

Universitatea Transilvania din Braşov  
Facultatea de Matematică şi Informatică

# DOCUMENTAȚIA PROIECTULUI

NutriSense

STUDENȚI

Lixandru Valentina Mariana  
Palatka Vanessa

# Cuprins

1. Prezentarea proiectului, ce isi propune, ce probleme rezolva
2. Tehnologiile folosite
3. Baza de date: diagrama bazei de date + scurta prezentare a tabelelor si a relatiilor dintre ele
4. Prezentarea API-ului: screenshot la Swagger pentru a putea vedea toate endpoint-urile + scurta descriere a CRUD-urilor
5. Prezentare despre cum poate fi utilizata aplicatia: tipuri de utilizatori, ce vede fiecare, autentificare etc
6. Concluzii si contributii (cum v-ati impartit task-urile, ce ati invatat in urma acestui proiect etc)
7. Link GIT catre codul proiectului

## 1. Prezentarea proiectului

NutriSense este o aplicaţie mobilă Android inovatoare, concepută pentru a transforma modul în care oamenii îşi gestionează alimentaţia zilnică. Aplicaţia îşi propune să fie ghidul personal al fiecărui utilizator către un stil de viaţă mai sănătos şi echilibrat.

Mulţi oameni vor să mănânce sănătos, dar nu ştiu cum să calculeze kaloriile sau să înţeleagă ce conţin alimentele. NutriSense rezolvă această problemă făcând totul simplu şi rapid.

Pentru a ajuta oamenii să țină o dietă echilibrată, aplicaţia NutriSense oferă:

- Calculează automat şi precis valorile nutriţionale pentru orice aliment, adaptându-se la cantitatea specifică introdusă de utilizator
- Urmăreşte în timp real consumul de calorii, proteine, carbohidraţi şi grăsimi, oferind o imagine clară asupra aportului nutriţional
- Facilitează căutarea şi salvarea reţetelor pe baza ingredientelor disponibile sau a preferinţelor culinare specifice
- Calculează BMI-ul (indicele de masă corporală) şi estimează necesarul optim de calorii şi hidratare zilnică, adaptate profilului individual

Prin integrarea acestor funcţionalităţi într-o interfaţă intuitivă şi prietenoasă, NutriSense devine partenerul de încredere în călătoria fiecărui utilizator către o alimentaţie mai conştientă şi un stil de viaţă mai sănătos.

## 2. Tehnologiile folosite

NutriSense este o aplicaţie Android scrisă în **Kotlin**, un limbaj de programare modern şi sigur. Pentru a stoca datele pe telefon, am folosit **Room Database**, care păstrează toate informaţiile despre utilizatori, alimente şi reţete chiar şi când nu ai internet.

Pentru a lua informaţiile despre nutriţie şi reţete, aplicaţia se conectează la internet şi foloseşte două servicii externe: *CalorieNinjas API* (pentru datele nutriţionale) şi *API Ninjas Recipe* (pentru reţete). Comunicarea cu aceste servicii se face prin **Retrofit**, o bibliotecă care face mai uşoară trimiterea şi primirea de date.

Aplicaţia foloseşte **Navigation Component** pentru a trece de la un ecran la altul şi **Material Design** pentru a arăta frumos şi modern. Pentru ca aplicaţia să nu se blocheze când lucrează cu datele, am folosit **Kotlin Coroutines**, care permit ca operaţiunile grele să se facă în fundal.

## 3. Baza de date

### 1. Tabela USERS

Stochează informaţiile utilizatorilor înregistraţi.

1. id - Cheia primară, auto-increment
2. email - Adresa de email (unicitate asigurată la nivel de aplicaţie)
3. password
4. firstName
5. lastName
6. age

7. createdAt

## 2. Tabela FOODS

Stochează alimentele căutate şi valorile lor nutriţionale.

