmaznia kell @ karaktert, illetve a „@” és a „.” karakter nem lehet egymás mellet, és egyik karakter sem lehet a cím végén

- account_non_locked: A fiók zároltságát jelöli, típusa: bit
- account_verified: A fiókhoz tartozó email ellenőrzöttségét jelöli, típusa: bit
- verification_deadline: Az email ellenőrzésének határidejét tartalmazza, típusa: datetime
- role_id: A fiók jogkörét azonosítja, idegen kulcs
- created_at: A fiók létrehozásának dátumát tárolja, típusa: datetime
- updated_at: A fiók utolsó módosításának dátumát tárolja, típusa: datetime
- sign_up_method: A fiók regisztrációjának módját tartalmazza (pl.: email, google)

- **verification_token**: Az email ellenőrzéséhez szükséges tokent tárolja
  - id: A token egyedi azonosítója, elsődleges kulcs, típusa: number
  - expiry_date: A token lejárati dátuma, a felhasználó regisztrációja után 15 nap, típusa: datetime
  - verification_token: univerzálisan egyedi azonosító (UUID), ami az emailben elküldésre kerül
  - used: A token elhasználásának állapotát tárolja, típusa: bit
  - user_id: A token tulajdonosát tartalmazza, az egyetlen felhasználó, aki a tokent tudja használni, idegen kulcs
- **password_reset_token**: Elfelejtett jelszó visszaállításához szükséges tokent tárolja
  - id: A token egyedi azonosítója, elsődleges kulcs, típusa: number
  - expiry_date: A token lejárati dátuma, a felhasználó kérvényezése után 15 perc, típusa: datetime
  - token: univerzálisan egyedi azonosító (UUID), ami az emailben elküldésre kerül
  - used: A token elhasználásának állapotát tárolja, típusa: bit
  - user_id: A token tulajdonosát tartalmazza, az egyetlen felhasználó, aki a tokent tudja használni, idegen kulcs
- **role**: Az alkalmazásban megtalálható jogköröket tartalmazza
  - role_id: A jog egyedi azonosítója, elsődleges kulcs, típusa: number
  - role_name: A jogkör neve, típusa: varchar
- **diet_option:** Az alkalmazásban található étkezési diétákat tartalmazza
  - id: A diéta egyedi azonosítója, típusa: number, elsődleges kulcs

        - label: A korlátozás megnevezése, típusa: varchar2
        - api_value: A Spoonacular API által elfogadott értékeket tartalmazza, típusa: varchar2
    - **user_diet_option:** A több-több kapcsolat feloldásaképp létrejött kapcsolótábla
        - id: A kapcsolat egyedi azonosítója, elsődleges kulcs
        - user_id: A felhasználó egyedi azonosítóját hivatkozza a users táblából, típusa: number, idegen kulcs
        - diet_option_id: A diéta egyedi azonosítóját hivatkozza a diet_option táblából, típusa: number, idegen kulcs
    - **fitness_goal:** A választható fitnesz célokat tartalmazza
        - id: A fitneszcél egyedi azonosítója, típusa: number, elsődleges kulcs
        - goal: A fitnesz cél megnevezése, típusa: varchar2
        - delta_weight(kg): A cél eléréséig szükséges heti súlyváltozást tartalmazzam, típusa: float

