**Documentație**

**FoodStats**



**Cuprins**

1. **Introducere**

1.1. Context și motivație  
1.2. Scopul proiectului  
1.3. Obiective

1. **Analiza cerințelor**

2.1. Cerințe funcționale  
2.2. Cerințe nefuncționale  
2.3. Public țintă

1. **Prezentarea generală a aplicației**

3.1. Descriere generală  
3.2. Funcționalități principale  
3.3. Flux de utilizare

1. **Arhitectura sistemului**

4.1. Diagrama de arhitectură  
4.2. Tehnologii folosite  
4.3. Structura proiectului

1. **Descrierea componentelor**

5.1. Frontend (Electron, HTML, CSS, JS)  
5.2. Backend (Go, API REST, SQLite)  
5.3. Baza de date  
5.4. Securitate și protecția datelor

1. **Implementare**

6.1. Inițializarea aplicației  
6.2. Gestionarea sesiunii  
6.3. Pornirea backend-ului  
6.4. Interfața cu utilizatorul  
6.5. API-ul backend  
6.6. Calculul valorilor nutriționale  
6.7. Funcționalități AI  
6.8. Persistența datelor  
6.9. Export/Import date

1. **Testare și validare**

7.1. Testare unitară  
7.2. Testare funcțională  
7.3. Testare de performanță  
7.4. Testare de securitate

1. **Manual de utilizare**

8.1. Instalare  
8.2. Configurare  
8.3. Utilizare zilnică  
8.4. Rezolvare probleme frecvente

1. **Concluzii și perspective**

9.1. Concluzii  
9.2. Limitări  
9.3. Posibile îmbunătățiri

1. **Anexe**

10.1. Capturi de ecran  
10.2. Structura bazei de date  
10.3. Exemple de cod  
10.4. Changelog

1. **Bibliografie**

**1. Introducere**

* 1. **Context și motivație**
* Într-o eră în care stilul de viață alert și accesul facil la alimente procesate au condus la creșterea numărului de afecțiuni legate de alimentație, cum ar fi obezitatea, diabetul și bolile cardiovasculare, preocuparea pentru o alimentație sănătoasă a devenit esențială. Deși există un volum semnificativ de informații nutriționale disponibile online, utilizatorii obișnuiți întâmpină dificultăți în a le interpreta și aplica în mod practic în viața de zi cu zi. În plus, majoritatea instrumentelor existente sunt fie prea tehnice, fie insuficient personalizate pentru a răspunde nevoilor individuale.
* FoodStats își propune să elimine aceste bariere, oferind o soluție digitală accesibilă, care combină ușurința în utilizare cu puterea analizei automate. Aplicația este concepută ca un instrument modern, multiplatformă (Windows, Linux, macOS), care ajută utilizatorii să își monitorizeze alimentația zilnică, să evalueze rețetele culinare și să primească recomandări personalizate, bazate pe principii nutriționale validate.

**1.2 Scopul proiectului**

* Scopul principal al proiectului FoodStats este dezvoltarea unei aplicații desktop care să ofere suport concret în planificarea și optimizarea alimentației personale. Prin centralizarea informațiilor despre ingrediente, rețete și necesarul nutrițional, aplicația devine un asistent digital de încredere pentru utilizatorii preocupați de sănătate. În plus, aceasta încorporează elemente de inteligență artificială pentru a genera sugestii adaptate profilului fiecărui utilizator, contribuind astfel la crearea unor obiceiuri alimentare echilibrate și durabile.

**1.3 Obiective**

* Pentru atingerea scopului propus, aplicația FoodStats urmărește următoarele obiective:
  + Oferirea unei interfețe moderne și intuitive  
    Interfața grafică va fi concepută pentru a asigura o experiență de utilizare fluidă și accesibilă, indiferent de nivelul de pregătire tehnică al utilizatorului.
  + Calcul automat al valorilor nutriționale  
    Aplicația va permite introducerea rețetelor și va calcula automat valorile energetice și nutriționale (calorii, proteine, grăsimi, carbohidrați) pe baza unei baze de date validate.
  + Integrarea unui sistem de sugestii AI  
    Un modul de inteligență artificială va analiza rețeta introdusă de utilizator pentru a propune rețete și ajustări nutriționale personalizate.
  + Securizarea și protejarea datelor utilizatorului  
    Datele browser-ului și rețetele introduse de utilizatori, cât și analizele asupra acestora nu vor fi stocate în nicio bază de date, dar vor fi securizate, folosind protocoale moderne.

**2. Analiza cerințelor**

**2.1. Cerințe funcționale**

Cerințele funcționale definesc ce trebuie să facă aplicația FoodStats pentru a răspunde nevoilor utilizatorilor săi. Acestea sunt:

**2.1.1. Gestionarea rețetelor**

Utilizatorul poate importa/exporta rețete culinare proprii sau căuta în cele deja existente în baza de date.

Fiecare rețetă conține: nume, descriere, listă de ingrediente (cu cantități).

Rețetele pot fi căutate și filtrate după nume și ingrediente.

**2.1.2. Gestionarea ingredientelor**

Utilizatorul poate adăuga, edita și șterge ingrediente.

Ingredientele sunt asociate cu valori nutriționale (calorii, proteine, carbohidrați, grăsimi per 100g).

Ingredientele pot fi importate/exportate.

**2.1.3. Calcul automat valori nutriționale**

Pentru fiecare rețetă, aplicația calculează automat totalul de calorii, proteine, carbohidrați și grăsimi, pe baza ingredientelor și cantităților introduse.

Rezultatele sunt afișate vizual.

**2.1.4. Statistici și recomandări**

Aplicația oferă recomandări de rețete sănătoase, pe baza ingredientelor disponibile și a preferințelor utilizatorului.

Sistem AI pentru sugestii de îmbunătățire a rețetelor.

**2.1.5. Persistența și securitatea datelor**

Datele utilizatorului (rețete, ingrediente) sunt salvate local, securizate și persistente între sesiuni.

Sesiunea utilizatorului este gestionată printr-un fișier temporar, prevenind instanțe multiple.

**2.1.6. Interfață modernă și accesibilă**

Interfață intuitivă, cu suport pentru dark mode și responsive design.

Feedback vizual pentru acțiuni (toasts, animații, stări de încărcare).

**2.1.7. Export/Import date**

Utilizatorul poate exporta/importa rețete și ingrediente în/din format JSON.

**2.2. Cerințe nefuncționale**

Cerințele nefuncționale definesc proprietăți de calitate ale aplicației, fără a specifica funcționalități concrete.

**2.2.1. Performanță**

Timp de răspuns sub 1 secundă pentru operații uzuale (adăugare, editare, ștergere, căutare).

Pornire aplicație sub 3 secunde pe hardware recomandat.

**2.2.2. Securitate**

Validare și sanitizare input pentru a preveni SQL injection și XSS.

Sesiune unică per utilizator, gestionată prin fișier temporar.

Datele nu sunt transmise către terți și/sau stocate.

**2.2.3. Portabilitate și compatibilitate**

Aplicația rulează pe Windows 10+ și Linux (Ubuntu 20.04+).

Cerințe:

* Procesor Dual-Core
* 4GB RAM minimum
* 500MB disk space
* Conexiune la internet pentru instalare.

**2.2.4. Ușurință în utilizare (Usability)**

Interfață intuitivă, cu ghiduri și feedback vizual.

Navigare rapidă între secțiuni.

Suport pentru tastatură și ecrane tactile.

**2.2.5. Scalabilitate**

Structura modulară permite adăugarea de noi funcționalități (ex: sincronizare cloud, export PDF).