1. userId - Referinţă către utilizator (Foreign Key)
2. name
3. originalQuery
4. requestedQuantityG
5. calories
6. proteinG
7. carbohydratesTotalG
8. fatTotalG
9. sodiumMg
10. potassiumMg
11. cholesterolMg
12. fiberG
13. sugarG
14. addedAt
15. isFavorite
16. consumedAt

## 3. Tabela RECIPES

Stochează reţetele găsite şi salvate de utilizatori.

1. userId - Referinţă către utilizator (Foreign Key)
2. title
3. ingredients
4. servings
5. instructions
6. searchQuery
7. addedAt
8. isFavorite

### Relaţiile dintre tabele:

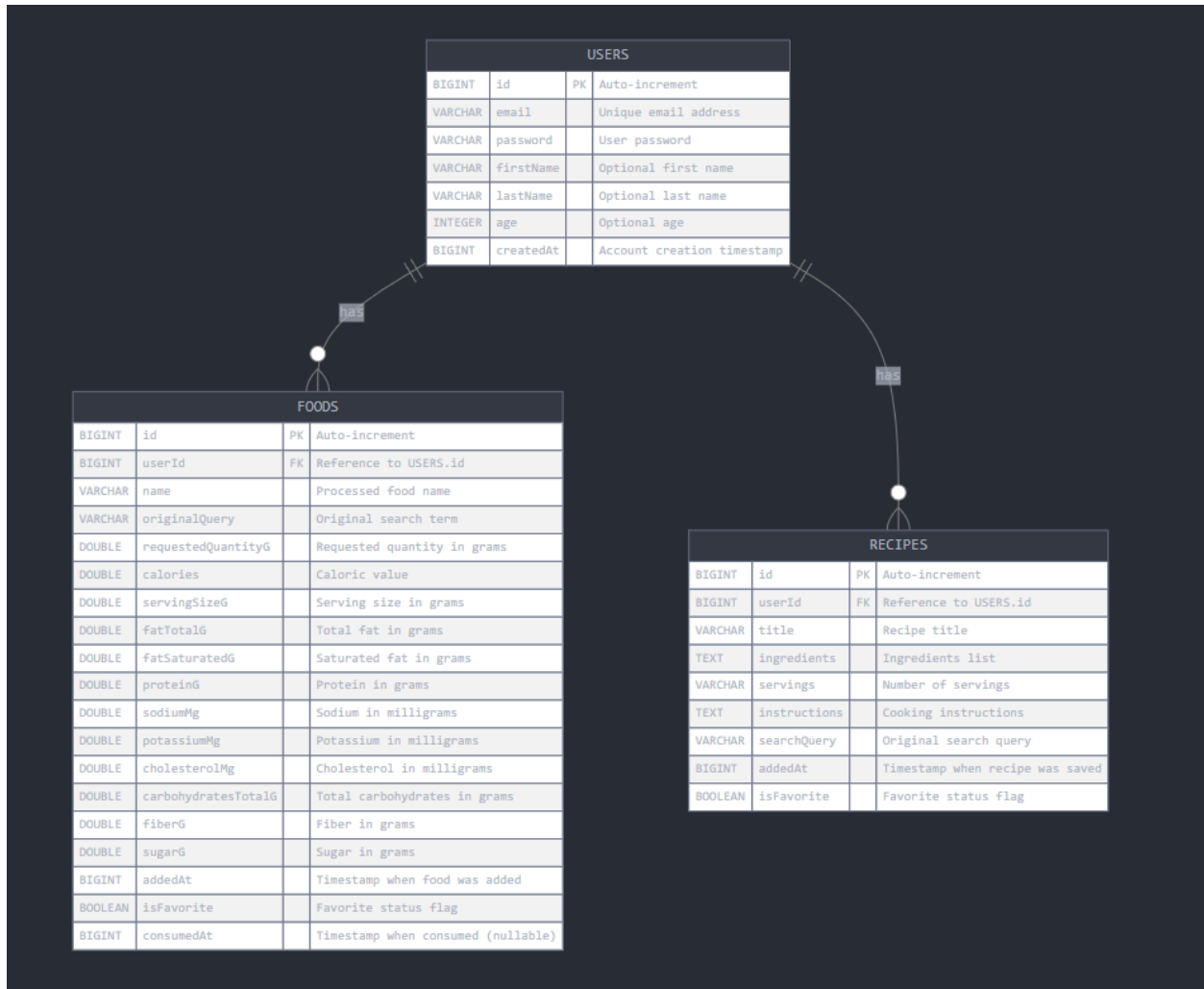
*Users*  $\leftarrow$  *Foods* (One-to-Many)