- **diet_plan:** A felhasználók által generált étkezési terveket tartalmazó tábla
  - plan_id: Az étkezési terv egyedi azonosító száma, típusa: number, elsődleges kulcs
  - activity_level: A célszámok számítása miatt szükséges szám, típusa: number
  - age: A felhasználó kora, típusa: number
  - start_weight: A felhasználó kiindulósúlya, típusa: float
  - goal_weight: A felhasználó által kijelölt célsúlyt tartalmazza, típusa: float
  - goal_calorie: Számolt érték, ami a felhasználó számára javasolt fogyasztandó kalória mennyiségét tartalmazza, típusa: number
  - goal_protein: Számolt érték, ami a felhasználó számára javasolt fogyasztandó protein mennyiséget tartalmazza, típusa: number
  - goal_carbohydrate: Számolt érték, ami a felhasználó számára javasolt fogyasztandó szénhidrát mennyiségét tartalmazza, típusa: number
  - goal_fat: Számolt érték, ami a felhasználó számára javasolt fogyasztandó zsírok mennyiségét tartalmazza, típusa: number
  - goal_date: Számolt érték, a terv generálásának dátuma, a célsúly, a kiindulósúly és a heti súlyváltozás értéke alapján számolódik ki, típusa: date
  - height: A felhasználó magassága, típusa: float
  - sex: A felhasználó neme, típusa: varchar
  - fitness_goal_id: A fitness célt tartalmazza, amihez a felhasználó generáltatta a tervet, a Fitness_Goal Fitness_ID mezőjét hivatkozza, típusa: number, idegen kulcs
  - user_id: A felhasználó egyedi azonosítóját hivatkozza a users táblából, típusa: number, idegen kulcs

- **daily_progress:** A felhasználó napi teljesítményét tartalmazza
  - id: A napi adatok egyedi azonosítója, elsődleges kulcs, típusa: number
  - calories_consumed: A felhasználó által bevitt kalória napi szummája, típusa: float
  - carbs_consumed: A felhasználó által bevitt szénhidrát napi szummája, típusa: float
  - comment: A felhasználó adott napra tett megjegyzése, típusa: varchar
  - date: Az adott napi teljesítmény dátuma, típusa: date
  - fats_consumed: A felhasználó által bevitt zsír napi szummája, típusa: float
  - protein_consumed: A felhasználó által bevitt protein napi szummája, típusa: float
  - weight: A felhasználó által megadott aznap mért súly, típusa: float
  - plan_id: Azt jelzi, hogy melyik tervhez tartozik a napi bejegyzés, idegen kulcs
  - user_id: Azt jelzi, hogy melyik felhasználóhoz tartozik a napi bejegyzés, idegen kulcs
- **food_entry**: A felhasználó által elfogyasztott ételt tartalmazza
  - id: A bejegyzés egyedi azonosítója, elsődleges kulcs, típusa: number
  - calories: Az elfogyasztott étel kalóriaértéke, típusa: float
  - carbs: Az elfogyasztott étel szénhidrátértéke, típusa: float
  - fats: Az elfogyasztott étel zsírértéke, típusa: float
  - name: Az elfogyasztott étel neve, típusa: varchar
  - protein: Az elfogyasztott étel fehérjeértéke, típusa: varchar
  - spoonacular_id: Az elfogyasztott étel Spoonacular API-ban tárolt ID-ja
  - dailyprogress_id: Azt jelzi, hogy az elfogyasztott étel melyik DailyProgress-hez tartozik, idegen kulcs
  - created_at: Az elfogyasztás rögzítésének dátuma. típusa: date
  - user_id: Azt tárolja, melyik felhasználó rögzítette fel a fogyasztást, idegen kulcs
- **favourite_recipe:** A felhasználó által megjelölt kedvenc receptek id-ját tartalmazza
  - id: A kapcsolat egyedi azonosítója, elsődleges kulcs, típusa: number
  - recipe_id: A Spoonacular recept idja, külső id, típusa: number