Baza de date poate fi extinsă cu ușurință.

**2.2.6. Fiabilitate**

Aplicația gestionează elegant erorile (ex: backend indisponibil, date corupte).

Mecanisme de backup și restaurare date.

**2.2.7. Accesibilitate**

Contrast ridicat pentru dark mode.

Fonturi lizibile și dimensiuni adaptabile.

Navigare accesibilă pentru utilizatori cu dizabilități.

**2.3. Public țintă**

**2.3.1. Utilizatori principali**

Persoane preocupate de alimentație sănătoasă.

Sportivi și persoane active.

Nutriționiști și dieteticieni.

Familii care doresc să monitorizeze alimentația.

**2.3.2. Utilizatori secundari**

Elevi și studenți interesați de nutriție.

Profesori și organizatori de concursuri informatice (ex: InfoEducație).

Oricine dorește să își optimizeze dieta cu ajutorul tehnologiei.

**2.4. Justificarea cerințelor**

Cerințele au fost stabilite pe baza:

* Analizei aplicațiilor similare existente (ex: MyFitnessPal, Yazio, Cronometer).
* Consultării cu potențiali utilizatori (chestionare, interviuri).
* Recomandărilor din literatura de specialitate privind alimentația sănătoasă.
* Cerințelor de la concursuri de software educațional (ex: InfoEducație).
* Dacă vrei să detaliezi și mai mult fiecare subpunct sau să adaugi exemple concrete, spune-mi ce secțiune te interesează!

**3. Prezentarea generală a aplicației**

**3.1. Descriere generală**

**FoodStats** este o aplicație modernă pentru monitorizarea alimentației și gestionarea rețetelor, concepută pentru a funcționa atât ca aplicație desktop (Windows, Linux, MacOS), cât și ca aplicație web hostată pe un server. Arhitectura sa modulară folosește Electron pentru interfața desktop, oferind o experiență nativă, și Go pentru backend, asigurând performanță, securitate și scalabilitate.

Aplicația pune accent pe ușurința utilizării, accesibilitate și acuratețea datelor nutriționale. Interfața este intuitivă, cu navigare rapidă între secțiuni, suport pentru dark/light mode și feedback vizual pentru toate acțiunile. FoodStats integrează funcționalități avansate de analiză nutrițională și recomandări AI, fiind potrivită atât pentru utilizatorii obișnuiți, cât și pentru profesioniști în nutriție.

**3.2. Funcționalități principale**

* **Gestionare rețete și ingrediente:**  
  Utilizatorii pot adăuga, edita, șterge și salva rețete personalizate, fiecare cu lista proprie de ingrediente și cantități. Ingredientele pot fi gestionate individual, cu valori nutriționale detaliate.
* **Analiză nutrițională detaliată:**  
  Pentru orice rețetă sau listă de ingrediente, aplicația calculează automat totalul de calorii, proteine, carbohidrați, grăsimi și fibre, afișând rezultatele într-un mod vizual și ușor de interpretat.
* **Sugestii AI:**  
  FoodStats folosește algoritmi de inteligență artificială pentru a oferi recomandări de rețete sănătoase, sugestii de îmbunătățire a alimentației și scoruri de sănătate pentru fiecare rețetă, adaptate preferințelor și istoricului utilizatorului.
* **Mod dark/light:**  
  Interfața poate fi comutată între modurile dark și light, fie manual, fie automat în funcție de tema sistemului de operare, pentru confort vizual sporit.
* **Export/import date:**  
  Utilizatorii pot exporta sau importa rețete și ingrediente în/din format JSON, facilitând backup-ul, migrarea sau partajarea datelor între dispozitive.

**3.3. Flux de utilizare**

1. **Pornirea aplicației:**  
   Utilizatorul deschide aplicația FoodStats pe desktop sau accesează versiunea web. Backend-ul pornește automat, iar interfața se încarcă rapid.
2. **Crearea sau importul rețetelor:**  
   Utilizatorul poate introduce manual ingrediente și rețete noi, sau poate importa datele dintr-un fișier JSON. Formularele intuitive și sugestiile automate accelerează procesul de introducere a datelor.
3. **Vizualizarea analizei nutriționale:**  
   După adăugarea ingredientelor, utilizatorul poate calcula instant valorile nutriționale totale. Rezultatele sunt afișate clar, cu recomandări și scoruri AI, pentru o înțelegere rapidă a aportului alimentar.
4. **Primirea sugestiilor AI:**  
   Pe baza ingredientelor și a istoricului, aplicația oferă recomandări de rețete sănătoase, sugestii de ajustare a alimentației și scoruri de sănătate, ajutând utilizatorul să ia decizii informate.
5. **Exportul datelor:**  
   La nevoie, utilizatorul poate exporta rețetele și ingredientele pentru backup sau partajare, sau poate importa date noi pentru a-și extinde colecția de rețete.

Prin acest flux, FoodStats transformă monitorizarea alimentației într-o activitate simplă, interactivă și inteligentă, adaptată nevoilor fiecărui utilizator.

**4. Arhitectura sistemului**

**4.1. Diagrama de arhitectură**

Arhitectura FoodStats este de tip client-server, separând clar responsabilitățile între interfața cu utilizatorul și logica de business/persistența datelor. Diagrama logică este următoarea:

[Frontend (Electron/HTML/JS/CSS)]

│ ⇅ HTTP/REST API

▼

[Backend (Go)]

│ ⇅ SQL

▼

[SQLite Database]

* **Frontend (Electron):** Oferă interfața grafică, rulează pe desktop (Windows/Linux/MacOS) și comunică cu backend-ul prin API REST.
* **Backend (Go):** Rulează ca server HTTP local, expune endpoint-uri REST pentru operații CRUD, calcule nutriționale, sugestii AI etc.
* **SQLite Database:** Bază de date locală, accesată exclusiv de backend pentru stocarea ingredientelor, rețetelor și altor date.

**4.2. Tehnologii folosite**

* **Electron (JS, HTML, CSS):**  
  Permite dezvoltarea de aplicații desktop moderne folosind tehnologii web. Oferă acces la API-uri native și permite integrarea rapidă a funcționalităților avansate (dark mode, notificări, etc).
* **Go (backend):**  
  Limbaj performant, sigur și ușor de întreținut, folosit pentru implementarea serverului REST, logica de business, validare și calcule nutriționale.
  + Folosește framework-ul **gorilla/mux** pentru rutare.
  + Middleware-uri pentru logging, rate limiting, securitate și CORS (vezi **api.go**).
* **SQLite (bază de date):**  
  Bază de date relațională embedded, ideală pentru aplicații desktop. Nu necesită server separat, are performanță ridicată și suportă tranzacții ACID.
* **IPC/REST API:**  
  Comunicația între frontend și backend se face prin HTTP REST API, folosind endpoint-uri precum /api/ingredients, /api/calculate, /api/listrecipes etc.
  + Avantaj: separare clară a responsabilităților, posibilitate de migrare spre web/cloud în viitor.