- Un utilizator poate avea multiple alimente salvate
- Fiecare aliment aparţine unui singur utilizator

*Users*  $\leftarrow$  *Recipes* (One-to-Many)

- Un utilizator poate avea multiple reţete salvate
- Fiecare reţetă aparţine unui singur utilizator

Universitatea Transilvania din Braşov  
Facultatea de Matematică şi Informatică



## 4. Prezentarea API-ului

### API-uri externe folosite:

#### 1. CalorieNinjas API

API pentru obținerea informațiilor nutriționale detaliate despre alimente.

**Endpoint:** <https://api.calorieninjas.com/v1/nutrition>

**Autentificare:** API Key în header (X-API-Key)

#### Exemplu de request:

GET /v1/nutrition?query=100g apple

X-API-Key: ZeMJNIaz7JLI2rDpSuL3jQ==nScJs7HfYucaDzUY

### Exemplu de response:

json

```
{
  "items": [
    {
      "name": "apple",
      "calories": 52,
      "serving_size_g": 100,
      "fat_total_g": 0.2,
      "fat_saturated_g": 0.1,
      "protein_g": 0.3,
      "sodium_mg": 1,
      "potassium_mg": 107,
      "cholesterol_mg": 0,
      "carbohydrates_total_g": 14,
      "fiber_g": 2.4,
      "sugar_g": 10
    }
  ]
}
```

### 2. API Ninjas Recipe API

API pentru căutarea reţetelor pe baza ingredientelor sau pe baza numelui reţetei.

**Endpoint:** <https://api.api-ninjas.com/v1/recipe>

**Autentificare:** API Key în header (X-API-Key)

### Exemplu de request:

GET /v1/recipe?query=chicken rice

X-API-Key: P7PaMWKKnD3FqEmU5N6q+iA==MvB2cPjgUqgZPqb

### Exemplu de response:

json

```
[
  {
    "title": "Chicken Rice Bowl",
```

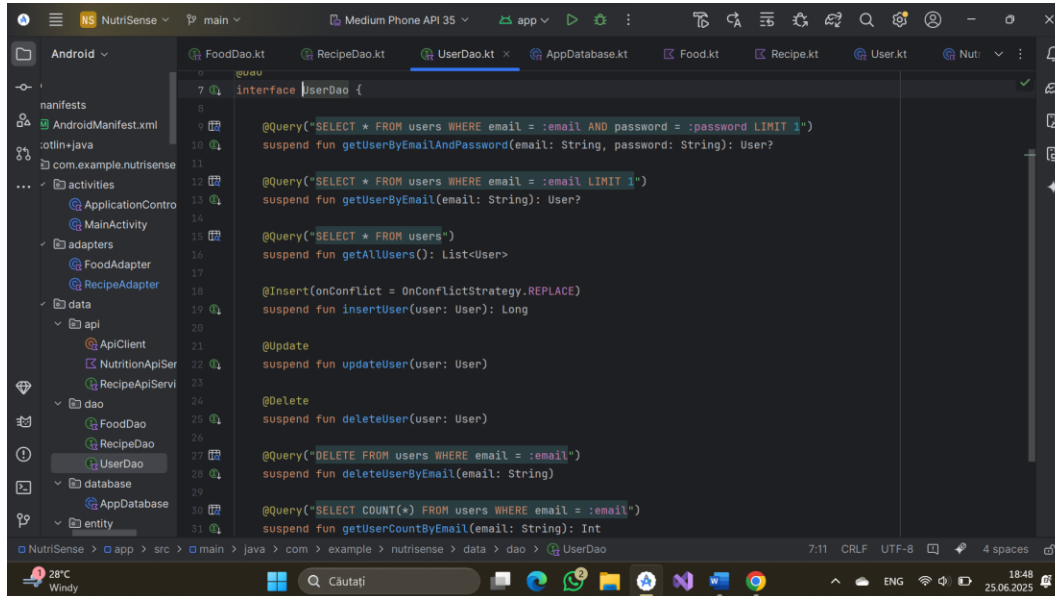
Universitatea Transilvania din Braşov  
Facultatea de Matematică şi Informatică

```
"ingredients": "1 cup rice, 200g chicken breast, vegetables",  
"servings": "2",  
"instructions": "1. Cook rice. 2. Grill chicken. 3. Mix together."
```

```
}  
]
```