        - user_id: A felhasználó azonosítója, idegen kulcs
    - **liked_recipe**: A felhasználó által kedvelt recepteket tartalmazza
        - id: A kapcsolat egyedi azonosítója, elsődleges kulcs, típusa: number
        - recipe_id: A Spoonacular recept idja, külső id, típusa: number
        - user_id: A felhasználó azonosítója, idegen kulcs
    - intolerance: Az alkalmazás által kezelt intoleranciákat tartalmazza
        - id: Az intolerancia egyedi azonosítója, elsődleges kulcs, típusa: number
        - api_value: A Spoonacular API számára értelmezhető érték, amit fel tudunk használni szűrésekhez, típusa: number
        - label: Az intolerancia emberközeli, olvasható formátumú elnevezését tárolja, típusa: varchar
    - user_intolerance: A felhasználó és az intoleranciák közötti kapcsolótábla.
        - id: A kapcsolat egyedi azonosítója, elsődleges kulcs, típusa: number
        - intolerance_id: Az intolerancia egyedi azonosítóját tartalmazza, idegen kulcs
        - user_id: A felhasználó egyedi azonosítóját tartalmazza, idegen kulcs
    - weekly_meal_plan: A Felhasználó által generált heti étkezési tervet és a hozzá kapcsolódó adatokat tartalmazza
        - id: Egyedi azonosító, ami azonosítja a felhasználó, év, hét, nap, recept kapcsolatot, elsődleges kulcs, típusa: number
        - user_id: A felhasználó egyedi azonosítója, aki készítette a tervet, idegen kulcs
        - planning_year: A terv érvényességének éve, típusa: number
        - week_number: A terv érvényességének hetének száma, típusa: number
        - day_of_week: Azt a napot azonosítja, amelyik naphoz tartozik a recept, típusa: varchar
        - recipe_id: A Spoonacular recept idja, külső id, típusa: number
        - title: A Spoonacular recept címe, típusa: varchar
        - image: A Spoonacular recept képének elérhetősége, típusa: varchar
        - calories: A Spoonacular recept kalóriatartalma 1 tálalásra, típusa: float
        - protein: A Spoonacular recept proteintartalma 1 tálalásra, típusa: float
        - carbs: A Spoonacular recept szénhidráttartalma 1 tálalásra, típusa: float
        - fat: A Spoonacular recept zsírtartalma 1 tálalásra, típusa: float
        - servings: A Spoonacular recept tálalásának mennyisége, típusa: number

        - ingredients_json: A recept hozzávalóit tartalmazza 1 tálalásra, típusa: longtext
    - **shopping_list**: Az adott terv alapján kigenerált bevásárlólistát tartalmazza
        - id: A lista egyedi azonosítója, elsődleges kulcs, típusa: number
        - planning_year: Az az év, amihez a lista tartozik, típusa: number
        - week_number: Annak a hétnek a száma, amihez a bevásárlólista tartozik, típusa: number
        - user_id: A felhasználó egyedi azonosítója, aki készítette a tervet, idegen kulcs
    - **shopping_list_item:** Az adott bevásárlólistához tartozó elemeket tartalmazza
        - id: Az elem egyedi azonosítója, elsődleges kulcs, típusa: number
        - amount: A hozzávaló mennyiségét tarolja el, típusa: float
        - checked: Azt tárolja, hogy az adott hozzávaló meglett-e már véve, típusa: bit
        - ingredient_id: A Spoonacular hozzávaló id-ja, külső id, típusa: number
        - name: A hozzávaló neve, típusa: varchar
        - unit: Annak a típusa, hogy miben mérhető a hozzávaló (WEIGHT, VOLUME, COUNT), típusa: varchar
        - shopping_list_id: A bevásárlólista egyedi azonosítója, amihez az adott elem tartozik, idegen kulcs

7. Interface Specifications

- **User Interface (UI)**: Wireframes, mockups, and descriptions of the user interface components.
    - https://www.figma.com/board/QwZUZ9plps4HhOKsez3JTW/Wireframe?node-id=0-1&node-type=canvas&t=aM0GCTmz8mvOHdn0-0
- **Application Programming Interface (API)**: Detailed descriptions of APIs, including endpoints, request/response formats, and authentication methods.
    - Az alkalmazás RESTful API-t használ, amely lehetővé teszi az adatok lekérdezését, módosítását és törlését az alkalmazás frontend és backend közötti kommunikáció során. Az API JSON formátumban fogadja és küldi az adatokat.
    - **Néhány végpont:** Itt a végpontok egy része található. Az összes még nem, de fejlesztés közben frissítésre fog kerülni amikor egy új végpont létrejön.

