



**Departamentul Automatică și Informatică Industrială**  
**Facultatea Automatică și Calculatoare**  
**Universitatea POLITEHNICA din București**



## **LUCRARE DE DIPLOMĂ**

### **Aplicație Web pentru gătit și evitarea risipei alimentare**

**Coordonator**

**Conf. Dr. Ing. Alexandra-Suzana Cernian**

**Absolvent**

**Roman Andrada-Liliana**

**2023**

## Cuprins

Introducere.....	5
Prezentarea domeniului lucrării.....	5
Scopul și obiectivele lucrării .....	6
Descrierea conținutului lucrării .....	7
Soluții tehnologice existente în domeniu .....	8
Banca pentru alimente .....	8
Too Good to Go.....	9
LoveFoodHateWaste .....	9
Analiza cerințelor tehnice specifice.....	10
Sondaj de opinie .....	10
Cerințe funcționale .....	12
Cerințe non-funcționale .....	12
Diagrame UML .....	13
Dezvoltarea aplicației .....	16
Mediile de dezvoltare .....	16
Dezvoltarea bazei de date .....	16
Tehnologii utilizate pentru Backend .....	17
Tehnologii utilizate pentru Frontend .....	18
Tehnologia utilizată pentru autentificare.....	20
Explicarea modelului arhitectural.....	20
Implementarea aplicației .....	23
Crearea bazei de date.....	23
Descriere componentelor cheie backend .....	25
Conectarea la baza de date .....	25
Adăugarea serviciului Okta.....	25
Crearea modelelor.....	25
Preluarea tuturor rețetelor .....	26
Preluarea tuturor ingredientelor.....	26
Serviciul comentarii.....	26
Serviciul favorite.....	27
Serviciul donații .....	28

Fișierele de configurare.....	29
Prezentarea interfeței .....	30
Descriere componentelor cheie frontend.....	34
Pagina de căutare a rețetelor.....	34
Pagina de afișare a detaliilor rețetelor .....	37
Pagina de căutare după ingrediente .....	39
Pagina de favorite.....	42
Pagina forumului Relife .....	44
Opțiunea de vizualizare a donațiilor .....	44
Opțiunea de adăugare a unei donații.....	45
Evaluarea lucrării.....	46
Funcționalitatea de căutare a rețetelor .....	46
Funcționalitatea de afișare a detaliilor rețetelor .....	47
Funcționalitatea de căutare după ingrediente .....	47
Funcționalitatea de vizualizare a forumului Relife .....	47
Retrospectiva și Concluziile Lucrării .....	48
Bibliografie.....	49

## Tabel pentru figurile adăugate

Figura 1: Interfața paginii de donare pentru site-ul web bancapentrualimente.ro.....	8
Figura 2: Interfața paginii de căutare rețete pentru site-ul web lovefoodhatewaste.com .....	9
Figura 3: Răspunsul întrebării cu numărul 1 .....	10
Figura 4: Răspunsul întrebării cu numărul 2 .....	11
Figura 5: Răspunsul întrebării cu numărul 3 .....	11
Figura 6: Diagrama Use Case.....	13
Figura 7: Diagrama de activități pentru căutarea rețetelor după ingrediente .....	14
Figura 9: Diagrama de activități pentru adăugarea și vizualizarea donațiilor .....	15
Figura 8: Diagrama de activități pentru vizualizarea și căutarea rețetelor înainte și după autentificare .....	15
Figura 10: Funcționarea arhitecturii .....	22
Figura 11: Tabelele create în baza de date folosind MySQL .....	24
Figura 12: Cod pentru conectarea backendului la baza de date și la serviciul Okta .....	25
Figura 13: Cod pentru implementarea relației One To Many <sup>[2]</sup> .....	25
Figura 14: Pagina Home - Secțiunea numărul 1 .....	30
Figura 15: Pagina Home - Secțiunea „Despre noi” .....	31
Figura 16: Pagina Home – A doua parte a secțiunii „Despre noi” .....	31
Figura 17: Pagina de Contact .....	32
Figura 18: Pagina de logare și opțiunea „Parolă uitată” .....	32
Figura 19: Pagina Menu de întâmpinare a utilizatorului logat .....	33
Figura 20: Opțiunea „Despre Risipă” .....	33
Figura 21: Opțiunea „Caută rețete” .....	34
Figura 22: Pagina ce conține detaliile unei rețete.....	37
Figura 23: Opțiunea „Refood’s Feature” .....	39
Figura 24: Rezultatul căutării după ingrediente inexistente .....	40
Figura 25: Opțiunea „Rețete Favorite” .....	43
Figura 26: Pagina când utilizatorul nu are nicio favorită .....	43
Figura 27: Opțiunea „Vezi Forum” .....	44
Figura 28: Pagina când nu există donații.....	45
Figura 29: Opțiunea „Adaugă Anunț” .....	46

## Introducere

În prezenta lucrare, am proiectat și dezvoltat un sistem software și anume o aplicație web pentru gătit menită să reducă risipa de mâncare. Întregul proces de implementare a platformei va fi detaliat în capitolele următoare cu scopul observării posibilelor greșeli în etapele de planificare sau dezvoltare a proiectului, eventuale limitări ale aplicației rezultate, precum și impactul și contribuția aplicației în domeniul risipei alimentare.

### Prezentarea domeniului lucrării

Întreaga lucrare se concentrează pe ideea de risipă alimentară, iar pentru început este necesar să cunoaștem sensul acestui termen. Acesta apare în contextul în care resurse alimentare aflate în stare bună pentru consum sunt irosite sau aruncate la gunoi în mod nejustificat. Când se produce irosirea alimentelor? Procesul de risipă alimentară poate apărea pe tot parcursul vieții unui produs, de la producerea și procesarea acestuia până la distribuirea și consumul respectivului aliment.<sup>[4]</sup>

Statisticile anuale efectuate la nivel global de organizația internațională FAO prin Programul pentru Mediu al Națiunilor Unite(UNEP) au arătat, în ultimele decenii, faptul că aproximativ 30-40% din alimentele produse în întreaga lume ajung să fie aruncate la gunoi, majoritatea aflându-se într-o stare bună de consum.<sup>[20]</sup> Potrivit acestor estimări, din procentul întreg al risipei la nivel mondial, în jur de o treime are loc în timpul ultimei etape din ciclul vieții unui aliment, și anume consumul individual.

Un factor care ar trebui să transforme problema risipei de mâncare într-o preocupare globală reprezintă impactul negativ al acesteia asupra mediului. În această categorie întâlnim, consumul de resurse inutil, cum ar fi apa și energia, folosite la producerea și distribuirea alimentelor risipite sau poluarea mediului prin cantitățile semnificative de deșeuri ce conduc la degradarea calității solului și a apelor și nu în ultimul rând emisiile de gaze cu efect de seră eliberate în timpul descompunerii alimentelor ce susțin agravarea încălzirii globale. Un alt raport al FAO arată ca risipa alimentară reprezintă 8-10% din totalul de emisii globale de gaze cu efect de seră.<sup>[20]</sup>

Risipa alimentară a existat încă de când tehnologiile de conservare erau extrem de limitate, însă s-a accentuat odată cu perioada Revoluției Industriale când au fost introduse mașinăriile industriale ce au condus către producții în masă nenecesare. A urmat apoi o epocă în care oamenii s-au deprins cu aruncarea alimentelor ce nu corespundeau unui anumit standard estetic, iar acest obicei a rămas până în perioada contemporană. În prezent, numeroase alimente bune pentru consum ajung să fie refuzate de cumpărători sau aruncate mai târziu pentru că au un aspect stricat.

În toată lumea, mai multe instituții guvernamentale sau organizații internaționale au adoptat măsuri de scădere a risipei de mâncare prin campanii de informare sau diverse alte metode. Spre exemplu, MADR a inaugurat ziua de 29 septembrie ca fiind Ziua Internațională de

conștientizare a risipei.<sup>[12]</sup> Franța a delegat o lege ce interzice supermarketelor să arunce mâncarea, aceștia trebuind să doneze alimentele rămase la sfârșitul zilei sau cu data de expirare apropiată către centre speciale sau organizații de caritate.<sup>[1]</sup>

În ciuda eforturilor și măsurilor adoptate la nivel global, problema risipei se restrânge în mare parte și la nivel individual și se poate combate doar prin adoptarea unei conduite moderate de consum. Fiecare consumator ar trebui să-și planifice cumpărăturile pentru a se asigura că nu achiziționează mai mult decât poate găti și mânca, să-și gestioneze corect și să utilizeze complet mâncarea achiziționată și, dacă este cazul, să aibă în vedere donarea surplusului de alimente neutilizate.

Tinând cont de toate cele afirmate mai sus, putem spune că risipa de mâncare este o problemă reală și actuală care impactează atât mediul înconjurător, cât și economia la nivel global. Aceasta poate fi cauzată de diverși factori printre care regăsim o infrastructură precară ce poate conduce la o supraproducție, la o stocare inadecvată, la un transport deficitar, la conservarea greșită sau la o reciclare insuficientă a alimentelor și nu în ultimul rând comportamentul iresponsabil al fiecărui consumator în parte.

Risipa produsă în timpul primelor trei etape din ciclul de viață al unui produs nu poate fi redusă decât de producătorii și distribuitorii de alimente, fie prin propria dorință, fie prin cerințele impuse de fiecare guvern în parte. În schimb, noi cetățenii, prin educarea noastră și a celor din jur, putem combate propria irosire de alimente. De aceea, aplicația implementată dorește să vină în sprijinul consumatorilor pentru a îi ghida către combaterea risipei alimentare din propriile locuințe.

### Scopul și obiectivele lucrării

Aplicația web creată în această lucrare este de tip multi-page application(MPA) și poate fi folosită prin deținerea unui cont de utilizator, dar pune o funcționalitate și la dispoziția userilor neautentificați. Am ales această metodă de informare și evitare a risipei alimentare ținând cont de impactul semnificativ pe care îl poate avea internetul asupra oamenilor în zilele de astăzi.

Scopul principal al aplicației este de a reduce problema contemporană referitoare la risipa alimentară. Măsura implementată în această lucrare se referă la o platformă web care permite utilizatorilor să-și creeze propriile metode de gestionare eficientă a alimentelor de care dispun sau să-și doneze mâncarea nefolosită.

Cele două metode abordate pentru scăderea risipei se referă strict la risipa din timpul consumului, sarcină care ne revine fiecăruia în parte și de care suntem responsabili doar noi. Oamenii care doresc să adopte un comportament potrivit pentru un consum moderat pot face asta cu ajutorul aplicației denumite „Refood”.