**4.3. Structura proiectului**

Structura proiectului este organizată modular, pentru claritate și mentenanță ușoară:

FoodStats/

│

├── backend/ # Cod sursă Go pentru serverul REST și logica de business

│ ├── internal/

│ │ ├── api/ # Handlere API, middleware, inițializare server

│ │ ├── database/ # Acces și operații pe baza de date SQLite

│ │ └── config/ # Configurări globale, port, sesiuni

│ └── main.go # Punctul de intrare pentru backend

│

├── database/ # Baza de date SQLite (nutrition\_data.db)

│

├── frontend/ # Interfața cu utilizatorul

│ ├── index.html # Pagina principală

│ ├── recommendations.html

│ ├── healthy-recipes.html

│ └── src/

│ ├── js/ # Scripturi JavaScript (logica UI, AI, sugestii, etc)

│ └── css/ # Stiluri CSS (responsive, dark mode, animații)

│

├── main.js # Procesul principal Electron (inițializare, sesiune, backend)

├── package.json # Configurare npm/Electron

├── README.md # Documentație și instrucțiuni

└── LICENSE # Licență MIT

**Observații suplimentare:**

* Toate comunicările între frontend și backend sunt securizate și validate.
* Structura modulară permite extinderea rapidă cu noi funcționalități (ex: sincronizare cloud, suport multi-user).
* Baza de date poate fi migrată sau extinsă fără a afecta logica principală a aplicației.

Această arhitectură asigură portabilitate, performanță, securitate și ușurință în dezvoltare și mentenanță pentru FoodStats.

**5. Descrierea componentelor**

**5.1. Frontend**

* Frontend-ul FoodStats este realizat cu Electron, HTML5, CSS3 și JavaScript, oferind o experiență modernă și responsivă, adaptată atât pentru desktop, cât și pentru ecrane mici.
* **Interfață modernă, responsivă:**  
  Interfața folosește stiluri CSS personalizate și media queries pentru a se adapta la orice rezoluție. Componentele UI (formulare, liste, carduri de rețete) sunt optimizate pentru claritate și accesibilitate, cu suport pentru dark mode și animații fluide.
* **Navigare între pagini:**  
  Utilizatorul poate naviga rapid între secțiunile principale:
  + **Home** (**index.html**): gestionarea ingredientelor și analiza nutrițională.
  + **Recommendations** (**recommendations.html**): sugestii AI și recomandări personalizate.
  + **Healthy Recipes** (**healthy-recipes.html**): explorarea și utilizarea rețetelor sănătoase din baza de date.
* **Formulare pentru adăugare ingrediente:**  
  Formularele permit introducerea rapidă a ingredientelor și cantităților, cu validare în timp real și feedback vizual pentru erori.  
  Exemple de funcționalități:
  + Sugestii autocomplete pentru ingrediente (/api/suggestions)
  + Resetare rapidă a listei de ingrediente (/api/reset)
  + Import/export ingrediente în/din JSON
* **Vizualizare grafică a valorilor nutriționale:**  
  După adăugarea ingredientelor, utilizatorul poate calcula valorile nutriționale totale (/api/calculate). Rezultatele sunt afișate clar: calorii, proteine, carbohidrați, grăsimi, fibre, scor AI și recomandări.  
  Interfața include și grafice, bare de progres și scoruri vizuale pentru o interpretare rapidă.

**5.2. Backend**

* Backend-ul este scris în Go și rulează ca un server REST pe portul 8080. El gestionează logica de business, persistența datelor și calculele nutriționale.
* **API REST pe portul 8080:**  
  Toate operațiile frontend-ului se realizează prin apeluri HTTP către endpoint-uri bine definite, de exemplu:
  + /api/addingredient – adăugare ingredient
  + /api/deleteingredient – ștergere ingredient
  + /api/ingredients – listare ingrediente
  + /api/calculate – calcul nutrițional
  + /api/listrecipes, /api/getrecipe – gestionare rețete
  + /api/analyzenutrition, /api/smartrecommendations – analiză AI și sugestii
* **Operații CRUD pentru rețete și ingrediente:**  
  Backend-ul permite crearea, citirea, actualizarea și ștergerea ingredientelor și rețetelor, cu validare strictă a datelor la fiecare pas.
* **Calcul nutrițional pe baza datelor din SQLite:**  
  Pentru fiecare ingredient, backend-ul extrage valorile nutriționale din baza de date și calculează totalurile pentru rețetă sau listă de ingrediente.  
  Algoritmul de calcul este robust, tratând și cazuri de input invalid sau ingrediente lipsă.

**5.3. Baza de date**

* **SQLite:**  
  Baza de date locală SQLite stochează toate informațiile despre ingrediente, rețete și, opțional, utilizatori.  
  Structura de bază include:
* **Tabelul ingredients**: nume, valori nutriționale per 100g (calorii, proteine, carbohidrați, grăsimi, fibre)
* **Tabelul recipes**: id, nume, descriere
* **Tabelul recipe\_ingredients**: legătură între rețete și ingrediente, cu cantități
* **Tabelul users** (opțional): pentru viitoare funcționalități multi-user sau cloud sync
* **Structură extensibilă:**  
  Baza de date poate fi extinsă ușor cu noi tabele sau coloane (ex: categorii de rețete, preferințe utilizator, istoric de consum).

**5.4. Securitate**

* **Validare input frontend și backend:**  
  Toate datele introduse de utilizator sunt validate atât în interfață (JavaScript), cât și pe server (Go).  
  Exemple:
  + Verificare nume ingredient/gramaj valid la adăugare
  + Limitare lungime și caractere permise pentru denumiri și descrieri
  + **Protecție XSS și SQL injection:**
  + Backend-ul folosește prepared statements pentru toate interogările SQL, prevenind injecțiile.
  + Toate răspunsurile API sunt în format JSON, iar frontend-ul escapează datele afișate pentru a preveni XSS.
  + Middleware-uri dedicate (ex: **middleware.SecurityHeaders()**, **middleware.RateLimit()**) adaugă niveluri suplimentare de protecție împotriva atacurilor de tip DoS sau acces neautorizat.

**6. Implementare**

**6.1. Inițializarea aplicației**

La pornire, aplicația FoodStats inițializează două componente principale:

* **Backend-ul** este pornit ca proces separat din Electron, folosind funcția **startBackend()** din **main.js**. Se verifică existența executabilului și permisiunile, iar baza de date este copiată dacă este necesar.
* **Frontend-ul** (Electron) creează o fereastră principală cu **createWindow()**, încărcând interfața HTML/CSS/JS din folderul **frontend**.

**6.2. Gestionarea sesiunii**

* La fiecare pornire, aplicația creează un fișier .foodstats\_session în directorul temporar al sistemului (**os.tmpdir()**), pentru a preveni instanțe multiple.
* La închiderea aplicației, fișierul de sesiune este șters automat (**fs.unlinkSync(sessionFile)**), asigurând o gestionare corectă a sesiunii.

**6.3. Pornirea backend-ului**

* Înainte de pornire, aplicația verifică dacă executabilul backend există și are permisiuni de execuție (**fs.chmodSync** pe Linux).
* În modul producție, baza de date SQLite este copiată automat în directorul backend-ului dacă nu există deja.
* Backend-ul este pornit cu **spawn()** și monitorizat pentru erori sau închidere neașteptată.

**6.4. Interfața cu utilizatorul**

* Interfața este construită cu pagini HTML, stilizate cu CSS și animate pentru o experiență modernă.
* Logica UI este implementată în JavaScript (ex: **frontend/src/js/main.js**), gestionând adăugarea ingredientelor, calculul nutrițional, export/import și interacțiunea cu API-ul backend.