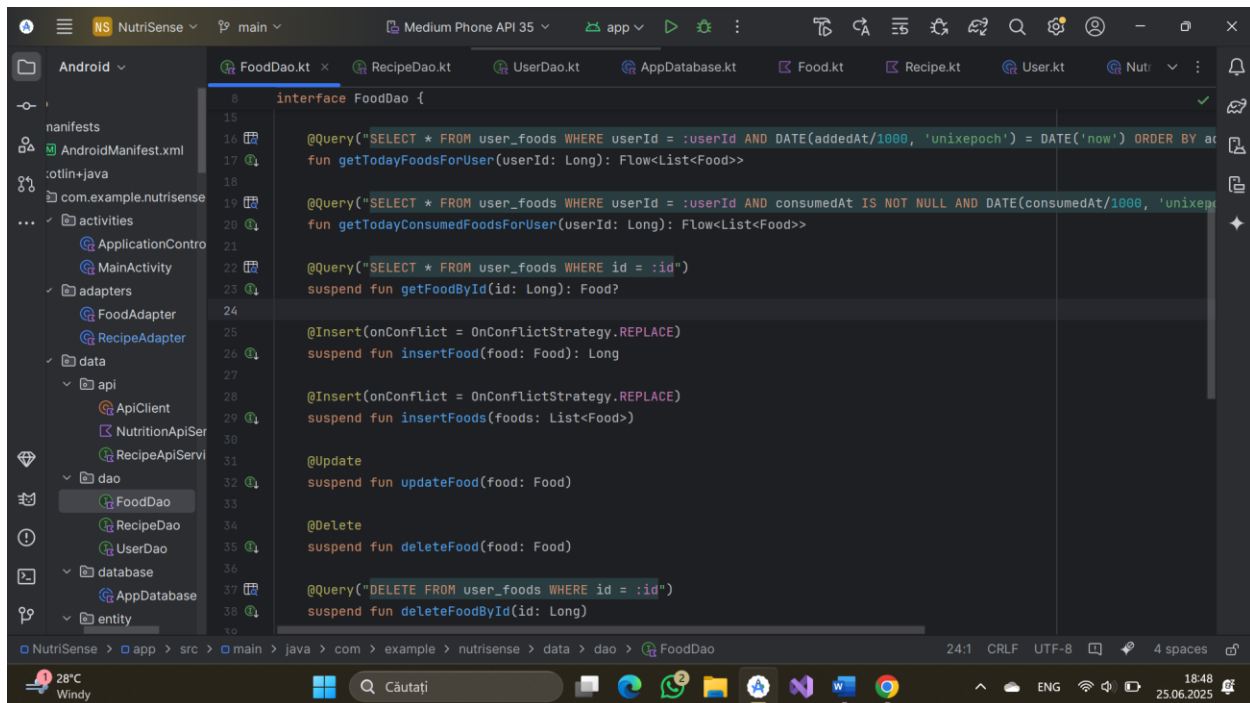
## API-ul intern (Room Database CRUD) folosit:

### 1. Users CRUD



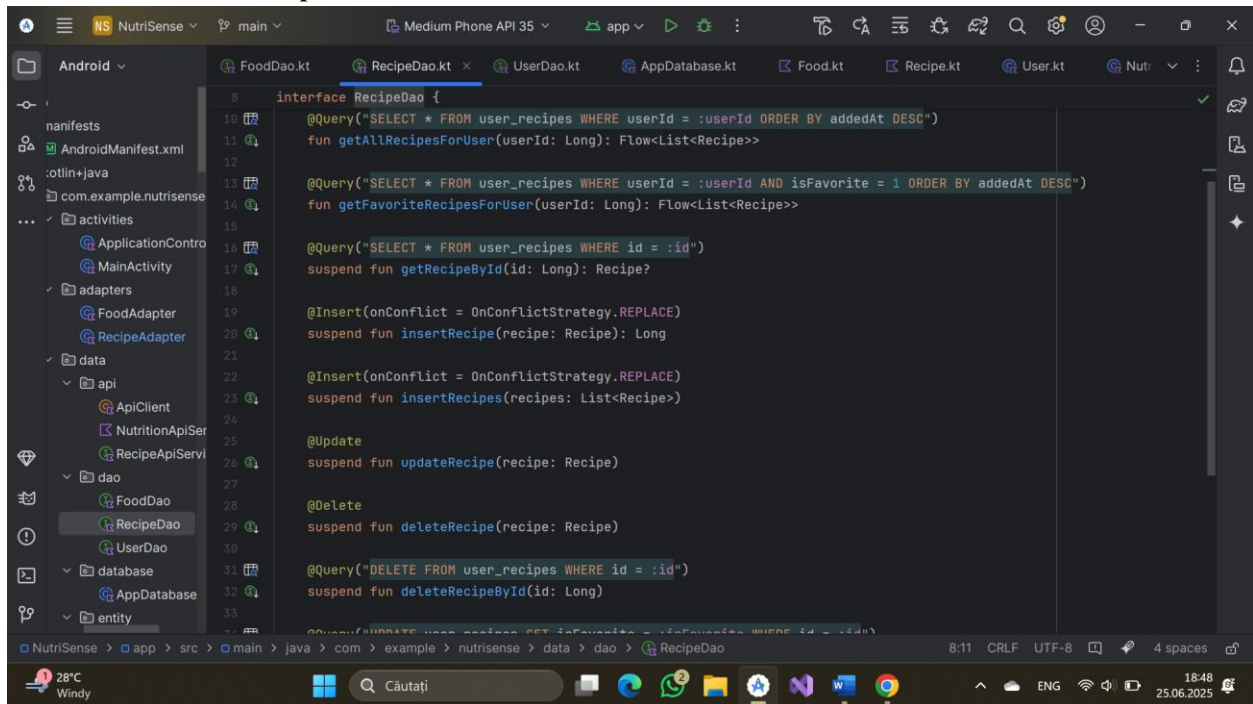
```
7 interface UserDao {  
8  
9     @Query("SELECT * FROM users WHERE email = :email AND password = :password LIMIT 1")  
10    suspend fun getUserByEmailAndPassword(email: String, password: String): User?  
11  
12    @Query("SELECT * FROM users WHERE email = :email LIMIT 1")  
13    suspend fun getUserByEmail(email: String): User?  
14  
15    @Query("SELECT * FROM users")  
16    suspend fun getAllUsers(): List<User>  
17  
18    @Insert(onConflict = OnConflictStrategy.REPLACE)  
19    suspend fun insertUser(user: User): Long  
20  
21    @Update  
22    suspend fun updateUser(user: User)  
23  
24    @Delete  
25    suspend fun deleteUser(user: User)  
26  
27    @Query("DELETE FROM users WHERE email = :email")  
28    suspend fun deleteUserByEmail(email: String)  
29  
30    @Query("SELECT COUNT(*) FROM users WHERE email = :email")  
31    suspend fun getUserCountByEmail(email: String): Int  
32  
33 }
```