Funcționalitatea principală a aplicației constă în sugerarea unor rețete ce conțin de la toate până la unul dintr-un număr nelimitat de ingrediente ce pot fi introduse de utilizator în aplicație.

Obiectului este ca oamenii să gătească rețete cu cât mai multe ingrediente pe care le dețin deja, evitând cumpărarea de noi alimente și degradarea proviziilor curente.

Cu ajutorul altei funcționalități la fel de importante, utilizatorii, fie ca vorbim de o persoană fizică sau un deținător de restaurant, cantină sau supermarket etc, pot alege să-și doneze mâncarea cu ajutorul forumului ce poartă numele „Relife”. Obiectivul constă în reducerea aruncării cantităților masive de mâncare care nu este expirată de către serviciile de alimentație publică și redirectionarea acestora către alegerea de a dona mâncarea rămasă către centre sau asociații caritabile sau chiar către persoane fizice în nevoie. De asemenea, se dorește și contribuția individuală a persoanelor în procesul de donare în funcție de propriile posibilități.

Pe deasupra, aplicația vine și în ajutorul oamenilor care se consideră neînspirati când se află în fața întrebării frecvente: Ce să gătesc? Pe platformă, este pus la dispoziție un meniu vast din care utilizatorii pot alege ce rețetă vor găti.

Un ultim obiectiv al aplicației implementat cu ajutorul unei pagini denumite „Despre Risipă”, ce este dedicată informării utilizatorilor, este oferirea de sfaturi și tehnici de urmat cu privire la măsurile individuale pe care le pot adopta pentru a contribui la reducerea risipei alimentare din propriile locuințe.

## Descrierea conținutului lucrării

În cadrul acestei secțiuni, vor fi menționate capitolele ce urmează a fi abordate pe parcursul lucrării pentru a putea explica pe scurt conținutul acestora și rezultatul așteptat în urma dezvoltării lor.

Capitolul denumit „Soluții tehnologice existente în domeniu” va detalia câteva produse software existente pe piață, similare din punct de vedere al funcționalităților cu aplicația dezvoltată în această lucrare, cu scopul de a observa prin ce se diferențiază față de restul sau ce lipsuri are comparativ cu exemplele alese.

Capitolul următor de „Analiza cerințelor tehnice specifice” va consta în observarea rezultatelor unui sondaj de opinie cu scopul stabilirii și eventual, ajustării cerințelor necesare pentru dezvoltarea aplicației.

Capitolul ce poartă numele „Dezvoltarea aplicației” conține detalii despre planificarea procesului de implementare a aplicației, cum ar fi IDE-urile, limbajele de programare folosite și tehnologiile de backend și frontend.

Capitolul numit „Implementarea aplicației” explică cum au fost create elementele aplicației referindu-se la codul asociat acestora. În cadrul acestui capitol, este descris modul de implementare a celor mai importante componente și este prezentată interfața platformei care servește drept ghid de utilizare pentru un posibil user.

Capitolul următor „Evaluarea lucrării” are ca scop determinarea diferențelor între cerințele planificate specificate în capitolul cu numărul trei și funcționalitățile obținute în urma implementării. Pe baza acestor diferențe se va face identificarea limitărilor aplicației și se vor căuta metode de actualizare care vor fi adăugate în posibile dezvoltări viitoare cu scopul îmbunătățirii.

Capitolul final denumit „Retrospectiva și Concluziile Lucrării” accentuează ideile cele mai importante discutate de-a lungul întregii lucrări concentrându-se asupra impactului și contribuția pe care le are aplicația în domeniul reducerii risipei alimentare. Capitolul urmărește descrierea aplicației în ansamblu și concluzionarea acestei lucrări.

## Soluții tehnologice existente în domeniu

În urma unor analize efectuate, am observat că pe piața actuală există numeroase aplicații web sau mobile dedicate evitării risipei alimentare. În acest capitol, am ales cele mai semnificative și populare soluții tehnologice cu scopul de a le analiza.

### Banca pentru alimente

Această aplicație web a fost creată de români și este o platformă folosită doar pentru donarea de alimente după cum sugerează și numele acesteia, însă se pot dona și bunuri, servicii sau bani în funcție de alegerea utilizatorului. Donarea implică atât persoanele fizice, cât și companiile și ONG-urile.<sup>[5]</sup>

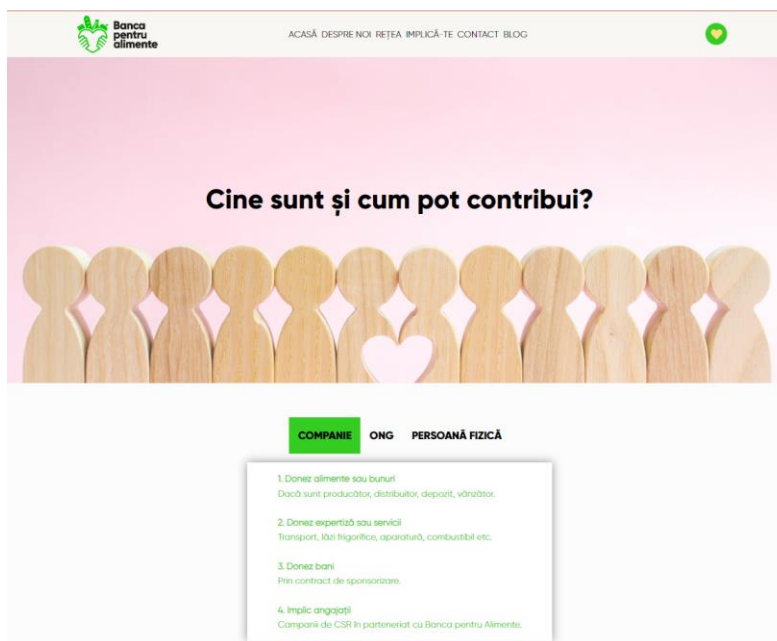


Figura 1: Interfața paginii de donare pentru site-ul web bancapentrualimente.ro



## Too Good to Go

Este o aplicație web și mobilă disponibilă în mai multe țări din Europa, inclusiv România care le permite utilizatorilor să achiziționeze la un preț redus diverse alimente care nu au fost vândute până la finalul zilei și care ar fi fost aruncate în mod normal din restaurantele, cofetăriile sau magazinele din apropierea lor. [19]

## LoveFoodHateWaste

Aplicația în cauză are atât o versiune web, cât și mobilă și este disponibilă în mai multe țări. În prezent, nu există o versiune adaptată utilizatorilor din România, dar este accesibilă și se poate folosi în limba engleză. Cu ajutorul acesteia, utilizatorii primesc numeroase idei de rețete care pot avea diverse filtre bazate pe cerințele alimentare ale utilizatorului (vegetarian, fără gluten, fără lactate, etc.). Aplicația încurajează reducerea risipei alimentare prin sugerarea unor rețete predefinite din posibilele resturi de alimente deținute. [11]

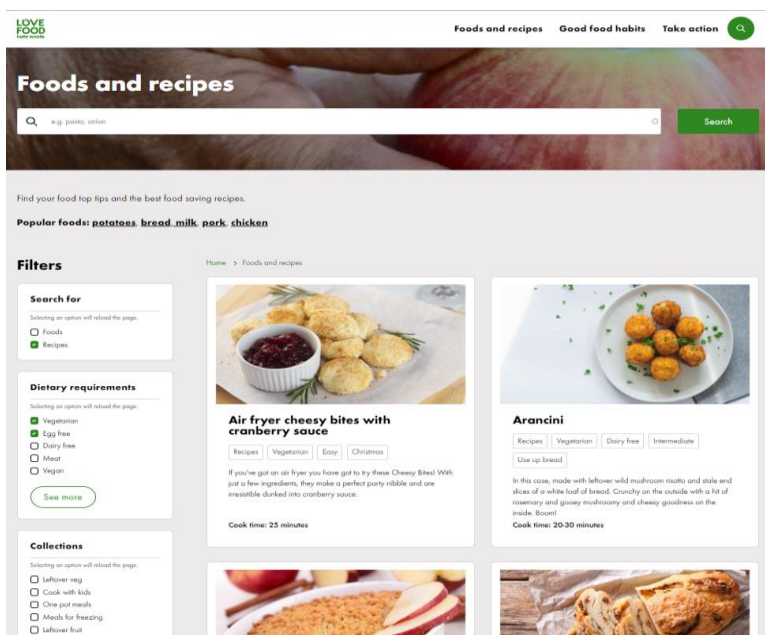


Figura 2: Interfața paginii de căutare rețete pentru site-ul web lovefoodhatewaste.com

În urma acestei cercetări a pieței, putem observa că există aplicații pentru gătit sau pentru nutriție care sugerează rețete ce pot fi filtrate în funcție de preferințele utilizatorului. Aceste aplicații conțin și idei de rețete ce refolosec anumite alimente sau care se bazează pe reciclarea unor preparate culinare. De asemenea, există și aplicații care se axează doar pe facilitarea procesului de donare prin folosirea internetului cu scopul conectării beneficiarilor cu donatorii.

Ținând cont de cele afirmate anterior și de soluțiile existente deja în domeniu, prezentate în acest capitol, se poate constata că, la momentul actual, nu există o aplicație web care să

înglobeze toate funcționalitățile implementate de fiecare aplicație în parte. Din acest motiv, aplicația creată în această lucrare aduce un plus de inovație prin întrunirea atât a caracteristicilor unei platforme de donare și de informare asupra problemei risipei de mâncare, cât și a funcționalităților unui program menit să îndrume utilizatorul către o gestionare corectă a alimentelor prin diverse metode.

## Analiza cerințelor tehnice specifice

În procesul de planificare a unui sistem software este necesară etapa de definire a unor cerințe minimale ce trebuiesc îndeplinite de aplicația web în cauză, pentru a se asigura implementarea funcționalităților dorite și respectarea unor standarde impuse de performanță și de calitate pentru site-uri.

În cadrul acestui capitol, vor fi enumerate o serie de specificații necesare ce vor fi divizate în două categorii în funcție de scopul propus și anume: cerințe funcționale și cerințe non-funcționale. La final, pe baza cerințelor stabilite se vor crea diagramele UML ale aplicației.

## Sondaj de opinie

Pentru această secțiune a lucrării de licență am creat și distribuit către colegi și cunoștințe un sondaj de opinie. Sondajul are un total de 40 de răspunsuri și conține 3 întrebări menite să faciliteze procesul de planificare a specificațiilor tehnice dorite. Prin obținerea acestui feedback chiar de la posibili utilizatori ai aplicației asigurăm succesul aplicației prin adaptarea cerințelor pentru a corespunde cu dorințele clienților.