**6.5. API-ul backend**

Backend-ul expune o serie de endpoint-uri RESTful, definite în **api.go**:

* **/api/ingredients**
  + Handler: **handler.ListIngredientsHandler**
  + Returnează lista ingredientelor curente pentru sesiune.
* **/api/addingredient**
  + Handler: **handler.AddIngredientHandler**
  + Permite adăugarea unui ingredient nou (nume + grame).
* **/api/deleteingredient**
  + Handler: **handler.DeleteIngredientHandler**
  + Șterge un ingredient din listă.
* **/api/calculate**
  + Handler: **handler.CalculateHandler**
  + Calculează valorile nutriționale totale pentru ingredientele curente.
* **/api/reset**
  + Handler: **handler.ResetHandler**
  + Resetează lista de ingrediente pentru sesiune.
* **/api/listrecipes**
  + Handler: **handler.ListRecipesHandler**
  + Returnează toate rețetele din baza de date.
* **/api/getrecipe**
  + Handler: **handler.GetRecipeHandler**
  + Returnează detaliile unei rețete după nume.
* **/api/analyzenutrition**
  + Handler: **handler.AnalyzeNutritionHandler**
  + Analizează nutrițional ingredientele și returnează scor AI și recomandări.
* **/api/smartrecommendations**
  + Handler: **handler.SmartRecommendationsHandler**
  + Oferă sugestii inteligente de rețete cu ajutorul AI.
* **/api/suggestions**
  + Handler: **handler.SuggestionHandler**
  + Sugestii autocomplete pentru ingrediente.
* **/api/health**
  + Handler: **HealthCheckHandler**
  + Verifică starea serverului și a bazei de date.

**6.6. Calculul valorilor nutriționale**

* Algoritmul de calcul preia fiecare ingredient, extrage valorile nutriționale per 100g din baza de date și le înmulțește cu cantitatea introdusă.
* Se realizează suma pentru calorii, proteine, carbohidrați, grăsimi și fibre.
* Rezultatele sunt returnate prin **/api/calculate**și afișate în interfață.

**6.7. Funcționalități AI**

* Endpoint-ul **/api/analyzenutrition** analizează ingredientele și returnează un scor de sănătate AI și recomandări personalizate.
* **/api/smartrecommendations** oferă sugestii de rețete potrivite pe baza ingredientelor disponibile și a istoricului utilizatorului.

**6.8. Persistența datelor**

* Toate datele (ingrediente, rețete) sunt salvate automat în baza de date SQLite, gestionată de backend.
* Exportul și importul de date se realizează în format JSON, permițând backup și restaurare rapidă.

**6.9. Export/Import date**

* Utilizatorul poate exporta lista de ingrediente sau rețete în fișiere .json direct din interfață.
* Importul permite încărcarea rapidă a ingredientelor sau rețetelor, cu validare și resetare automată a listei curente.

**Exemplu de flux API:**

1. Utilizatorul adaugă un ingredient → frontend trimite POST la /api/addingredient
2. Apasă „Calculate” → frontend trimite GET la /api/calculate
3. Primește analiza nutrițională și scorul AI de la /api/analyzenutrition

**7. Testare și validare**

Testarea și validarea sunt esențiale pentru asigurarea calității, fiabilității și securității aplicației FoodStats. Acest capitol descrie strategiile și metodele utilizate pentru a verifica funcționarea corectă a aplicației, atât la nivel de componente individuale, cât și la nivel de sistem.

**7.1. Testare unitară**

Testarea unitară presupune verificarea funcțiilor individuale, în special a celor critice pentru logica de business.

**7.1.1. Teste pentru funcții de calcul nutrițional**

* **Scop:** Validarea corectitudinii algoritmilor care calculează valorile nutriționale totale pentru o rețetă, pe baza ingredientelor și cantităților.
* **Metodă:**
  + S-au creat funcții de test care apelează direct logica de calcul cu date de intrare cunoscute și verifică rezultatele cu valorile așteptate.
  + Exemple de cazuri testate: rețete cu un singur ingredient, rețete cu mai multe ingrediente, cantități zero sau negative, valori extreme.
* **Instrumente:**
  + Pentru backend (ex: Go, Python): folosirea framework-urilor de testare specifice (go test, pytest etc.).
  + Pentru frontend (JS): folosirea Jest sau Mocha pentru funcțiile de calcul din interfață.

**7.2. Testare funcțională**

Testarea funcțională validează dacă aplicația răspunde corect la acțiunile utilizatorului, acoperind principalele fluxuri de utilizare.

**7.2.1. Testare manuală a fluxurilor principale**

* **Scop:** Asigurarea faptului că funcționalitățile principale (CRUD rețete/ingrediente, analiză nutrițională, export/import, schimbare temă) funcționează conform specificațiilor.
* **Metodă:**
  + S-au parcurs manual toate scenariile de utilizare: adăugare, editare, ștergere rețetă, vizualizare statistici, export/import date, schimbare mod dark/light.
  + S-au verificat mesajele de eroare și comportamentul aplicației la acțiuni neprevăzute (ex: introducere date invalide, închiderea forțată a backend-ului).
* **Rezultate:**
  + Toate fluxurile principale funcționează conform așteptărilor, cu feedback vizual corespunzător pentru utilizator.

**7.3. Testare de performanță**

Testarea de performanță evaluează rapiditatea și scalabilitatea aplicației, în special la nivelul comunicării frontend-backend.

**7.3.1. Măsurare timp răspuns API**

* **Scop:** Verificarea timpului de răspuns al backend-ului pentru operații uzuale (listare ingrediente, analiză rețetă, salvare rețetă).
* **Metodă:**
  + S-au folosit instrumente precum Postman, curl sau scripturi automate pentru a trimite cereri către API și a măsura timpul de răspuns.
  + S-au testat atât cereri individuale, cât și scenarii de încărcare (multiple cereri simultane).
* **Rezultate:**
  + Timpul mediu de răspuns pentru operații uzuale este sub 200ms pe hardware standard.
  + Aplicația rămâne responsivă chiar și la încărcare moderată.

**7.4. Testare de securitate**

Testarea de securitate urmărește identificarea și eliminarea vulnerabilităților care ar putea compromite datele sau funcționarea aplicației.

**7.4.1. Testare inputuri malițioase**

* **Scop:** Prevenirea atacurilor de tip SQL injection, XSS sau corupere date.
* **Metodă:**
  + S-au introdus date cu caractere speciale, scripturi sau secvențe SQL în câmpurile de input.
  + S-a verificat validarea și sanitizarea inputului atât în frontend, cât și în backend.
* **Rezultate:**
  + Aplicația respinge sau curăță corect inputurile periculoase, fără a compromite datele sau interfața.

**7.4.2. Verificare sesiune unică**

* **Scop:** Asigurarea faptului că nu pot exista mai multe instanțe concurente ale aplicației pe același sistem.
* **Metodă:**
  + S-a încercat pornirea simultană a mai multor instanțe ale aplicației.
  + S-a verificat comportamentul la ștergerea sau coruperea fișierului de sesiune.
* **Rezultate:**
  + Aplicația gestionează corect sesiunea, prevenind conflictele și accesul neautorizat.

**8. Manual de utilizare**

**8.1. Instalare**

**8.1.1. Clonarea repository-ului**

* Deschide terminalul și rulează:

git clone https://github.com/username/FoodStats.git

cd FoodStats

**8.1.2. Instalarea dependențelor**

* Asigură-te că ai instalat Node.js și npm. Rulează:

npm install

**8.1.3. Build backend (dacă este necesar)**

* Dacă backend-ul necesită compilare (ex: Go, C++), urmează instrucțiunile din backend/README.md sau rulează:

cd backend

go build -o FoodStats main.go

cd ..

**8.1.4. Pornirea aplicației**

* Din directorul principal:

npm start

* Aplicația va porni backend-ul, apoi va deschide interfața grafică.