Funkció	URL	HTTP metódus	Paraméterek	Leírás
Felhasználók lekérése	/api/admin/getusers	GET	Header: Authorization: Bearer „”;	Minden felhasználó listázása
Felhasználó szerepkörének frissítése	/api/admin/update-role	PUT	Header: Authorization: Bearer „”; Body: userId, role	Megváltoztatja a felhasználó szerepkörét
Felhasználó lekérése ID alapján	/api/admin/user/{id}	GET	Header: Authorization: Bearer „”; Path: id	Egy konkrét felhasználó lekérdezése
Szerepkörök lekérése	/api/admin/roles	GET	Header: Authorization: Bearer „”;	Az összes szerepkör visszaadása
Felhasználó zárolási státuszának frissítése	/api/admin/update-lock-status	PUT	Header: Authorization: Bearer „”; Body: userId, checked (boolean)	Felhasználó fiók zárolásának/engedélyezésének beállítása
Felhasználó ellenőrzési státuszának frissítése	/api/admin/update-verification-status	PUT	Header: Authorization: Bearer „”; Body: userId, checked (boolean)	Felhasználó ellenőrzött státuszának módosítása

Felhasználó jelszavának frissítése	/api/admin/update-password	PUT	Header: Authorization: Bearer „”; Body: userId, password	Felhasználó új jelszavának beállítása
Felhasználó ajánlórendszerenének engedély státusz módosítása	/api/admin/update-profiling-status	PUT	Header: Authorization: Bearer „”; Body: userId, checked (boolean)	Felhasználói ajánlás engedélyezése vagy tiltása

CSRF token lekérése	/api/csrf-token	GET	-	CSRF tokent lehet megszerezni a különböző adatbázis módosító API hívások ellenőrzéséhez
Testsúly naplózása	/api/progress/weight/save	POST	Header: Authorization: Bearer „”; Body: weight, comment	Elmenti a felhasználó aktuális testsúlyát és opcionális megjegyzését

Napi előrehaladás lekérése	/api/progress	GET	Header: Authorization: Bearer „”;	Lekéri az adott napi előrehaladási adatokat (testsúly, jegyzet)
Előrehaladási előzmények lekérése	/api/progress/all	GET	Header: Authorization: Bearer „”;	Visszaadja a felhasználó összes eddigi előrehaladási naplóját
Összes diéta lekérése	/api/diet-option/list	GET	-	Visszaadja az összes elérhető diétát
Felhasználó diétáinak lekérése	/api/diet-option/load-by-user	GET	Header: Authorization: Bearer „”	Visszaadja a bejelentkezett felhasználóhoz tartozó diétákat
Felhasználó diétáinak módosítása	/api/diet-option/add-to-user	POST	Header: Authorization: Bearer „”; Body: diets (lista)	Módosítja a felhasználó diétáit a bejövő lista és a tárolt diéták alapján (TODO)
Étkezési terv létrehozása	/api/plan/create	POST	Header: Authorization: Bearer „”; Body: sex, weight, height, age, activityLevel, goalType, weightGoal	Új étkezési terv és célkalkuláció létrehozása a felhasználó adatai alapján