Ați întâlnit și folosit vreodată aplicații pentru gătit care să sugereze rețete în timp real bazat pe ingredientele introduse de dumneavoastră?

40 de răspunsuri

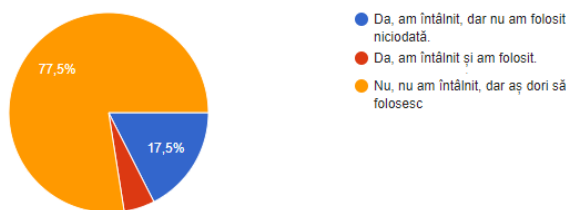


Figura 3: Răspunsul întrebării cu numărul 1

În urma rezultatului la această prima întrebare din sondaj, putem constata că numărul aplicațiilor care implementează o funcționalitate de căutare rețete în timp real în funcție de ingrediente alese este redus. Totodată, trebuie să luăm în considerare și faptul că majoritatea ar încerca o astfel de aplicație, deci cu siguranță si-ar câștiga popularitatea în rândul publicului.

Considerați ca o astfel de aplicație v-ar ajuta să vă gestionați și să folosiți complet alimentele pe care le dețineți deja pentru a nu achiziționa altele?

40 de răspunsuri



Figura 4: Răspunsul întrebării cu numărul 2

Pe baza acestui rezultat din sondaj, deducem că toți subiecții consideră ca o aplicație cu o astfel de funcționalitate ar ajuta la reducerea cumpărăturilor necalculate. Putem observa că nimeni nu-și planifică un meniu săptămânal pe baza căruia să achiziționeze strict provizii pentru uzul prevăzut.

Ați donat vreodată mâncare?

40 de răspunsuri

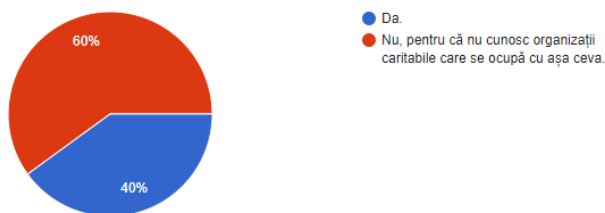


Figura 5: Răspunsul întrebării cu numărul 3

În partea de final a sondajului se demonstrează faptul că unul dintre motivele pentru care oamenii nu donează mâncare este pentru că nu știu de existența organizațiilor care primesc acest tip de donații. Deși aceste organizații caritabile există, ele nu sunt foarte cunoscute. Prin urmare, se constată că o platformă online de donații este o modalitate potrivită de informare cu privire la centrele de donații de alimente existente și de gestionare a anunțurilor cu donații realizate de utilizatori.

Rezultatul întregului chestionar demonstrează că adăugarea funcționalității de căutare în timp real după ingrediente trebuie inclusă în cerințele funcționale și totodată, că procesul de donare ar fi mai ușor de gestionat în mediul online, deci implementarea unui astfel de serviciu pe o platformă ar asigura creșterea numărului de donații alimentare.

## Cerințe funcționale

Acest tip de cerințe descriu modul de funcționare și comportamentul aplicației. Prin specificarea lor, putem stabili ce fel de acțiuni pot realiza utilizatorii și ce fel de reacțiuni trebuie să aibă sistemul software în diverse situații.

În urma alegerii unui set de cerințe minimal pe baza aplicațiilor deja existente în domeniu și a cunoașterii modelului general de dezvoltare a site-urilor web, a urmat adăugarea unor funcționalități specifice. Rezultatul chestionarului demonstrează faptul că acestea trebuie integrate pentru obținerea de plus valoare față de referințe. Funcționalitățile specifice vor fi afișate și detaliate în cadrul capitolului de implementare.

Se regăsesc următoarele cerințe:

- afișarea unei pagini de Home la accesarea aplicației;
- posibilitatea accesării rețetelor și fără autentificare printr-un buton care redirecționează utilizatorul către pagina de vizualizare a rețetelor;
- adăugarea opțiunii de autentificare; aceasta se face prin conturi predefinite;
- implementarea unui sidebar menu pentru navigarea în aplicație ce identifică utilizatorul curent logat;
- afișarea unui meniu de rețete structurat pe mai multe pagini;
- capacitatea de filtrare a rețetelor în funcție de categoria lor;
- posibilitatea observării detaliilor despre rețete și a pașilor de realizare;
- adăugarea opțiunii de bifare a ingredientelor deținute la realizarea unei rețete;
- vizualizarea comentariilor pentru fiecare rețetă în parte;
- posibilitatea de a adăuga nou comentariu;
- implementarea unui feedback bazat pe stele ce poate fi oferit de user
- adăugarea căutării rețetelor după cuvinte conținute în titlu;
- abilitatea de a scrie ingredientele deținute și de a căuta rețete după acestea în timp real;
- posibilitatea de a sugera rețete bazate pe inputul primit de la user;
- implementarea unei paginii de favorite specifice preferințelor fiecărui utilizator;
- capacitatea de a adăuga sau șterge rețete de la favorite;
- implementarea forumului pentru donarea mâncării;
- posibilitatea de adăugare a unui nou anunț și de ștergere a anunțurilor proprii;
- accesarea unei pagini de sfaturi pentru evitarea risipei;
- implementarea metodei de deconectare;

## Cerințe non-funcționale

Specificarea cerințelor non-funcționale se concentrează pe asigurarea securității aplicației, optimizarea aspectului interfeței cu utilizatorul prin respectarea normelor impuse de domeniu și definirea calităților platformei precum performanța, adaptabilitatea și robustețea.

Printre acestea se enumeră:

- implementarea unei aplicații scalabile la diferite dimensiuni de ecran;
- crearea unei interfețe sugestive și ușor de folosit;
- alegerea unor metode rapide de căutare și afișare a rețetelor;
- afișarea unui loading spinner în cazul așteptării unui răspuns de la backend;
- implementarea unor mesaje de eroare în aplicație pentru momentele rare de lipsă a răspunsului;
- folosirea minim a două fonturi diferite pentru tot textul afișat în interfață
- adăugarea unor design-uri inedite și captivante
- folosirea serviciului de securitate oferit de Okta
- inspirarea încrederii utilizatorilor prin culorile și nuanțele din interfață

## Diagrame UML

Cerințele funcționale stabilite în secțiunea anterioară alcătuiesc activitățile principale existente în cadrul aplicației care au fost folosite ulterior la crearea diagramei cazurilor de utilizare.

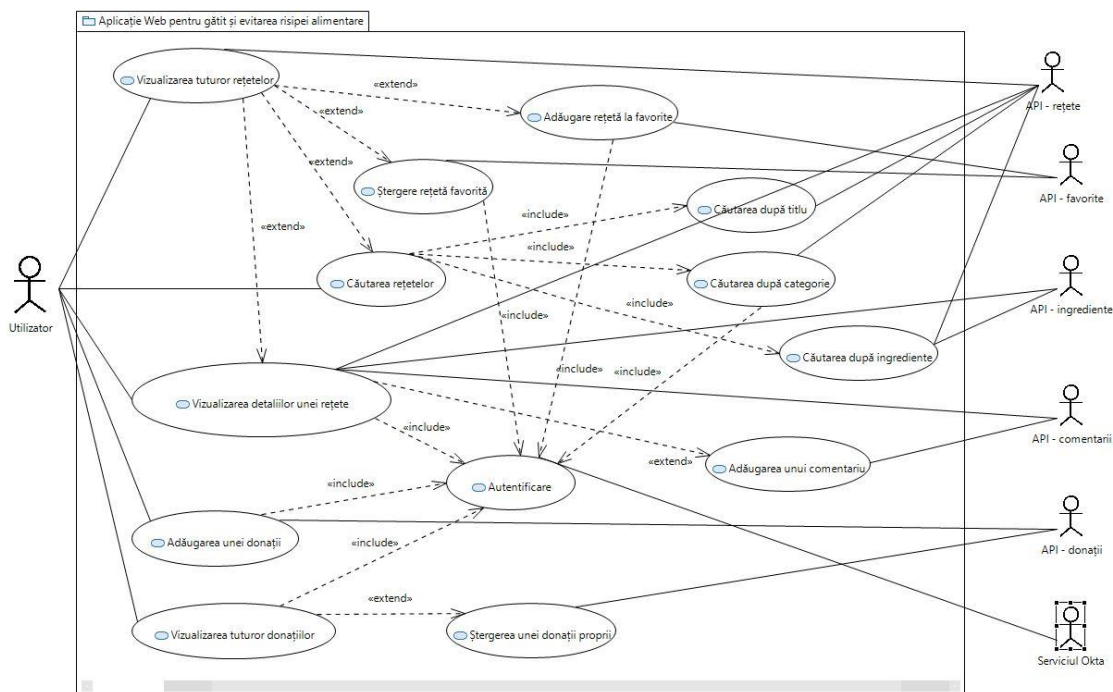


Figura 6: Diagrama Use Case

Acest tip de diagramă oferă o vizualizare structurată asupra tuturor modurilor de utilizare. În cadrul acesteia, se regăsesc funcționalitățile principale: vizualizarea și căutarea rețetelor, accesarea detaliilor unei rețete și vizualizarea sau adăugarea unei donații. Dintre acestea, doar vizualizarea rețetelor și căutarea după categorie sau titlu se pot folosi de către utilizatorii

neautentificați. Prin urmare, restul funcționalităților necesită efectuarea login-ului și apelarea serviciului de autentificare implementat cu ajutorul actorului Okta. De asemenea, se evidențiază și serviciile care trebuie apelate pentru realizarea acțiunilor. Actorii din dreapta cu denumirea API reprezintă cererile care trebuie formulate către baza de date pentru preluarea datelor din tabela corespunzătoare denumirii. Spre exemplu, opțiunea de vizualizare a detaliilor unei rețete implică accesarea informațiilor despre aceasta și a ingredientelor corespunzătoare, precum și preluarea tuturor comentariilor aferente rețetei.

În continuare, am creat diagramele de activități pentru cele mai semnificative funcționalități ale aplicației și anume, căutarea rețetelor după ingrediente introduse de la tastatura de utilizator, vizualizarea rețetelor fără autentificare și după login care s-au creat în două diagrame separate și adăugarea și vizualizarea postărilor cu donații. Cu ajutorul acestor reprezentări grafice, putem observa fluxul de activități și decizii executate în timpul desfășurării tuturor proceselor descrise.