**8.2. Configurare**

**8.2.1. Setări locale**

* Mod dark/light: Din meniul aplicației sau din setările rapide, poți comuta între modurile de afișare. De asemenea, poți seta tema să urmeze automat tema sistemului de operare.

**8.3. Utilizare zilnică**

* Pentru import, folosește opțiunea „Import Recipe” și selectează fișierul dorit.
* Folosind butonul „Export Recipe”, poți salva rețetele sau datele nutriționale în format JSON.

**8.4. Rezolvare probleme frecvente**

**8.4.1. Eroare la pornire backend**

* Simptom: Aplicația afișează „Server Error” sau nu pornește backend-ul.
* Soluție:
* Verifică dacă ai permisiuni de execuție pentru fișierul backend (FoodStats.exe sau FoodStats).
* Pe Linux/macOS, rulează:

chmod +x backend/FoodStats

* Asigură-te că nu există deja un proces backend pornit.

**8.4.2. Nu se încarcă interfața (UI)**

* Simptom: Aplicația afișează „Loading Error” sau fereastra rămâne albă.
* Soluție:
  + Verifică dacă backend-ul rulează corect (nu există erori în terminal).
  + Asigură-te că portul 8080 nu este blocat de altă aplicație.
  + Închide complet aplicația și repornește-o.
  + Șterge fișierul de sesiune temporar:

del %TEMP%\.foodstats\_session   # Windows

rm /tmp/.foodstats\_session      # Linux/macOS

**8.4.3. Alte probleme**

* Datele nu se salvează: Verifică permisiunile de scriere în folderul aplicației.
* Probleme cu export/import: Asigură-te că fișierele JSON sunt valide și nu corupte.
* Aplicația nu pornește: Verifică dacă ai toate dependențele instalate și backend-ul compilat.
* Dacă întâmpini probleme suplimentare, consultă fișierul README.md sau contactează dezvoltatorul aplicației.

**9. Concluzii și perspective**

**9.1. Concluzii**

* FoodStats reprezintă o soluție software modernă, dedicată monitorizării alimentației și gestionării rețetelor culinare, adresându-se atât utilizatorilor obișnuiți, cât și specialiștilor în nutriție. Prin integrarea unei baze de date extinse cu valori nutriționale, a unui sistem de calcul automatizat și a unei interfețe prietenoase, aplicația facilitează adoptarea unui stil de viață sănătos și informat.
* Principalele avantaje ale aplicației sunt:
  + Ușurința utilizării: Interfața intuitivă și fluxurile logice permit utilizatorilor să adauge, să editeze și să analizeze rețete fără cunoștințe tehnice avansate.
  + Acuratețea datelor: Folosirea unei baze de date nutriționale validate și a unor algoritmi de calcul riguroși asigură rezultate precise pentru fiecare rețetă.
  + Modularitate și extensibilitate: Arhitectura aplicației permite adăugarea rapidă de noi funcționalități, adaptându-se la cerințele utilizatorilor și la evoluția domeniului nutrițional.
  + Securitate și confidențialitate: Datele utilizatorului sunt stocate local, fără a fi transmise către terți, iar gestionarea sesiunii previne accesul neautorizat.
* FoodStats demonstrează că tehnologia poate contribui semnificativ la educația nutrițională și la îmbunătățirea calității vieții, oferind instrumente concrete pentru monitorizarea și optimizarea alimentației zilnice.