Étkezési terv lekérése	/api/plan	GET	Header: Authorization: Bearer „”;	Visszaadja a felhasználóhoz tartozó aktuális étkezési tervet
Étkezési terv törlése	/api/plan	DELETE	Header: Authorization: Bearer „”;	Törli a felhasználó étkezési tervét és az ahhoz kapcsolódó étkezési naplókat és a napi előrehaladásokat
Fitneszcélok lekérése	/api/fitness-goal/all	GET	Header: Authorization: Bearer „”;	Visszaadja az összes elérhető fitneszcélt
Étkezés naplózása	/api/food-entry/save	POST	Header: Authorization: Bearer „”; Body: spoonacularId, name, calories, protein, carbs, fats felhasználó	Elmenti a felhasználó új étkezési bejegyzését
Mai étkezések lekérése	/api/food-entry/list	GET	Header: Authorization: Bearer „”;	Visszaadja a felhasználó mai napra vonatkozó összes étkezési bejegyzését
Étkezés törlése	/api/food-entry/delete/{foodEntryId}	DELETE	Header: Authorization: Bearer „”; Path: foodEntryId	Törli a megadott azonosítójú étkezési bejegyzést
Felhasználó regisztrációja	/api/auth/public/register	POST	Body: email, username, password, role, fistName, lastName	Új felhasználó regisztrálása és ellenőrző token generálása
Felhasználó bejelentkezése	/api/auth/public/login	POST	Body: username, password	Bejelentkezés és JWT token generálása
Profil lekérése	/api/auth/user	GET	Header: Authorization: Bearer „”;	Lekéri a bejelentkezett felhasználó adatait
Elfelejtett jelszó kérése	/api/auth/public/forgot-password	POST	Query param: email (string)	Jelszó visszaállító

				token generálása és e-mail küldése
Jelszó visszaállítása	/api/auth/public/reset-password	POST	Body: token, newPassword	Jelszó visszaállítása a token felhasználásával
Jelszó módosítása	/api/auth/change-password	POST	Body: token, oldPassword, newPassword	Jelszó módosítása a felhasználó kérésére
Felhasználó email fiók ellenőrzése	/api/auth/public/verify-user	POST	Query param: token (string)	Felhasználó e-mail címének ellenőrzése
Felhasználói ajánlás engedélyezése/ tiltása	/api/auth/user/update-profiling-status	PUT	Header: Authorization: Bearer „”; Body: checked (boolean)	Felhasználói ajánlás engedélyezése vagy tiltása
Felhasználói ajánlás engedélyezése	/api/auth/user/enable-profiling	PUT	Header: Authorization: Bearer „”;	Felhasználói ajánlás engedélyezése
Összes intolerancia lekérdése	/api/intolerance/list	GET	-	Adatbázisban elérhető intoleranciák lekérése
Felhasználó intoleranciáinak lekérdezése	/api/intolerance/load-by-user	GET	Header: Authorization: Bearer „”;	Bejelentkezett felhasználó intoleranciáinak megszerzése
Intolerancia mentése felhasználóhoz	/api/intolerance/save-to-user	POST	Header: Authorization: Bearer „”; Body: intolerances (list)	Intolerancia hozzárendelése felhasználóhoz
Recept kedvelése	/api/recipe/like	POST	Header: Authorization: Bearer „”; Body: SpoonacularRecipe recipe (lásd lejjebb)	Kedvelés hozzáadása a recepthez

Recept kedvelésének visszavonása	/api/recipe/unlike	POST	Header: Authorization: Bearer „”; Body: SpoonacularRecipe recipe (lásd lejjebb)	Kedvelés eltávolítása a recepttől
Recept kedvelésének ellenőrzése	/api/recipe/isLiked	GET	Header: Authorization: Bearer „”; Body: id	Bejelentkezett felhasználó és recept kedvelési státuszának ellenőrzése
Recept kedvelésének száma	/api/recipe/likeCount	GET	Body: id	A recept kedvelésszámának ellenőrzése
Recept kedvenchez adása	/api/recipe/favourite/add	POST	Header: Authorization: Bearer „”; Body: SpoonacularRecipe recipe (lásd lejjebb)	Kedvencnek jelölt recept a felhasználóhoz rendelése
Recept kedvencek közül eltávolítása	/api/recipe/favourite/remove	POST	Header: Authorization: Bearer „”; Body: SpoonacularRecipe recipe (lásd lejjebb)	Kedvencnek jelölt recept eltávolítása a felhasználótól
Kedvenc receptek lekérdezése	/api/recipe/favourite	GET	Header: Authorization: Bearer „”;	Felhasználó kedvenc receptjeinek lekérdezése
Recept kedvenc státusz ellenőrzése	/api/recipe/isFavourite	GET	Header: Authorization: Bearer „”; Body: id	Bejelentkezett felhasználó és recept kedvenc státuszának ellenőrzése
Recept megtekintés elküldése	/api/recipe/seen	POST	Header: Authorization: Bearer „”; Body: SpoonacularRecipe recipe (lásd lejjebb)	Ha a felhasználó engedélyezte az ajánlásokat, „seen” interakció küldése a Gorse rendszernek