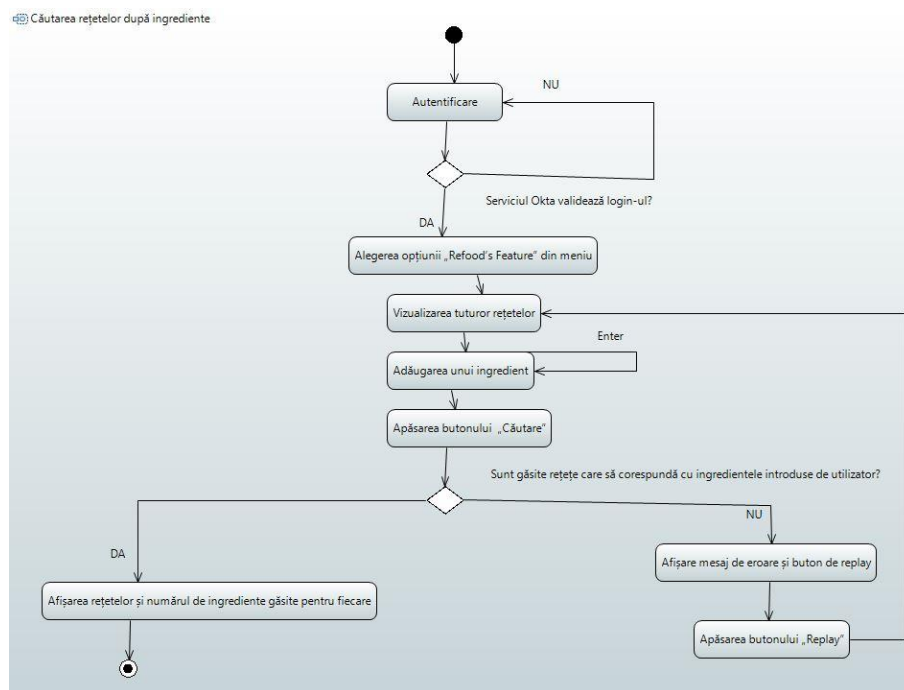


Figura 7: Diagrama de activități pentru căutarea rețetelor după ingrediente

În figura anterioră, se ilustrează modul de funcționare al procesului de căutare a rețetelor după ingredientele introduse de la tastatură de utilizator. Putem remarca faptul că această funcționalitate poate fi folosită doar de utilizatorii autentificați. După alegerea opțiunii corespunzătoare din meniul lateral, user-ul poate observa toate rețetele afișate în pagină. Apoi, se așteaptă ca acesta să adauge ingredientele de la tastatură, finalizând cuvinte prin apăsarea tastei enter. Această activitate se poate executa ori de câte ori dorește utilizatorul. După adăugarea tuturor ingredientelor dorite, este necesară apăsarea butonului de căutare care pornește algoritmul de căutare. Se verifică dacă există un rezultat în urma căutării și, în caz afirmativ, se afișează rețetele care conțin ingredientele după care s-a executat căutarea și numărul total de apariții al ingredientelor introduse printre ingredientele rețetei. În caz contrar,

se va afișa un mesaj de eroare și un buton de replay care la apăsare va restarta funcționalitatea prin accesarea stării de început de vizualizare a tuturor rețetelor.

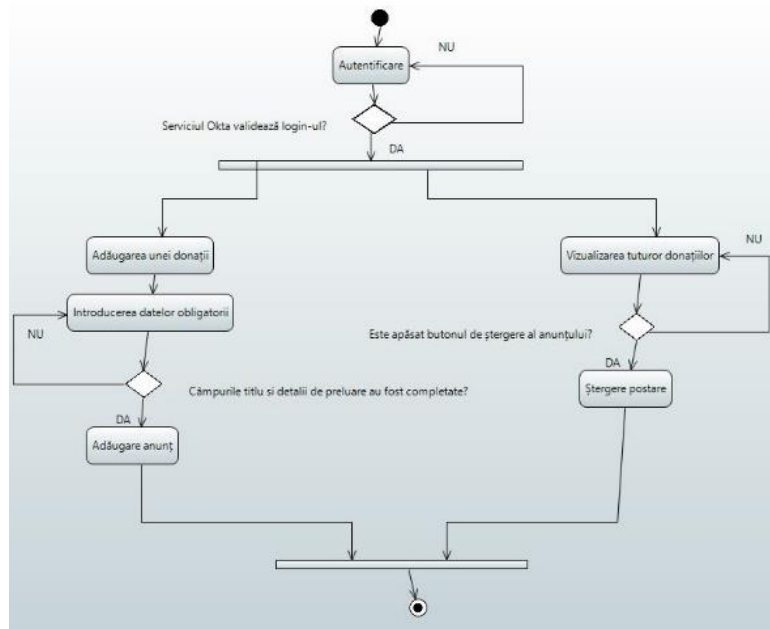
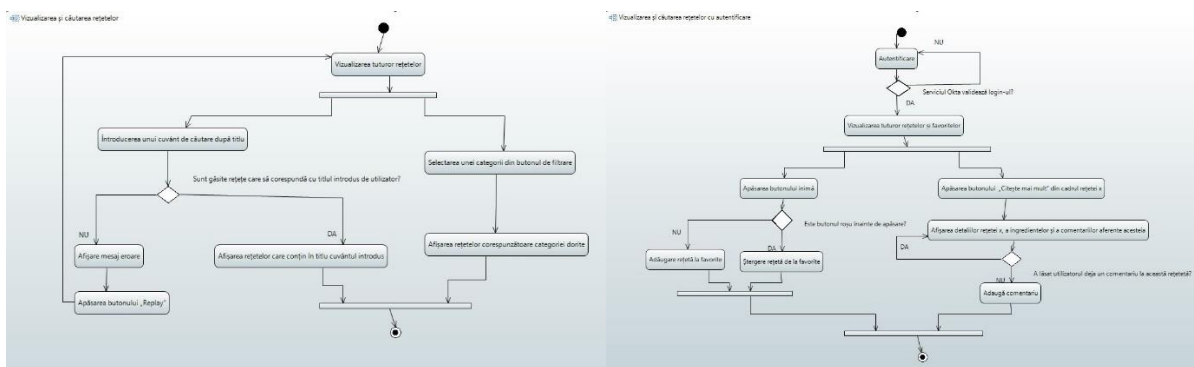


Figura 8: Diagrama de activități pentru adăugarea și vizualizarea donațiilor

În figura cu numărul 9, este reprezentată diagrama de activități pentru procesul de adăugare și vizualizare a tuturor donațiilor, accesat prin alegerea opțiunii forum Relife din cadrul meniului lateral. Toate funcționalitățile stabilite și descrise în cadrul acestui capitol pot fi accesate și folosite doar de utilizatorii logați, deoarece conțin date specifice contului. Există o singură excepție de la aceste funcționalități și anume, vizualizarea și căutarea după titlu și categorie a rețetelor care poate fi observată în imaginea de mai jos, alături de varianta procesului când utilizatorul este conectat la cont.



În figura anterioară din partea stângă, se evidențiază diagrama activităților pentru funcționalitatea de vizualizare și căutare a rețetelor accesată de utilizatori neautentificați. În



partea alăturată, se remarcă aceeași funcționalitate însă pentru utilizatorii logați care accesează pagina “Refood Feature” din meniul lateral al user-ului. Cel de-al doilea proces înglobează de asemenea și fluxul de acțiuni și decizii inclus în prima diagramă. În cadrul primei diagrame, sunt afișate toate rețetele și se așteaptă introducerea unui titlu și apăsarea butonului de căutare sau selectarea unei categorii de rețete de observat. Pentru opțiunea de căutare după titlu s-a luat în calcul scenariul în care nicio rețetă nu corespunde căutării. În acest caz, se va afișa un mesaj corespunzător și se va aștepta apăsarea butonului replay pentru revenirea la starea inițială dinaintea căutării. În a doua diagramă, în plus față de activitățile decrise anterior, se pot observa favoritele utilizatorului evidențiate prin butoanele cu inimi roșii. La apăsarea butonului, în funcție de culoarea prezentă, se șterge sau adaugă rețeta la favorite și se actualizează culoarea inimii corespunzătoare acțiunii. De asemenea, utilizatorii pot accesa pagina cu detaliile rețetei și comentariile aferente acesteia prin apăsarea butonului “Citește mai mult”. User-ul curent poate adăuga un feedback dacă nu a realizat deja această acțiune pentru rețeta curentă.

Cu ajutorul acestor capitole, am discutat și stabilit cerințele pe care aplicația le va îndeplini. În continuare, urmează un alt pas al procesului de dezvoltare al unui software care implică înțelegerea arhitecturilor web, alegerea mediilor IDE pentru frontend și backend și nu în ultimul rând discutarea și selectarea celor mai potrivite tehnologii pentru dezvoltarea și implementarea aplicației.

## Dezvoltarea aplicației

Dezvoltarea aplicației a presupus cunoașterea opțiunilor existente de medii de dezvoltare și limbaje de programare, găsirea unor framework-uri potrivite cu funcționalitățile alese, alegerea unui sistem de gestiune a bazelor de date, înțelegerea modului de comunicare între frontend, backend și baza de date, crearea proiectelor de început și aprofundarea cunoștințelor.

### Mediile de dezvoltare

Partea de backend a fost scrisă cu ajutorul mediului de dezvoltare IntelliJ Community Edition luând în considerare suportul oferit pentru dezvoltarea aplicațiilor Java și integrarea framework-urilor necesare proiectului în cauză, precum Spring Boot.

Partea de frontend a fost realizată folosind editorul de cod Visual Studio Code. În urma experienței personale cu acest editor am ales să optez pentru această variantă luând în considerare faptul că acest mediu IDE este cel mai des utilizat cu scopul de dezvoltare a părții de frontend a unei aplicații pentru că oferă suport integrat pentru React.

### Dezvoltarea bazei de date

Pentru dezvoltarea și gestionarea bazei de date relaționale am ales drept sistem de gestiune MySQL Database datorită familiarizării cu această aplicație și din motive de compatibilitate cu tehnologia de backend aleasă.



Pentru un moment, în urma specificării cerințelor aplicației am considerat că vor trebui implementate următoarele: o tabelă pentru stocarea rețetelor, o altă tabelă individuală pentru ingredientele rețetelor( în cadrul capitolelor următoare se va specifica legătura dintre aceste două tabele) și trei tabele independente pentru comentarii, favorite și anunțurile de donații. Mai multe detalii despre baza de date creată și procesul de integrare a tabelelor și înregistrărilor se vor găsi în capitolul ce prezintă pe larg detaliile de implementare a aplicației.