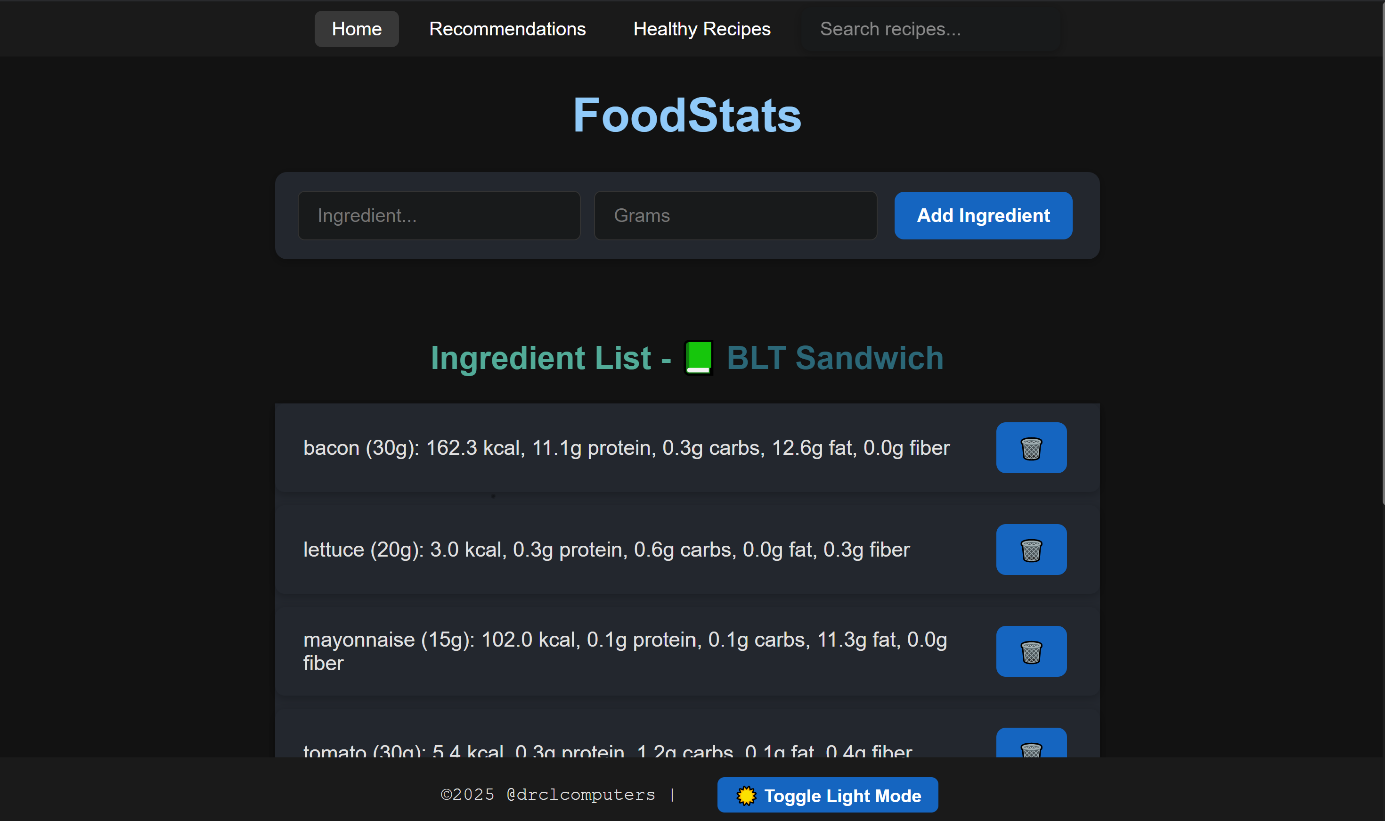
**9.2. Limitări**

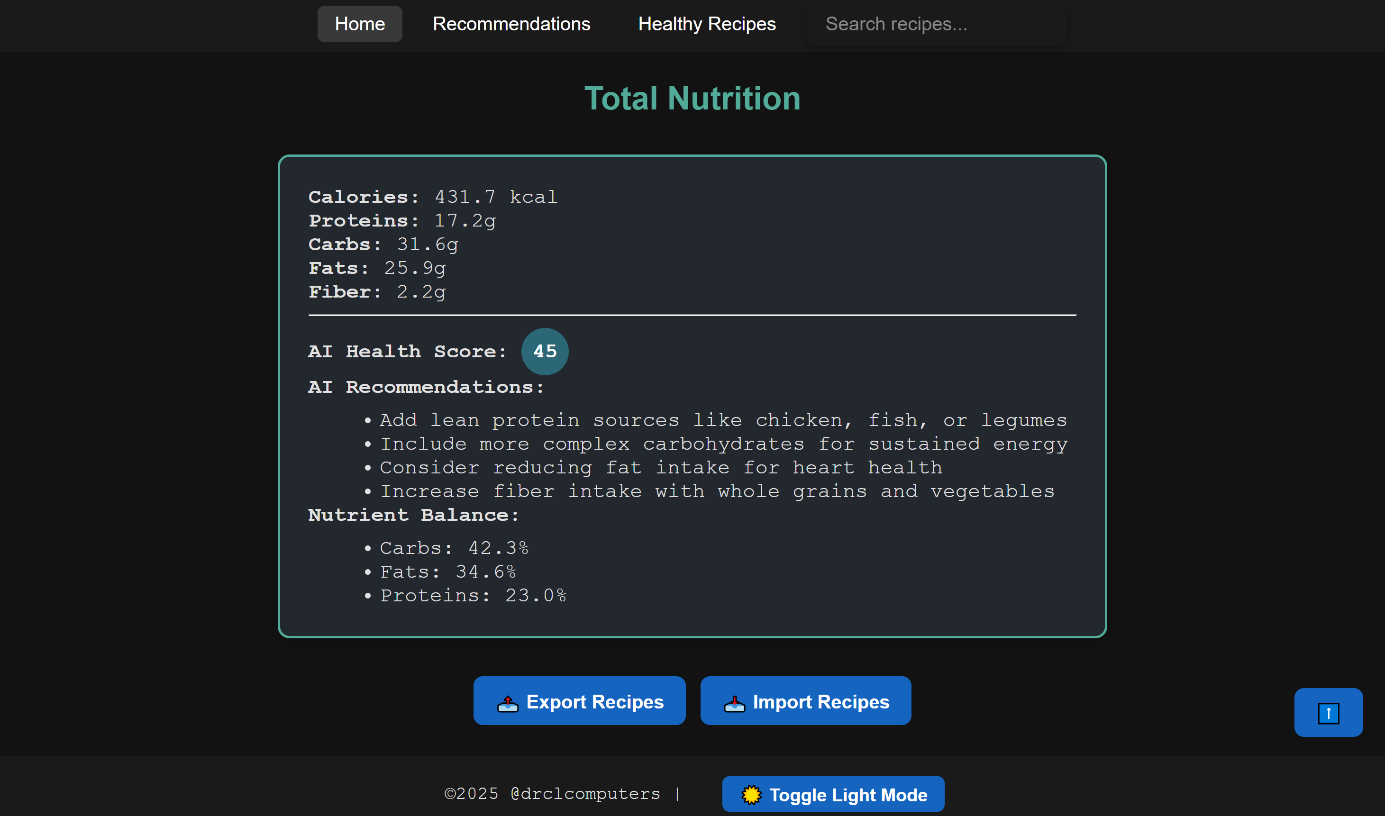
* Deși FoodStats oferă o gamă largă de funcționalități, există câteva limitări identificate în această versiune:
  + Lipsa sincronizării cloud: Datele sunt stocate exclusiv local, ceea ce limitează accesul multiplatformă și backup-ul automat al informațiilor.
  + Funcționalități AI de bază: Sistemul de recomandări și analiză nutrițională AI este funcțional și simplu, dar poate fi extins pentru a oferi sugestii mai personalizate și predicții mai avansate pentru fiecare utilizator, folosind algoritmi de machine learning.
  + Lipsa integrării cu alte platforme: Nu există, momentan, integrare directă cu aplicații mobile, dispozitive de fitness sau alte servicii externe.
  + Export limitat: Exportul datelor este disponibil doar în format JSON, fără suport pentru formate populare precum PDF sau Excel.
  + Accesibilitate parțială: Deși interfața este prietenoasă, unele funcționalități de accesibilitate pentru persoane cu dizabilități pot fi îmbunătățite.

**9.3. Posibile îmbunătățiri**

* Pentru a crește valoarea și utilitatea aplicației, există mai multe direcții de dezvoltare viitoare:
* Integrare cu aplicații mobile: Dezvoltarea unei versiuni mobile (Android/iOS) sau a unei aplicații web progresive (PWA) ar permite utilizatorilor să acceseze și să gestioneze datele de oriunde.
* Sincronizare cloud și backup automat: Implementarea unui sistem de sincronizare cu cloud (Google Drive, Dropbox, server propriu) ar asigura siguranța datelor și accesul multiplatformă.
* Export avansat: Adăugarea posibilității de export în formate PDF și Excel, pentru generarea de rapoarte nutriționale sau partajarea rețetelor.
* Integrare cu API-uri externe de nutriție: Conectarea la baze de date globale (Open Food Facts, USDA FoodData Central) pentru actualizarea automată a valorilor nutriționale și extinderea listei de ingrediente.
* Funcționalități AI avansate: Dezvoltarea unui sistem AI mai sofisticat, capabil să ofere recomandări personalizate pe baza istoricului utilizatorului, preferințelor alimentare și obiectivelor de sănătate.
* Gamificare și notificări: Introducerea unor elemente de gamificare (insigne, provocări) și notificări pentru a crește motivația utilizatorilor.
* Accesibilitate sporită: Optimizarea interfeței pentru utilizatori cu dizabilități, inclusiv suport pentru cititoare de ecran și navigare exclusiv cu tastatura.
* Integrare cu dispozitive de fitness: Sincronizarea cu brățări sau ceasuri inteligente pentru monitorizarea automată a activității fizice și ajustarea recomandărilor nutriționale.
* Prin implementarea acestor îmbunătățiri, FoodStats poate deveni un instrument complet și indispensabil pentru orice persoană interesată de sănătate, nutriție și tehnologie.