Recept elolvasás elküldése	/api/recipe/read	POST	Header: Authorization: Bearer „”; Body: SpoonacularRecipe recipe (lásd lejjebb)	Ha a felhasználó engedélyezte az ajánlásokat, „read” interakció küldése a Gorse rendszernek
Ajánlás kérése	/api/recipe/recommendations	GET	Header: Authorization: Bearer „”;	Ha a felhasználó engedélyezte az ajánlásokat, ajánlások kérése
Napi visszamaradott célértékek lekérdezése	/api/recommendation/remaining-macros	GET	Header: Authorization: Bearer „”;	Bejelentkezett felhasználó visszamaradott napi céltápanyagértékeinek lekérdezése
Heti terv mentése	/api/weekly-planner/save	POST	Header: Authorization: Bearer „”; Body: year, weekNumber, plan (list), shoppingList (list)	Bejelentkezett felhasználó heti étkezési tervének mentése
Heti terv lekérdezése	/api/weekly-planner/load	GET	Header: Authorization: Bearer „”; Body: year, week	Bejelentkezett felhasználó heti étkezési tervének lekérdezése
Bevásárló lista lekérdezése	/api/weekly-planner/shopping-list	GET	Header: Authorization: Bearer „”;	Bejelentkezett felhasználó heti bevásárlólistájának lekérdezése
Bevásárló lista egy elemének „megvásárolt” állítása	/api/weekly-planner/shopping-list/item/toggle	PUT	Header: Authorization: Bearer „”; Body: itemId, checked (boolean)	Bejelentkezett felhasználó heti bevásárlólistájának egy elemének „megvásárolt” állapotra állítása

- **SpoonacularRecipe**: A DTO feépítése:
  - id (Long)
  - title (String)
  - image (String)
  - readyInMinutes (int)
  - servings (int)
  - healthScore (double)
  - vegeterian (boolean)
  - vegan (boolean)
  - glutenFree (boolean)
  - dairyFree (boolean)
  - veryHealthy (boolean)
  - cheap (boolean)
  - veryPopular (boolean)
  - sustainable (boolean)
  - lowFodmap (boolean)
  - dishTypes (List<String>)
  - diets (List<String>)
  - cuisines (List<String>)
  - occasions (List<String>)
  - extendedIngredients (List<String>)
  - nutrition
  - instructions (String)
- **Hibakezelés és Válaszkódok:** Az API válaszokban használt státuszkódok és hibák:
  - 200 OK – Sikeres válasz.
  - 400 Bad Request – Hibás kérés (pl. érvénytelen paraméterek).
  - 401 Unauthorized – Hitelesítés szükséges (pl. hiányzó vagy érvénytelen token).
  - 403 Forbidden – A felhasználó nem rendelkezik a szükséges jogosultságokkal.
  - 404 Not Found – Kért erőforrás nem található.
  - 500 Internal Server Error – Szerverhiba.