## Tehnologii utilizate pentru Backend

Spring Boot este un framework open-source care asigură gestionarea automată a dependențelor și simplificarea modului de configurare prin fișiere precum pom.xml sau application.properties care pot fi modificate de dezvoltatori în scopul obținerii unei configurații dorite.<sup>[21]</sup>

În cadrul proiectului, s-au folosit adnotările puse la dispoziție de Spring Boot pentru crearea modelelor preluate din baza de date, pentru definirea claselor de configurare și pentru crearea serviciilor RESTful.

Alte două avantaje majore oferite de Spring Boot sunt Spring Data JPA și Spring Data REST care simplifică crearea serviciilor RESTful. Prima oferă suport pentru lucrul cu bazele de date relaționale și include repository-uri JPA. Astfel endpoint-urile pentru operațiile CRUD( creare, citire, modificare și ștergere – metode HTTP: POST, GET, PUT, DELETE) ale entităților definite sunt create în mod automat și pot fi personalizate prin suprascriere. De asemenea, integrează și metode pentru filtrarea și paginarea datelor primite de la endpoint-uri. Spring Data REST este o extensie a Spring Data JPA care realizează generarea automată de servicii predefinite RESTful bazate pe repository-urile JPA deja existente.<sup>[8]</sup>

JDK înglobează o gamă de utilitare necesare pentru crearea, testarea, gestionarea și rularea aplicațiilor scrise folosind limbajul de programare Java.<sup>[8]</sup>

Maven este folosit pentru simplificarea modului de gestionare și instalare a dependențelor externe ale proiectului regăsite în interiorul fișierului de configurare numit pom.xml.<sup>[21]</sup>

Lombok este o librărie ce pune la dispoziția programatorului getters, setters și constructori predefiniți ce pot fi folosiți prin adnotările corespunzătoare.<sup>[8]</sup> Această tehnologie este folosită pentru evitarea scrierii codului de tip boilerplate și pentru optimizarea timpului necesar implementării.

Mysql-connector-java este o tehnologie folosită pentru a face posibilă conectarea și comunicarea aplicațiilor de tip Java cu bazele de date create în MySQL pentru a prelua datele necesare.<sup>[14]</sup>

Inițializarea proiectului a fost realizată folosind Spring Initializr, iar structura folderului `com.springbootapp.refoodbe` al acestuia conține următoarele elemente sub formă de foldere:

- `conf` – folder în care există două fișiere de configurare create folosind adnotarea `@Configuration`. Acestea specifică rutele securizate sau interzic anumite metode HTTP pentru anumite clase;
- `control` – director ce include fișiere de tip REST controllere care gestionează endpoint-urile cu ajutorul cărora se asigură primirea cererilor HTTP și furnizarea răspunsurilor către client prin adnotarea `@CrossOrigin` care conține adresa de plecare a cererii.
- `models` – aici se regăsesc clasele care creează entitățile ce stochează informațiile obținute în urma mapării modelelor pe tabelele din baza de date cu ajutorul adnotărilor `@Entity`, `@Table` și `@Data`;
- `repos` – conține interfețele responsabile cu accesarea bazei de date. Acestea extind `JpaRepository` și adaugă metode denumite folosind convențiile puse la dispoziție de Spring Data JPA.
- `requests` –director ce include fișiere folosite pentru trimiterea de date între aplicație și client. Sunt folosite fie în services, fie în controllere pentru a manevra datele cerute și primite.
- `services` – folder unde se regăsesc fișiere care gestionează partea de logică prin implementarea funcționalităților specifice ale aplicației. Sunt folosite adnotările `@Service` și `@Transactional`. Acestea pot să interacționeze cu controllerele pentru a accesa datele obținute în urma procesării cererilor cu scopul de a le prelucra într-un mod diferit.

## Tehnologii utilizate pentru Frontend

React este o bibliotecă Javascript pentru construirea de aplicații web interactive și reactive. Arhitectura acestui framework se bazează pe conceptul de componente, ceea ce înseamnă că pentru gestionarea lucrului cu interfețe utilizator(UI), se împarte structura în mai multe componente, fiecare reprezentând o bucată izolată de cod. Este folosit un flux de date unidirecțional, transmis de sus în jos de la componentele părinte către copii. Fiecare componentă are propriul scop și se ocupă cu randarea HTML-ului pentru partea sa corespunzătoare din aplicație.<sup>[16]</sup>

Componentele pot avea o stare(state) și pot afișa codul HTML cu ajutorul JSX. Acesta este o extensie a limbajului Javascript prin care se pot combina limbajele de tip markup, HTML sau XML, cu Javascript. Când starea unei componente este modificată, React actualizează doar acele elemente corespunzătoare schimbării din DOM.<sup>[10,16]</sup>

Am ales folosirea acestei biblioteci pentru suportul oferit pentru site-urile de tip responsive și pentru integrarea framework-urilor moderne precum Bootstrap. În ceea ce privește crearea proiectului, React pune la dispoziție un tool în command-line pentru generarea unor starter files cu scopul de facilita procesul.

Rularea aplicațiilor se face cu ajutorul NPM, un manager de pachete care vine instalat odată cu Node.js și cu ajutorul căruia se pot instala și gestiona module javascript.<sup>[15]</sup> Pachetele NPM folosesc hot reloads, caracteristică specifică React, care permite fluidizarea procesului de dezvoltare prin actualizarea în timp real în browser a serverului ce permite vizualizarea modificărilor făcute fără repornirea serverului sau reîmprospătarea manuală a paginii.<sup>[17,7]</sup>

Structura proiectului se împarte în trei foldere principale:

- node\_modules - acest director conține toate dependențele proiectului; în urma comenzii npm start sunt instalate automat în acest fișier toate pachetele externe necesare funcționării aplicației;
- public - include:
  - favicon-ul aplicației;
  - index.html – fișier ce conține HTML-ul și DOM-ul ;
  - manifest.json – oferă informații despre proiect;
  - robots.txt – transmite instrucțiuni roboților de căutare;

Acestea sunt fișiere create automat la executarea instrucțiunii npm de creare a unui proiect;

- src- include:
  - Images – director ce conține toate pozele din aplicație;
  - auth – unde regăsim 3 fișiere de configurare a autentificării;
  - models – folder pentru transmiterea entităților către backend(fișiere .ts);
  - layouts – director care conține toate elementele interfeței (fișiere .tsx);
  - App.css, App.js – fișiere ce cuprind CSS-ul și Javascript-ul global;
  - App.tsx – fișier care conține toate rutele existente în aplicație

Pentru dezvoltarea React, am creat fișiere cu extensia .tsx care permit combinarea limbajului de programare TypeScript, superset al JavaScript-ului, cu sintaxa JSX. TypeScript-ul este un limbaj open-source ce sprijină programarea orientată pe obiect, adică lucrul cu clase, obiecte, moșteniri, interfețe, etc. și care îmbunătățește timpul de implementare și executare al unei aplicații prin detectarea erorilor în timpul dezvoltării.<sup>[9]</sup>

CSS este un limbaj de stilizare folosit pentru modificarea și personalizarea vizuală a interfeței aplicației. Permite asocierea stilurilor unui element sau a unei grupări de elemente care au aplicată o clasă cu același nume. Cu ajutorul acestuia, se pot specifica culorile, font-urile, dimensiunile elementelor sau a textului și multe alte aspecte ce țin de conținut.<sup>[3]</sup>

Bootstrap este un framework ce conține numeroase clase predefinite de CSS și Javascript care pot fi folosite pentru a ușura procesul de dezvoltare, deoarece programatorii pot opta pentru alegerea unor stiluri și comportamente preconfigurate, în loc să fie nevoiți să le creeze de la zero.<sup>[6]</sup> Sunt atașate la className-ul elementelor pentru a prelua caracteristicile specifice bootstrap-ului asociat.

## Tehnologia utilizată pentru autentificare

Pentru această aplicație, am ales pentru implementarea funcționalității de conectare/authentificare a unui utilizator, platforma de administrare a identității și a accesului (Identity and Access Management) pusă la dispoziție de Okta.

Okta este serviciu IAM de tipul third party administrator care presupune externalizarea procesului de autentificare către o altă companie specializată în acest domeniu. Organizațiile decid să utilizeze o entitate externă datorită faptului că aceste platforme furnizează servicii avansate și complete de autentificare și includ metode simplificate de gestiune a utilizatorilor. Astfel, se evită implementarea tuturor acestor funcționalități în cadrul proiectelor desfășurate.<sup>[13]</sup>

Pentru a folosi această platformă, pentru început, am creat un cont de developer pe Okta-dev și am adăugat o nouă aplicație în secțiunea care se ocupă cu gestionarea aplicațiilor. Aici se regăsesc mai multe detalii, cum ar fi Client ID-ul( identificator unic asociat aplicației), URL-ul de redirectionare către pagina de Sign-in și de reîntoarcere către browser-ul web principal, dar și tipul fluxului de autentificare care în cazul aplicației create este authorization code, etc.

În momentul login-ului, utilizatorul este nevoit să-și introducă datele( Nume utilizator și Parolă ). După aceea, Okta le validează prin verificarea credențialelor în propria baza de date. Dacă sunt validate cu succes, se generează un token de acces pentru user-ul logat. Se folosește protocolul OpenID Connect pus la dispoziție de Okta care se ocupă cu furnizarea de token-uri pentru transmiterea de informații între client și server. Token-ul este criptat folosind algoritmul JWT (JSON Web Tokens), unic fiecărui utilizator și este preluat, după autentificare, în aplicație, pentru a permite accesarea datelor despre cont, numite resurse protejate. Aceste date includ informații despre id-ul unic al utilizatorului, email-ul acestuia, perioada de valabilitate a contului și permisiunile asociate.<sup>[13]</sup>

De asemenea, Okta gestionează și delogarea sau menținerea sesiunii de autentificare a unui utilizator pe toată durata folosirii aplicației. În cadrul aplicației, delogarea se face manual, la alegerea utilizatorului prin apăsarea unui buton denumit “Deconectare”.

În cazul unei posibile extinderi, am optat pentru simplitatea oferită de Okta prin protocolul OAuth 2.0 care permite autentificarea prin servicii terțe. La conectare, utilizatorul în lipsa unui cont creat, se poate loga printr-un cont deja existent la organizații predefinite, cum ar fi Google, Facebook. Aceasta este o metodă des întâlnită în aplicațiile web prezente în zilele noastre.