### 2. Foods CRUD



```
8 interface FoodDao {  
9  
10    @Query("SELECT * FROM user_foods WHERE userId = :userId AND DATE(addedAt/1000, 'unixepoch') = DATE('now') ORDER BY addedAt")  
11    fun getTodayFoodsForUser(userId: Long): Flow<List<Food>>  
12  
13    @Query("SELECT * FROM user_foods WHERE userId = :userId AND consumedAt IS NOT NULL AND DATE(consumedAt/1000, 'unixepoch') = DATE('now')")  
14    fun getTodayConsumedFoodsForUser(userId: Long): Flow<List<Food>>  
15  
16    @Query("SELECT * FROM user_foods WHERE id = :id")  
17    suspend fun getFoodById(id: Long): Food?  
18  
19    @Insert(onConflict = OnConflictStrategy.REPLACE)  
20    suspend fun insertFood(food: Food): Long  
21  
22    @Insert(onConflict = OnConflictStrategy.REPLACE)  
23    suspend fun insertFoods(foods: List<Food>)  
24  
25    @Update  
26    suspend fun updateFood(food: Food)  
27  
28    @Delete  
29    suspend fun deleteFood(food: Food)  
30  
31    @Query("DELETE FROM user_foods WHERE id = :id")  
32    suspend fun deleteFoodById(id: Long)  
33  
34 }
```

### 3. Recipes CRUD



## 5. Prezentare despre cum poate fi utilizată aplicația

### Tipuri de utilizatori

#### *Utilizatori neautentificați*

- Pot vedea doar ecranul de login/înregistrare
- Nu au acces la funcționalitățile aplicației

#### *Utilizatori autentificați*

- Dashboard principal cu toate opțiunile
- Calcularea și salvarea informațiilor nutriționale
- Gestionarea bazei de date personale de alimente
- Căutarea și salvarea rețetelor
- Configurarea profilului și a obiectivelor

### Procesul de autentificare

#### *Înregistrare (2 pași)*

1. Pasul 1 (LoginFragment): • Introducerea email și parolă • Validarea formatului email • Validarea lungimii parolei (min. 6 caractere)
2. Pasul 2 (Register2Fragment): • Introducerea informațiilor opționale (nume, prenume, vârstă) • Crearea contului în baza de date • Configurarea automată a preferințelor implicite



### *Login*

- Introducerea email şi parolă
- Verificarea în baza de date locală
- Păstrarea sesiunii în SharedPreferences

### Fluxul utilizatorului autentificat

#### *1. Dashboard Principal*

- Calculate Food Nutrition - Funcţia principală
- View Food History - Istoricul alimentelor
- Recipe Search - Căutarea reţetelor
- Recipe Collection - Colecţia de reţete
- Settings & Goals - Configurări
- My Profile - Profilul utilizatorului

#### *2. Calcularea nutriţiei (CalculateNutritionFragment)*

1. Introducerea numelui alimentului
2. Specificarea cantităţii în grame
3. Apelarea API-ului CalorieNinjas
4. Afişarea rezultatelor detaliate
5. Salvarea automată în baza de date

#### Informaţii afişate:

- Calorii totale
- Proteine, Carbohidraţi, Grăsimi
- Sodiu, Potasiu, Colesterol
- Fibre, Zahăruri

#### *3. Istoricul alimentelor (SearchHistoryFragment)*

- Sumar zilnic: Calorii şi macronutrienţi consumaţi
- Lista tuturor alimentelor salvate cu RecyclerView
- Marcarea ca favorite
- Marcarea ca consumate azi
- Ştergerea din baza de date
- Vizualizarea detaliilor nutriţionale

#### *4. Căutarea reţetelor (RecipeSearchFragment)*

1. Introducerea ingredientelor disponibile
2. Apelarea API-ului de reţete
3. Afişarea rezultatelor
4. Salvarea automată în colecţie