- **External Interfaces**: Specifications for any external interfaces or integrations.
    - **Spoonacular API**: Az alkalmazáson belüli ételeket és tápanyag adatokat az Spoonacular külső API fogja szolgáltatni. A kommunikációhoz REST API-t használ HTTPS-en keresztül. A kapott válaszok JSON formátumban érkeznek meg. Bővebb dokumentáció megtalálható itt: https://spoonacular.com/food-api/docs
        - **Recipe Search**
            - HTTP metódus: GET
            - Hozzáférési pont: https://api.spoonacular.com/recipes/complexSearch
            - Paraméterei: A * karakterrel jelölt paraméterek kötelezők
                - apiKey*: A Spoonacular API kulcsa, amely regisztráció után érhető el.
                - query*: A keresett étel neve vagy kulcsszava.
                - cuisine: Konyha típusa (pl. italian, mexican, chinese).
                - diet: Diéta típusa (pl. vegetarian, vegan, gluten free).
                - intolerances: Ételintoleranciák, vesszővel elválasztva (pl. dairy, gluten).
                - excludeIngredients: Hozzávalók, amelyeket ki szeretnél zárni a keresésből (pl. onion, garlic).
                - includeIngredients: Hozzávalók, amelyeket szeretnél, hogy szerepeljenek a receptben (pl. chicken, rice).
                - type: Recept típusa (pl. main course, dessert, snack).
                - number: A visszaadott találatok száma (alapértelmezett: 10, maximum: 100).
                - offset: A találatok kezdőpontja (hasznos lapozásnál).
                - addRecipeInformation: Ha true, részletes információkat ad vissza a receptekről (pl. elkészítési idő, kalóriák).
                - fillIngredients: Ha true, a válaszban szereplő összes hozzávaló mennyiségét is feltünteti.
                - imageSize: A recepthez tartozó képek mérete (pl. small, medium, large).
                - sort: Rendezési sorrend (pl. calories, popularity, time).
                - sortDirection: Rendezés iránya (asc vagy desc).

        - maxReadyTime: A recept elkészítési idejének maximális hossza percben.
        - minCalories: A recept minimális kalóriatartalma.
        - maxCalories: A recept maximális kalóriatartalma.
        - minProtein: A recept minimális fehérjetartalma (grammban).
        - maxProtein: A recept maximális fehérjetartalma (grammban).
        - minFat: A recept minimális zsírtartalma (grammban).
        - maxFat: A recept maximális zsírtartalma (grammban).
        - minCarbs: A recept minimális szénhidráttartalma (grammban).
        - maxCarbs: A recept maximális szénhidráttartalma (grammban).
- **Search Recipes by Ingredients**
    - HTTP metódus: GET
    - Hozzáférési pont: https://api.spoonacular.com/recipes/findByIngredient
    - Paraméterek: Lásd: https://spoonacular.com/food-api/docs#Search-Recipes-by-Ingredients
- **Get Recipe Information**
    - HTTP metódus: GET
    - Hozzáférési pont: https://api.spoonacular.com/recipes/{id}/information
    - Paraméterek: Lásd: https://spoonacular.com/food-api/docs#Get-Recipe-Information
- **Ingredient Search**
    - HTTP metódus: GET
    - Hozzáférési pont: https://api.spoonacular.com/food/ingredients/search
    - Paraméterek: Lásd: https://spoonacular.com/food-api/docs#Ingredient-Search

- **Autocomplete Ingredient Search**
  - HTTP metódus: GET
  - Hozzáférési pont: https://api.spoonacular.com/food/ingredients/autocomplete
  - Paraméterek: Lásd: https://spoonacular.com/food-api/docs#Autocomplete-Ingredient-Search
- **Search Grocery Product**
  - HTTP metódus: GET
  - Hozzáférési pont: https://api.spoonacular.com/food/products/search
  - Paraméterek: Lásd: https://spoonacular.com/food-api/docs#Search-Grocery-Products
- **Get Product Information**
  - HTTP metódus: GET
  - Hozzáférési pont: https://api.spoonacular.com/food/products/{id}
  - Paraméterek: Lásd: https://spoonacular.com/food-api/docs#Get-Product-Information
- **Get Ingredient Information**
  - HTTP metódus: GET
  - Hozzáférési pont: https://api.spoonacular.com/food/ingredients/{id}/information
  - Paraméterek: Lásd: https://spoonacular.com/food-api/docs#Get-Ingredient-Information