## Explicarea modelului arhitectural

Arhitectura unei aplicații web este una destul de standard în ceea ce privește elementele conținute și modul de relaționare dintre acestea. Înțelegerea structurii arhitecturale și cunoașterea tipurilor de arhitecturi existente, din care putem alege, sunt lucruri necesare în

vederea implementării unui model arhitectural cât mai avantajos pentru aplicația pe care o dezvoltăm.

Am ales tipul component-based architecture, deoarece serviciul client și cel de server vor fi două entități distincte ce vor rula independent, dar vor comunica la nevoie prin metode explicate în cadrul acestui subcapitol. Serverul va conține logica de afaceri, iar componenta client se va ocupa cu afișarea interfeței.

Arhitectura aplicației creată în această lucrare conține 5 elemente. Există serviciul Client care reprezintă interfața platformei. Această interfață este vizibilă utilizatorilor ce accesează aplicația. Un user poate interacționa cu diverse elemente din partea vizuală care, în funcție de scopul pentru care au fost adăugate, au implementate în cod o metodă predefinită ce conține un comportament specific.

Un exemplu ce va urma să fie inclus în dezvoltarea aplicației, reprezintă un buton sub formă de inimioară de culoare gri sau roșie cu următoarea funcționalitate. La apăsarea acestuia, se va adăuga rețeta curentă la favorite. În urma apăsării, imaginea inimii rămâne colorată roșu până când utilizatorul decide să o scoată de la favorite prin reapăsarea butonului. Acest model descrie un comportament ce implică interacțiunea client-server, însă există și elemente cu care utilizatorii pot interacționa care au implementate funcționalități strict vizuale.

În cazul aplicației web pentru gătit și evitarea risipei de mâncare, vom adăuga pentru vizualizarea ingredientelor o metodă care ne permite să bifăm elementele deținute deja, pentru a putea fi utilizată drept listă la cumpărături. Toate aceste acțiuni sunt metode ce trebuie definite în frontend. În plus, elementul client conține și browser-ul web la care este rulat aplicația.

Partea de server după cum îi spune și numele, conține serverul web care primește cereri de la browser și care furnizează răspunsul după accesarea datelor de la baza de date. Această componentă reprezintă serviciul de backend care a fost creat folosind framework-ul Spring Boot.

Request-urile sunt făcute în puncte specifice de acces numite endpoint-uri din cadrul unui API. Pentru o descriere mai generală, API-ul este ruta care conține mai multe endpoint-uri specifice. Transmiterea datelor de la browser-ul web către componenta server se realizează cu ajutorul protocolului HTTP. Metoda de comunicare constă în cereri HTTP trimise de browser către server și răspunsuri întoarse care conțin datele cerute, tot de tip HTTP.

Server-ul este conectat la baza de date dezvoltată și în momentul apariției unei cereri, accesează și interoghează doar tabelele necesare pentru formularea răspunsului dorit de client. Răspunsul care ajunge înapoi la client este prelucrat de frontend cu scopul de a îl afișa în interfața aplicației pentru a îl vizualiza utilizatorul. Spre exemplu, se apasă butonul de vizualizare a tuturor donațiilor, iar în momentul respectiv, clientul formulează o cerere și serverul furnizează răspunsul ce conține toate anunțurile de donații preluate din tabela donații din baza de date.

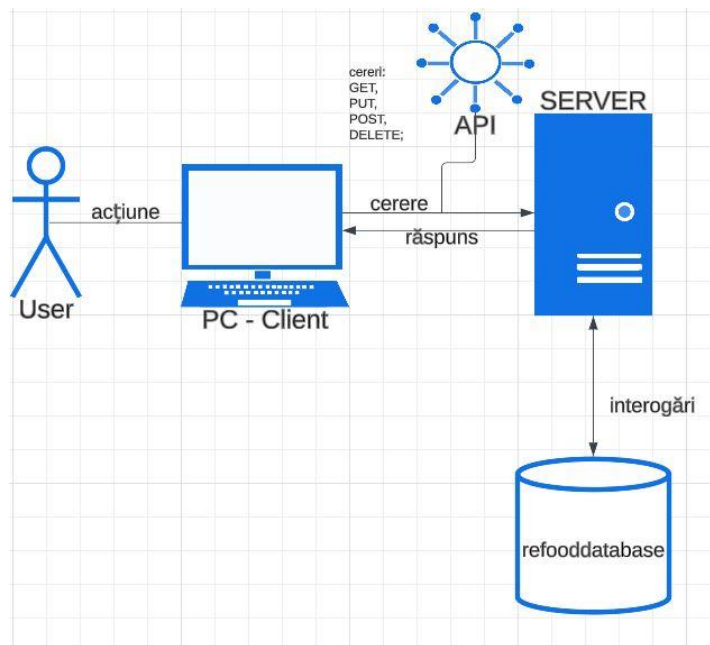


Figura 10: Funcționarea arhitecturii

Cererile făcute de browser pot fi de tip: GET, PUT, POST, DELETE. Pentru aplicația creată a fost implementată o singură metodă POST pentru actualizarea favoritelor unui utilizator. În continuare, voi detalia cererile și răspunsurile realizate între serviciul de frontend și backend. Pentru început a fost nevoie să implementez o cerere de tip GET pe tabela care conține toate înregistrările din tabela rețetă pentru a le putea afișa într-un meniu pe care utilizatorul să-l acceseze.

Pentru accesarea detaliilor unei rețete va fi necesar un răspuns care să includă toate datele despre fiecare rețetă în parte combinate cu toate înregistrările existente de ingrediente. În cadrul serviciului de frontend va trebui să se implementeze o metodă care să cupleze fiecare rețetă cu ingredientele sale corespondente.

Aceleași metode de GET vor trebui create pentru preluarea tuturor datelor ce populează tabela care stochează comentariile lăsate la fiecare rețetă (element ce va apărea tot în pagina despre detaliile rețetei) și tabela care cuprinde toate anunțurile postate în aplicație.

Metodele PUT și DELETE sunt implementate în cadrul gestionării favoritelor pentru fiecare user. Pe pagina care afișează toate rețetele, în dreptul fiecăreia apare un buton sub formă de inimă gri. Pentru a fi sugestiv când utilizatorul se află cu mouse-ul deasupra inimii aceasta se colorează roșu. Când butonul este apăsat, acesta se colorează roșu și rămâne colorat. În acest timp, se execută metoda de adăugare a unei noi înregistrări de tip favorite. Opus, când pentru o rețetă deja favorită se apasă butonul cu inimă roșie se accesează o metodă de ștergere a înregistrării corespondente.

## Implementarea aplicației

În cadrul acestui capitol, se vor descrie modul de implementare a unor elemente semnificative din cadrul aplicației.

### Crearea bazei de date

Pentru început, am creat o nouă bază de date în SGBD-ul prezentat anterior care este denumită „refooddatabase”. Apoi, a urmat identificarea tabelelor necesare aplicației. Funcționalitățile stabilite impun folosirea următoarelor tabele de sine stătătoare și anume: o tabelă pentru funcționalitatea de afișare a rețetelor, o tabelă separată de ingrediente pentru caracteristica de căutare după ingrediente deținute, o tabelă pentru stocarea comentariilor care oferă posibilitatea de a lăsa comentarii la rețete, o tabelă pentru rețetele favorite ce permite adăugarea și ștergerea favoritelor și nu în ultimul rând, o tabelă pentru anunțurile de donații.

Pentru implementarea acestei baze de date am ales adăugarea codului în trei scripturi diferite SQL pentru a facilita gestionarea eventualelor modificări necesare din motive de compatibilitate. La fiecare re-rulare a scripturilor, baza de date „refooddatabase” era automat ștearsă și recreată conform schimbărilor din cod.

Descrierea celor 5 tabele:

- Tabela numită „reteta” care conține toate rețetele folosite în aplicație identificate prin atributul id (tip `BIGINT(20) NOT NULL AUTO_INCREMENT`). Aceasta deține, de asemenea, și informații cu privire la detaliile rețetelor precum titlul acesteia, categoria(desert, prânz, cină, mic dejun), timpul de pregătire, gradul de dificultate(ușoară, medie, grea), modul de realizare adăugat sub formă de pași(tip `varchar(45) DEFAULT NULL`) separați prin expresia “Pas” din fața fiecărei etape noi care în interfață se va elimina înlocuindu-se cu o număratoare automată a pașilor de execuție a rețetei. Tabela mai include și două poze stocate sub formă de text care conțin adresele la care se găsesc imaginile online, una pentru vizualizarea rezultatului rețetei și cea de-a doua pentru observarea ingredientelor necesare. Această tabelă conține 40 de înregistrări inspirate din rețetele de pe site-ul *teo's kitchen*.<sup>[18]</sup>
- Tabela cu numele „ingrediente” este tabela care stochează ingredientele tuturor rețetelor, fiecare având un id cheie primară cu același tip menționat mai sus și un atribut `reteta_id` care este cheie străină cu ajutorul căreia se creează referiță către cheia primară id din tabela „reteta”. Această tabelă conține și denumirea și cantitatea ingredientului(tip `text DEFAULT NULL`).
- Tabela denumită „comentarii” definește comentariile lăsate pentru fiecare rețetă în parte, diferențiate prin atributul unic id și care include informații despre email-ul utilizatorului care și-a oferit feedback-ul, data în care a fost lansat comentariul( tip `datetime(6) DEFAULT NULL`), rating-ul ales( tip `decimal (3,2) DEFAULT NULL`)



care specifică numărul de stelute oferite și opțional un text pentru exprimarea opiniei cu privire la rețeta în cauză. După creare, am adăugat două înregistrări cu comentarii lăsate la prima rețetă de către doi utilizatori existenți prin care am verificat corectitudinea metodei implementate de GET.

- Tabela ce poartă numele „favorite” este o tabelă folosită pentru reținerea rețetelor favorite pentru fiecare user. Folosește atributul cheie primară id, email-ul utilizatorului care a adăugat rețeta la favorite și informații despre rețeta în cauză. Pentru această tabelă nu am adăugat nicio înregistrare, deoarece acestea trebuie să fie create doar la cererea user-ului în interfață.
- Tabela numită „anunturi” include toate anunțurile regăsite în cadrul forumului Relife, fiecare având un id not null de tip integer cu autoincrementare, un email al user-ului, un titlu al postării, o descriere de tip text default null și o dată de postare de tip datetime(6) care în caz că nu este specificată are o valoare default null. Anunțurile sunt adăugate în baza de date exclusiv prin interfață. Această tabelă este goală în momentul creării.