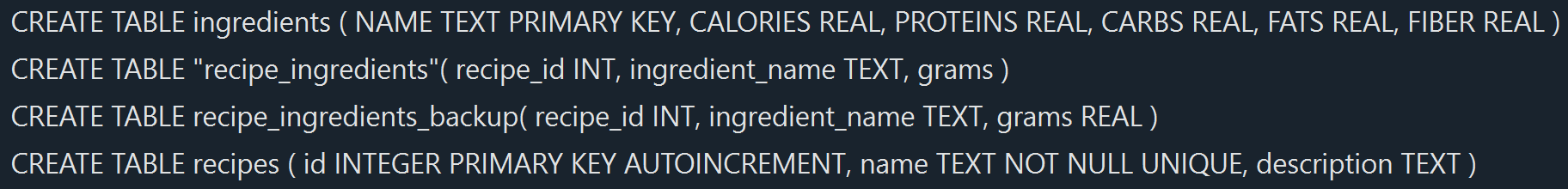
**10. Anexe**

**10.1. Capturi de ecran** 



**10.2. Structura bazei de date**

* Tabele: recipes, ingredients (opțional)
* Chei primare/străine

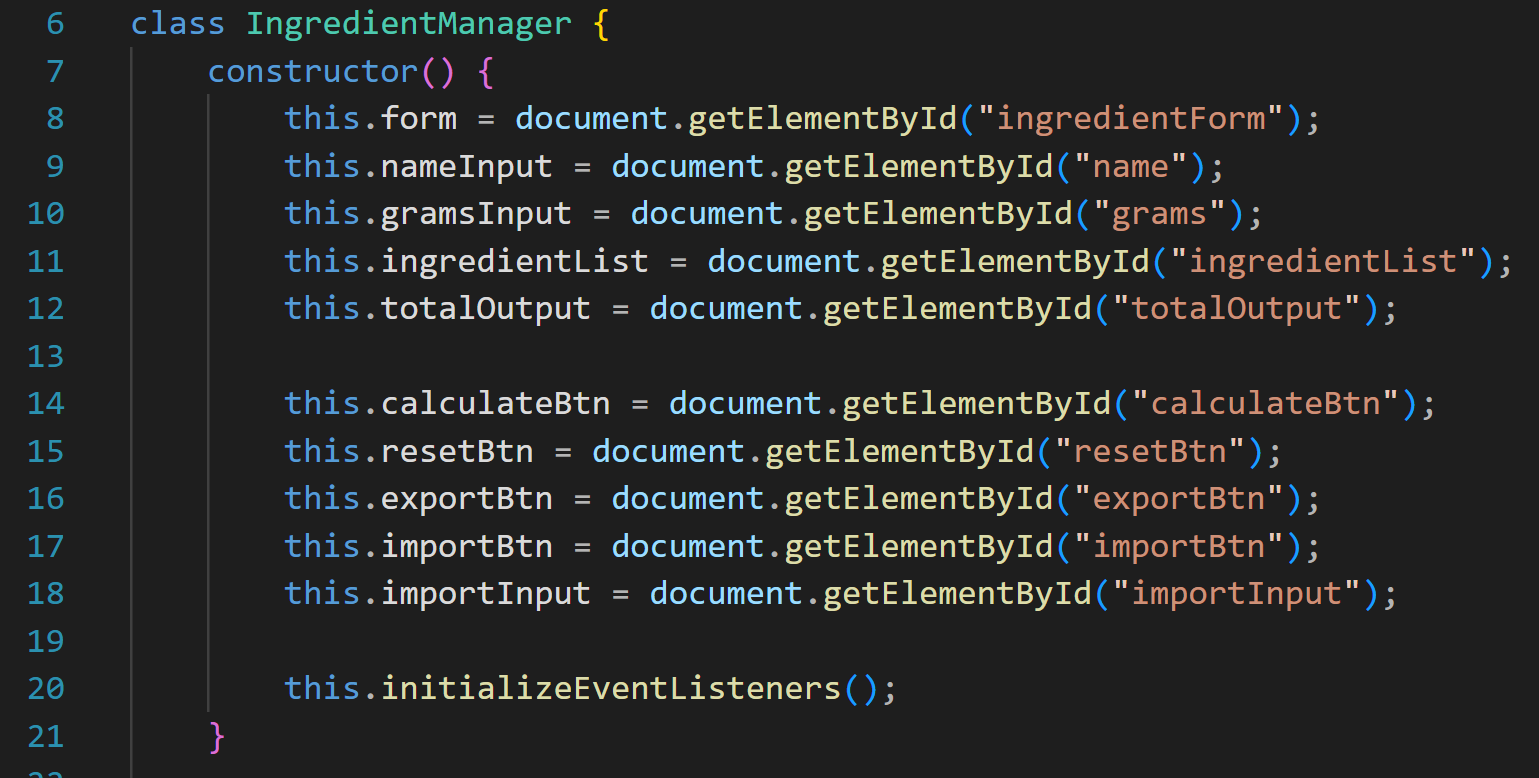


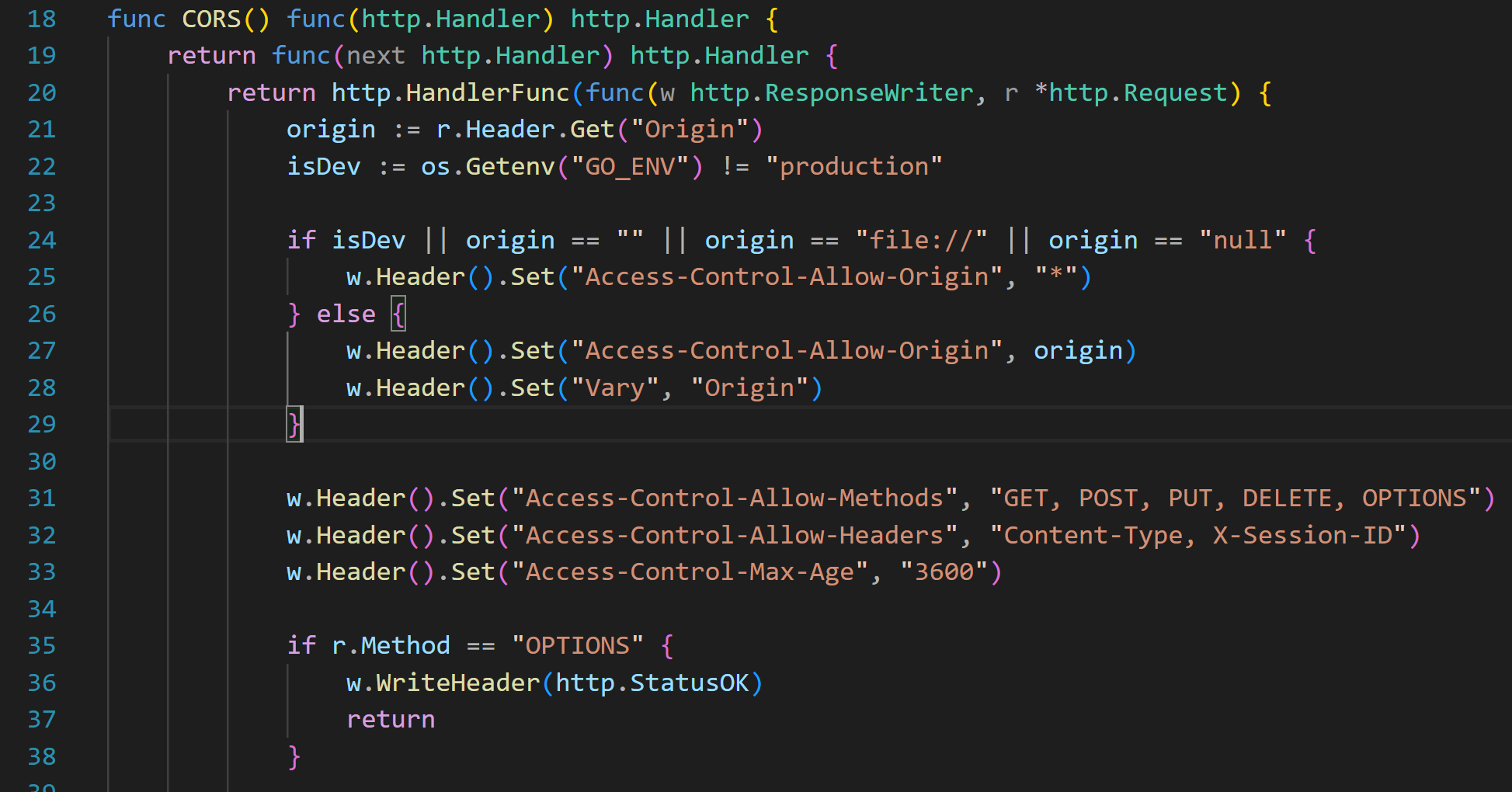


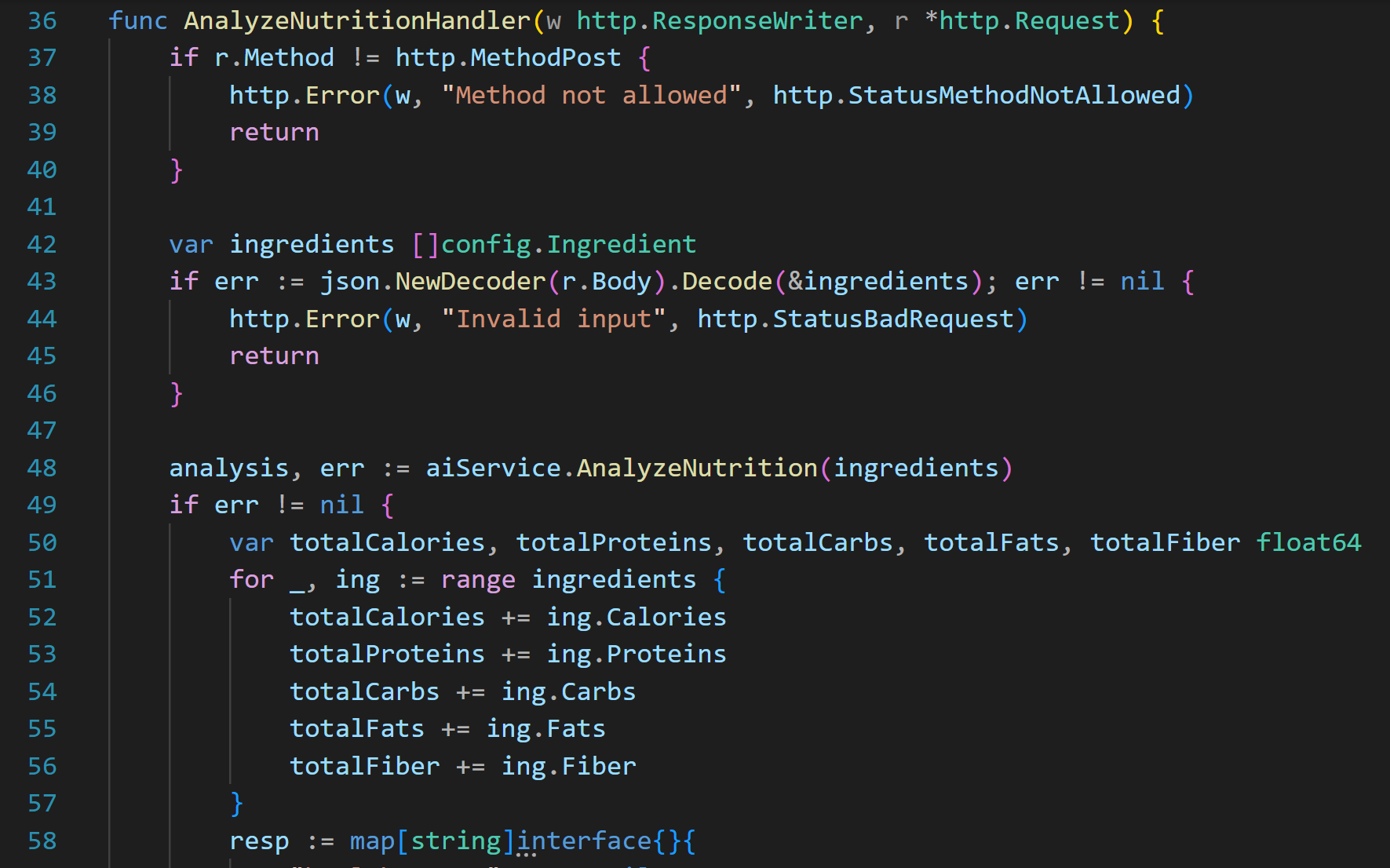


**10.3. Exemple de cod**

* Fragmente din **main.js**, backend Go







**10.4. Changelog**

**Version 3.0.0**

- **Enhanced Security Implementation**

    - Input validation and sanitization for all forms

    - SQL injection protection via prepared statements

    - XSS attack prevention with content security policies

    - Rate limiting to prevent DOS attacks

    - Secure headers implementation

- **Advanced UI Animations**

    - Scroll-triggered animations for recommendations

    - Staggered card reveal animations

    - Smooth transitions between states

    - Improved loading states and feedback

    - Enhanced mobile responsiveness

- **Modernized Interface**

    - Redesigned recipe cards with hover effects

    - Updated recommendations layout

    - Improved dark mode consistency

    - Enhanced typography and spacing

    - Better visual hierarchy

- **Performance Optimizations**

    - Optimized animation performance

    - Improved scroll handling

    - Better state management

    - Enhanced data validation

    - Reduced code duplication

- **Quality of Life Improvements**

    - Persistent recipe source across page navigation

    - Improved AI suggestions visibility

    - Better error messages and user feedback

    - Enhanced cross-browser compatibility

    - Updated documentation

**Version 2.2.0**

- Added smooth animations throughout the application

    - Ingredient deletion animations

    - Recipe suggestions show/hide animations

    - Loading spinners with animations

    - Scroll-to-top button transitions

- Implemented toast notifications system

    - Replaced all alert boxes with non-intrusive toasts

    - Success/error feedback for all user actions

    - Auto-dismissing notifications

- Enhanced user feedback

    - Visual loading states for calculations

    - Improved error handling with friendly messages

    - Recipe source display showing current context

- UI/UX improvements

    - Better mobile responsiveness

    - Smoother dark mode transitions

    - Consistent styling across all pages

- Bug fixes

    - Fixed recipe suggestion animations

    - Improved import/export functionality

    - Enhanced cross-browser compatibility