5. *Colecţia de reţete (RecipeHistoryFragment)*

- Lista tuturor reţetelor salvate
- Marcarea ca favorite
- Vizualizarea detaliată (ingrediente, instrucţiuni)
- Ştergerea din colecţie

6. *Setări şi obiective (SettingsFragment)*

Informaţii personale:

- Vârsta, genul, unităţile de măsură
- Greutatea şi înălţimea
- Nivelul de activitate fizică

Obiective nutriţionale:

- Obiectivul zilnic de calorii
- Obiectivul zilnic de apă
- Calcularea automată a recomandărilor pe baza BMR

Calcul automate:

- BMI cu categorizarea (Subponderal/Normal/Supraponderal/Obez)
- Necesarul caloric zilnic în funcţie de activitate

7. *Profilul utilizatorului (ProfileFragment)*

Informaţii afişate:

- Email-ul utilizatorului
- Obiectivele curente (calorii, apă)
- Greutatea şi alte statistici
- BMI şi categoria acestuia
- Ultima actualizare a greutateii

Gestionarea datelor

*SharedPreferences*

Date globale:

- Starea autentificării
- Email-ul utilizatorului curent
- Preferinţele de temă

Date specifice utilizatorului:

- Obiectivele nutriţionale personalizate
- Informaţiile fizice (greutate, înălţime, vârstă)
- Preferinţele de unităţi (metric/imperial)
- Setările de notificări

#### *Room Database*

- Persistenţa offline - toate datele rămân disponibile fără internet
- Sincronizare în timp real cu LiveData şi Flow
- Validarea datelor la nivel de repository

## **6. Concluzii şi contribuţii**

NutriSense a fost dezvoltat printr-un efort colaborativ al echipei, unde fiecare membru şi-a adus expertiza specifică pentru a crea o aplicaţie completă şi funcţională.

### *1. Lixandru Valentina Mariana*

Domenii de responsabilitate:

- Sistemul de preferinţe utilizator - implementarea SharedPreferences pentru setări personalizate şi persistenţa datelor
- Integrarea Recipe API - dezvoltarea funcţionalităţii de căutare şi descoperire a reţetelor
- Gestionarea colecţiei de reţete - implementarea salvării şi organizării reţetelor în baza de date
- Asigurarea calităţii - identificarea şi rezolvarea bug-urilor pentru o experienţă stabilă
- Design - contribuţii la îmbunătăţirea aspectului vizual şi a fluxurilor utilizatorului

### *2. Palatka Vanessa*

Domenii de responsabilitate:

- Sistemul de autentificare - implementarea login-ului şi înregistrării în două etape cu validări robuste
- Elementele grafice personalizate - crearea drawable-urilor şi a componentelor vizuale care definesc identitatea aplicaţiei
- Integrarea CalorieNinjas API - dezvoltarea funcţionalităţii de căutare şi calculare a informaţiilor nutriţionale
- Gestionarea datelor nutriţionale - implementarea salvării şi gestionării alimentelor în baza de date locală
- Design - contribuţii la interfaţa intuitivă şi atractivă a aplicaţiei

### *Lecţii învăţate:*

- Planificarea arhitecturii: O structură bine gândită de la început economiseşte timp şi evită refactorizările majore.
- Validarea datelor: Verificările la toate nivelurile (UI, Repository, Database) previn erorile şi îmbunătăţesc stabilitatea.
- UX la fel de important ca funcţionalitatea: Caracteristicile tehnice perfecte sunt inutile fără o interfaţă intuitivă şi prietenoasă.

- Stările Android: Ciclul de viaţă al fragmentelor şi navigarea necesită gestionare atentă pentru a evita crash-urile.
- Gestionarea API-urilor: Serviciile externe pot eşua - sunt necesare timeout-uri, retry logic şi mesaje clare de eroare.

Aceste principii ne-au ghidat spre o aplicaţie stabilă şi o experienţă de dezvoltare mai eficientă.

## **7. Link GIT către codul proiectului**

<https://github.com/Vanessa-unitbv/NutriSense.git>