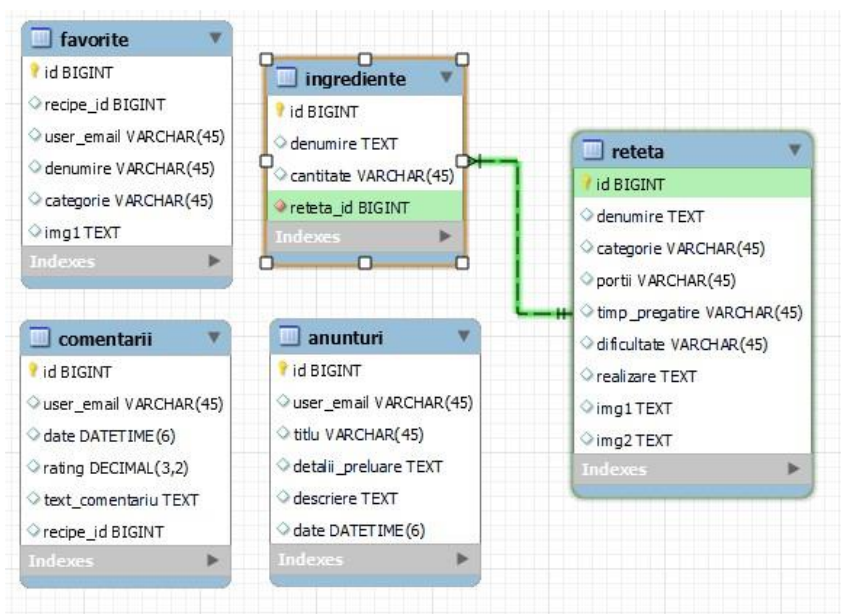


Figura 11: Tabelele create în baza de date folosind MySQL

Legătura între tabela „reteta” și tabela „ingrediente” este una de tipul One To Many ceea ce înseamnă că o rețetă poate avea unul sau mai multe ingrediente. Implicit, un ingredient va avea corespondență doar o rețetă (relație implementată în figura 13). Am ales această abordare pentru a putea integra și atributul cantitate în interiorul tablei „ingrediente”. Astfel putem găsi ingredientul „ardei” de n ori printre înregistrările existente, dar de fiecare dată cu o cantitate diferită. Deși implementarea în DBMS s-a complicat puțin, această rezolvare a simplificat implementarea în frontend și în backend.



## Descriere componentelor cheie backend

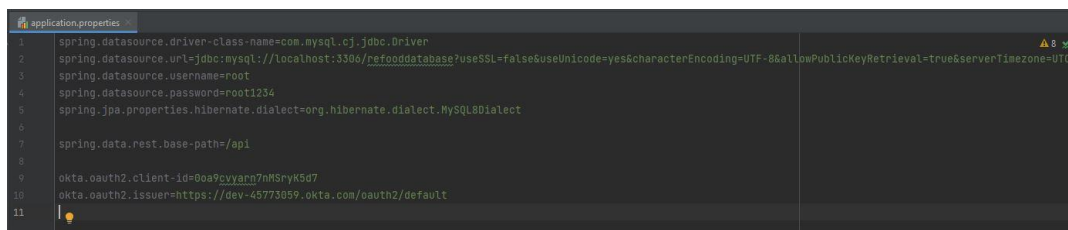
În cadrul acestei secțiuni, vor fi prezentate elementele cheie implementate în serviciul de backend.

### Conectarea la baza de date

Pentru integrarea bazei de date în serviciul de backend am editat conținutul fișierului de configurare `application.properties` și am specificat adresa url a sursei de date, username-ul și parola pentru accesarea și conectarea la baza de date creată și am specificat dialectul necesar pe care Spring Data JPA o să-l utilizeze pentru interogarea tabelor. Am adăugat și dependența cu `artifactId = mysql-connector-j` în fișierul `pom.xml`.

### Adăugarea serviciului Okta

În fișierul `pom.xml` am creat o nouă dependență cu `artifactId = okta-spring-boot-starter`, iar în cadrul `application.properties` am specificat clientul care este un identificator oferit la crearea unui cont de developer pe site și link-ul către issuer-ul pus la dispoziție Okta.



```
1 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
2 spring.datasource.url=jdbc:mysql://localhost:3306/recipe?useSSL=false&useUnicode=yes&characterEncoding=UTF-8&allowPublicKeyRetrieval=true&serverTimezone=UTC
3 spring.datasource.username=root
4 spring.datasource.password=root1234
5 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
6
7 spring.data.rest.base-path=/api
8
9 okta.oauth2.client-id=00a9c5vvarn7nM5ryK5d7
10 okta.oauth2.issuer=https://dev-45773059.okta.com/oauth2/default
11
```

Figura 12: Cod pentru conectarea backendului la baza de date și la serviciul Okta

### Crearea modelelor

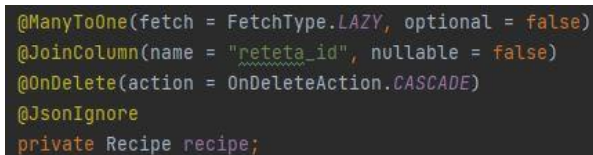
Am creat 5 fișiere de tip model pentru fiecare din cele 5 tabele detaliate în cadrul secțiunii anterioare. Aceste entități vor conține datele preluate de la baza de date.

Clasele sunt adnotate cu `@Data` pentru că stochează date, `@Table(name = „numele tabeli din baza de date”)` pentru că sunt modelate după tabele și `@Entity` pentru că vor fi folosite ca entități pentru gestionare și prelucrare.

În conținutul unei clase, vor fi definite toate câmpurile conținute de tabelă, adnotate cu `@Column(name = „numele câmpului din baza de date”)` și implementate ca atribute private ale clasei create.

Pentru atributele cheie privată „id” care sunt generate automat se vor folosi adnotările `@Id` și `@GeneratedValue(strategy = GenerationType.IDENTITY)`.

Pentru clasele care conțin atribute care stochează date s-a folosit tipul de date „Date” prin importarea `java.util.Date` și adnotarea `@CreationTimestamp`.



```
@ManyToOne(fetch = FetchType.LAZY, optional = false)
@JoinColumn(name = "recipe_id", nullable = false)
@OnDelete(action = OnDeleteAction.CASCADE)
@JsonIgnore
private Recipe recipe;
```

Figura 13: Cod pentru implementarea relației One To Many<sup>[2]</sup>

### Preluarea tuturor rețetelor

Pentru afișarea rețetelor preluate din tabela „reteta” din baza de date și afișarea lor în frontend în pagina de căutare a rețetelor, a fost necesară implementarea clasei publice „Recipe” cu atributele private setate în funcție de câmpurile tabelului din baza de date. Apoi, a fost creată interfața „RecipeRepo” care extinde JpaRepository <Recipe, Long> și care creează automat prin Spring Data JPA endpoint-ul cu adresa „/api/recipes” unde se vor găsi toate obiectele rețetă. În acest fișier, s-au creat două metode: căutare după denumire – „findByDenumireContaining” și sortare după categorie – „findByCategorie”, care folosesc convenții de numire și anume prefixul „findBy” și care specifică crearea automată a logicii metodei și a endpoint-urilor cu url-ul /search/ + denumirea\_metodei (această adresă permite specificarea denumirii sau categoriei, a numărului de pagini și a numărului total de obiecte pe pagină) cu ajutorul framework-ului JPA. Parametrii de căutare sunt obținuți prin cererea @RequestParam și mai apoi sunt folosiți pentru căutarea rețetelor care conțin denumirea căutată sau selectarea doar acelor ce corespund categoriei specificate. Ambele rezultate returnate sunt paginate prin folosirea „Page<Recipe>” și a obiectului „Pageable”.

### Preluarea tuturor ingredientelor

Pentru început, am creat clasa „Ingredient” și am setat atributele private pentru a corespunde cu cele din baza de date. În cazul câmpului „reteta\_id” care conține relația One To Many am folosit codul exemplificat în figura numărul 13. După aceea, am implementat interfața „IngredientRepo” ca extensie a JPA cu ajutorul căreia s-a creat automat logica de afaceri pentru metoda „findByRecipeId” care sortează ingredientele după atributul id al rețetei. Pentru preluarea tuturor ingredientelor în funcție de o rețetă, a fost necesară doar o cerere de tip @GetMapping("/recipes/{retetaId}/ingredients”) creată în fișierul „IngredientController”. Această metodă primește ca parametru doar id-ul rețetei după care se face căutarea, specificat prin @PathVariable(value = „retetaId”) pentru a prelua variabila din calea endpoint-ului. Acest id este verificat dacă există folosind instanțierea clasei „RecipeRepo”. Dacă rețeta nu există se va afișa un mesaj de eroare potrivit, dar dacă rețeta a fost găsită atunci se va crea o listă de elemente de tipul „Ingredient” care va conține ingredientele rețetei căutate, accesate prin apelarea metodei „findByRecipeId(retetaId)”. La final, se vor returna ingredientele găsite sub forma unei entități răspuns.

Următoarele trei servicii, față de manipulările de date descrise anterior care conțin doar metode de preluare, au în plus metode de postare și ștergere.

### Serviciul comentarii

După cum este menționat mai sus, primii pași pentru preluarea datelor sunt foarte similari. Am creat clasa „Comment” cu atributele necesare și metodele specificate pe parcursul acestui capitol. Apoi, în interfața „CommentRepo” am adăugat două metode: de căutare a tuturor comentariilor lăsate la o anumită rețetă – „findByRecipeId” și de căutare după email-ul unui utilizator în cadrul comentariilor sortate după id-ul rețetei – „findByUserEmailAndRecipeId”. Prima metodă oferă posibilitatea de paginare a rezultatelor. Comentariile au fost concepute să nu poată fi șterse, deci în fișierul „CommentController” există doar metodele de GET și POST ambele cu adresa securizată.

Pentru a determina în frontend dacă un user a lăsat comentariu deja pentru o anumită rețetă, printr-o mapare de tipul GET, se utilizează o metodă ce preia drept parametrii credențialele autentificării, token-ul, din header-ul "Authorization" și recipeId specificat în calea endpoint-ului. Email-ul este accesat folosind un fișier individual denumit JWT care include o metodă „Extraction”. Se apelează funcția de extragere cu parametrii token și "\sub\", elementul de unde poate fi regăsită informația despre email. Fișierul decodează partea de payload din interiorul token-ului, folosind algoritmul Base64, iar mai apoi se împarte stringul rămas după virgulă și se caută elementul corespunzător cu al doilea parametru de căutare al funcției. În cazul în care, la returnarea funcției email-ul este gol, atunci cererea HTTP nu este validată și se va afișa un mesaj de eroare. În continuare, pentru cazul în care există adresa de mail în cerere și aceasta a fost extrasă cu succes, se folosește obiectul „commentService” și se apelează metoda userCommentListed(userEmail, recipeId) din clasa „CommentService”. În această parte, se caută comentariul după email și id-ul rețetei folosind metoda descrisă la început din interfața „CommentRepo” și se returnează true, dacă a fost găsit review-ul sau false în caz contrar.