8. Non-Functional Requirements

- **Performance**: Describe the performance requirements, such as response time and throughput.
  - A webalkalmazás teljesítménybeli célja, hogy a felhasználó számára élvezhető válaszidők mellett, funkcionálisan teljesen működő szolgáltatást nyújtson.
- **Security**: Security requirements, including authentication and authorization.
  - A felhasználói adatok titkosítva lesznek eltárolva. A titkosítást és a hozzáféréskezelést Spring Security-vel lesz megvalóstva.
- **Usability**: Usability standards and requirements.
  - A webalkalmazás egy könnyen érthető és használható, intuitív felületet tartalmaz a kezeléshez.
  - A keresőnek hatékonyan kell működnie, és lehetővé kell tennie a receptek gyors keresését az alapanyagok és az étkezési trendek alapján.
  - A felület reszponzív lesz a különböző képernyőméretekhez.
- **Reliability**: Requirements for system reliability and availability.

    - A webalkalmazás folyamatosan és megbízhatóan elérhetőnek kell lennie.
- **Scalability**: Requirements for system scalability.
    - A webalkalmazásnak tartania kell az élvezhető felhasználást a felhasználók és az adatok növekedése mellett is.
- **Compliance**: Legal and regulatory compliance requirements.
    - Jogi követelménybe nem ütközünk.

9. Technology Stack

- **Programming Languages**: Java (Maven), JavaScript (React), HTML
- **Frameworks and Libraries**: Description of frameworks and libraries to be used.
    - **Spring Framework:** A Spring Framework átfogó programozási és konfigurációs modellt kínál a modern Java-alapú vállalati alkalmazásokhoz - bármilyen telepítési platformon. Sokszínű komponenspalettája megkönnyíti a fejlesztők munkáját, ezzel támogatva a könnyebben átlátható és jól működő alkalmazások előállítását. Ezek közül a következőket használjuk:
        - **Spring Web:** Webalkalmazások alapját adja, amelyek a Spring MVC-t használják. Apache Tomcat-et használ alapértelmezett containerként.
        - **Spring Data JPA:** A Spring Data JPA célja, hogy jelentősen javítsa az adathozzáférési rétegek implementálását azáltal, hogy automatikus Query-ket generál a létrehozott Model-ekhez, de persze képesek vagyunk egyedi Query-ket is írni.
        - **Spring Security:** A Spring Security egy hatékony és nagymértékben testreszabható hitelesítési és hozzáférés-ellenőrzési keretrendszer. Ez a biztonsági standard a Spring alapú alkalmazásoknál.
    - **MySQL Server Driver:** Az MySQL Server-hez történő csatlakozáshoz szükséges
    - **Lombok:** Java annotációs könyvtár, ami csökkenti az ismételődő kód írását.
    - **React:** Egy JavaScript alapú könyvtár, amit a felhasználói felület fejlesztésére használhatunk. Lehetővé teszi komponensek létrehozását, ami segíti a jobban átlátható kód írását, és a felhasználóbarát megjelenítést.
- **Development Tools**: Jetbrains IntelliJ IDEA
- **Deployment Environment**: Description of the deployment environment, including hardware and software.
    - Az alkalmazás teljes mértékben konténerizált, így Docker segítségével bármely eszközről futtatható.
    - **Ajánló rendszer**: Az ajánló rendszert a Gorse biztosítja. A Gorse egy nyílt forráskódú ajánlórendszer-motor, amely Go nyelven íródott. Különböző kollaboratív, felhasználó-a-felhasználóhoz, illetve elem-az-elemhez ajánlatokat ad a felhasználó tevékenysége alapján.