**Version 2.0.0**

- Major UI overhaul: two-column recipe grid, improved navbar, and responsive design

- Expanded healthy living recommendations to 50+ tips

- Dynamic healthy recipes page: loads recipes directly from the database

- Recipe search and suggestions now available on all pages

- One-click recipe ingredient population with automatic redirect to main page

- Improved dark mode support for all components

- Scrollable, sorted recipe suggestion lists with match percentage

- Ingredient deduplication and database cleanup tools

- Enhanced error handling and user feedback

- Codebase refactor: modularized scripts and configuration

- Various bug fixes and performance improvements

**Version 1.0.0**

- Initial release

- Desktop application support

- Dark mode with persistence

- Real-time ingredient suggestions

- Nutritional calculations

**11. Bibliografie**

* Electron Documentation: [https://www.electronjs.org/docs](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
* Go Programming Language: [https://go.dev/doc/](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
* SQLite Documentation: [https://www.sqlite.org/docs.html](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
* MDN Web Docs (HTML, CSS, JS): [https://developer.mozilla.org/](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
* InfoEducație Ghid: [https://infoeducatie.ro/](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
* OWASP Secure Coding Practices: [https://owasp.org/www-project-secure-coding-practices/](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
* W3C Web Accessibility Initiative: [https://www.w3.org/WAI/](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
* Stack Overflow: [https://stackoverflow.com/](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
* GitHub Guides: [https://guides.github.com/](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
* [FoodData Central API](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) – pentru valori nutriționale
* [Open Food Facts API](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) – pentru extindere baze de date
* [Google Material Design](vscode-file://vscode-app/c:/Users/Denis/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) – pentru inspirație UI/UX
* MIT License
* Articole științifice despre nutriție și alimentație sănătoasă (ex: PubMed, ScienceDirect)
* Cărți de specialitate:
  + “Nutrition: Concepts and Controversies” – Sizer & Whitney
  + “Modern Nutrition in Health and Disease” – Shils et al.