În schimb, pentru a posta un comentariu, avem nevoie de o metodă POST care a fost creată tot în cadrul clasei „CommentController” folosind endpoint-ul „/api/comments/secure”. Aceasta primește ca parametru token-ul de autentificare și un obiect de tipul „CommentRequest”. Fișierul de tipul request a fost folosit pentru stocarea datelor preluate din frontend. Prin urmare, va conține rating-ul lăsat de utilizator, id-ul rețetei la care se lasă review-ul și opțional un text drept comentariu. Se extrage email-ul user-ului ca în metoda descrisă anterior și după executare se face verificarea variabilei pentru a nu fi null. Dacă aceasta nu este null, atunci se va apela metoda postComment cu ajutorul instanței obiectului commentService. Se furnizează drept parametrii userEmail și obiectul commentRequest. În continuare, se verifică dacă respectivul comentariu nu a fost deja creat folosind metoda „findByUserEmailAndRecipeId” printr-un obiect commentRepo. Dacă acesta nu există deja, se continuă procesul de adăugare a comentariului. Mai întâi, se instatiază un obiect de tipul „Comment”, apoi se folosesc setters creați automat pentru fiecare atribut al obiectului. Atributele rating, recipeId și textComentariu sunt setate după preluarea lor folosind getters din clasa „CommentRequest”. În cazul textului, mai întâi este necesară verificarea existenței acestuia, deoarece este declarat ca opțional. Se folosește metoda isPresent() și după aceea este setat prin getter-ul specific, iar în caz contrar, acesta se setează null. Se continuă prin salvarea comentariului prin metoda pusă la dispoziție de JPA: commentRepo.save(comment).

### Serviciul favorite

Pentru început, am implementat clasa „Favorite” ce conține toate atributele descrise în cadrul capitolului „Crearea bazei de date” și folosind metodele expuse în subcapitolul „Crearea modelelor”. După aceea, în interfața creată cu numele „FavoriteRepo” care extinde repository-ul pus la dispoziție de JPA s-a implementat o metodă de afișare a tuturor rețetelor în funcție de utilizator, ce include posibilitatea de paginare a rezultatului și o altă metodă de căutare a favoritei după user și id-ul rețetei. Pentru acest serviciu, am creat atât metoda de adăugare a unei favorite, cât și de ștergere a acesteia. Endpoint-ul de actualizare a favoritei, de tipul PUT, se regăsește în fișierul „RecipeController” și este mapat la adresa „/secure/return”. Metoda extrage email-ul utilizatorului conform procesului ce implică folosirea JWT.Extraction(token, "\sub\""), unde token-ul este parametrul preluat din formularea cererii, în cadrul headers →>

Authorization. Al doilea parametru este id-ul rețetei care trebuie adăugată la favorite. Apoi este apelată metoda „returnRecipe” prin obiectul „recipeService”. În continuare, „recipeId” este preluat și folosit pentru căutarea rețetei folosind un obiect „recipeRepo” și metoda „findById”. Dacă rețeta nu există, procesul nu continuă și se afișează o eroare care specifică inexistența rețetei căutate. În mod contrar, se creează o nouă instanță a clasei „Favorite” și se setează attributele necesare. Email-ul și id-ul rețetei sunt cunoscute, iar denumirea, categoria și imaginea sunt obținute prin getters automați ai obiectului creat pe baza „recipeRepo”. În final se salvează favorita folosind: “favoriteRepo.save(favorite)”.

Metoda de ștergere a favoritei este implementată în fișierul „FavoriteController”. Aici este mapată metoda denumită „deleteRecipe” la adresa endpoint-ului „/secure/delete”. Parametrii acestei metode sunt aceiași ca în cazul adăugării unei noi favorite, și anume token-ul și id-ul rețetei. Se extrage din nou email-ul din token și se verifică utilizatorul. Dacă există, este apelată metoda „deleteRecipe”, prin obiectul „favoriteService”, cu aceiași doi parametri specificați anterior. Aici, se creează un obiect al clasei „Favorite” care este opțional și se stochează rezultatul metodei „findByUserEmailAndRecipeId” utilizată cu ajutorul unui obiect favoriteRepo. Dacă favorita este găsită, atunci se șterge folosind obiectul „favoriteRepo” și metoda .delete(favorite.get()), iar în caz contrar, se afișează mesajul: „Favorita căutată nu există”.

#### Serviciul donații

În cadrul acestui serviciu, s-a folosit clasa publică „Donation” care conține toate attributele specificate în tabela „anunturi” din baza de date. În interfața „DonationRepo”, creată în același mod ca restul interfețelor din backend, s-au implementat metodele „findByUserEmailAndId” ce returnează opțional un obiect de tip „Donation” care corespunde cu id-ul și email-ul după care se realizează căutarea și „findDonationsByUserEmail” ce accesează toate donațiile corespunzătoare unui anumit utilizator.

Pentru serviciul de donații sunt necesare următoarele acțiuni: de preluare a tuturor anunțurilor cu donații, de căutare a donațiilor realizate de utilizatorul curent, de postare(creare) a unui nou anunț și de ștergere a anunțurilor proprii. În continuare, se va detalia implementarea fiecărei metode. Pentru preluarea tuturor donațiilor s-a folosit endpoint-ul cu suport de paginare creat automat prin repository-ul JPA cu adresa „/api/donations”, unde se găsesc stocate toate înregistrările de tip donație preluate din baza de date. În interfață, utilizatorul curent va avea dreptul de ștergere doar asupra postărilor de donații create de el. Prin urmare, s-a folosit acțiunea de căutare a anunțurilor după email-ul user-ului, creată prin metoda menționată mai sus din interfața „DonationRepo”.

Acțiunea de adăugare a unei noi donații este o metodă mai complexă, ce a implicat crearea clasei „DonationRequest” unde au fost adăugate attributele ce urmează a fi setate de utilizator folosind aplicația și anume: titlu, detaliiPreluare și elementul opțional descrierea. În fișierul „DonationController”, am implementat o metodă de tipul POST mapată către calea endpoint-ului „/secure/create”. Aceasta primește doi parametri, primul este token-ul de autentificare preluat din header-ul cererii, iar al doilea este obiectul de tip „DonationRequest” specificat în cadrul body-ului. Metoda preia token-ul și extrage email-ul utilizatorului curent care dorește să

posteze anunțul utilizând fișierul JWT descris în serviciile anterioare. Se verifică dacă mail-ul este valid și se execută „postDonation” a obiectului „donationService”. În fișierul „DonationService” s-a implementat logica acțiunii de adăugare a unei noi donații prin implementarea metodei „postDonation” care primește drept parametrii: email-ul user-ului și un obiect de tip „DonationRequest”. Aici, se creează o nouă instanță a clasei „Donation” și se setează fiecare atribut în parte folosind setters. Spre exemplu, se setează email-ul obiectului folosind „donation.setUserEmail()” și se specifică drept parametru email-ul preluat din metoda „postDonation”. Pentru setarea titlului, detaliilor de preluare și a descrierii, se accesează attributele request-ului de donații prin utilizarea de getters: donationRequest.getTitlu(). În cazul descrierii, care este un atribut opțional, se verifică dacă acesta există în cerere folosind metoda .isPresent(). Data donației este creată folosind funcția „LocalDate.now()”. La final, se salvează obiectul donation folosind instanța „donationRepo” și metoda save().

Metoda ștergere a unei donații este creată tot în cadrul fișierului „DonationController” și poartă numele „deletePost”. Aceasta este de tipul DELETE, mapată la adresa „/secure/delete” și primește doi parametrii: token-ul utilizatorului care efectuează cererea și id-ul donației care trebuie ștearsă. Am specificat anterior faptul că s-a folosit metoda „findDonationsByEmail” pentru aflarea tuturor donațiilor realizate de un utilizator. Procesul de ștergere a unei postări poate fi realizat doar de user-ul care are email-ul său atașat la postare. În serviciul de backend, pentru ștergerea unei donații, nu se face această verificare. Cu toate acestea, în cadrul frontend-ului a fost implementată restricția specificată și va fi detaliată în capitolul denumit „Descriere componentelor cheie frontend”. În continuare, metoda extrage email-ul și verifică să nu fie null, pentru a îl trimite drept parametru alături de id-ul donației pentru metoda „deletePost” accesată folosind obiectul „donationService”. Aici, se creează un obiect opțional de tip „Donation” care va stoca donația găsită în urma rezultatului executării donationRepo.findByUserEmailAndId și, în cazul în care aceasta lipsește, se afișează un mesaj de eroare. În caz contrar, se șterge obiectul găsit, accesat prin donation.get(), folosind metoda delete a repository-ului de donații.

#### Fișierele de configurare

În cadrul serviciului de backend, am implementat două clase de configurare. Prima este denumită „DataConfig” și implementează RepositoryRestConfigurer, iar a doua se numește „SecurityConfig”. Cu ajutorul primei clase, se suprascrive, folosind adnotarea @Override, metoda „configureRepositoryRestConfiguration” și se specifică expunerea id-urilor pentru endpoint-urile entităților rețetă, ingredient, comentariu și donație. Tot aici, se apelează metoda „disableHttpAction” din cadrul aceluiși fișier care permite doar executarea cererilor GET pentru clasa „Recipe” și „Ingredient”, dezactivând opțiunile de PUT, POST, DELETE și PATCH. A doua clasă este folosită pentru configurarea securizării pentru endpoint-urile /comments, /favorites și /donations care, în urma metodei aplicate, pot fi accesate doar prin autentificare și stabilirea unei reguli de autentificare prin specificarea tipului de token și anume jwt. În cadrul ambelor clase, se realizează configurația CORS pentru a permite accesul la API doar pentru originea definită http://localhost:3000 unde se regăsește serviciul client și de unde se vor efectua toate cererile către server.



## Prezentarea interfeței

În cadrul acestei secțiuni, se va prezenta exclusiv interfața creată și se vor exemplifica diversele moduri de interacționare ale user-ului cu aplicația web. Aceste instrucțiuni vor servi drept ghid de utilizare pentru useri.

La accesarea browser-ului web unde se află aplicația, utilizatorul va fi întâmpinat de o pagină de Home. Această pagina conține un navbar, patru secțiuni menite să descrie aplicația și un footer.

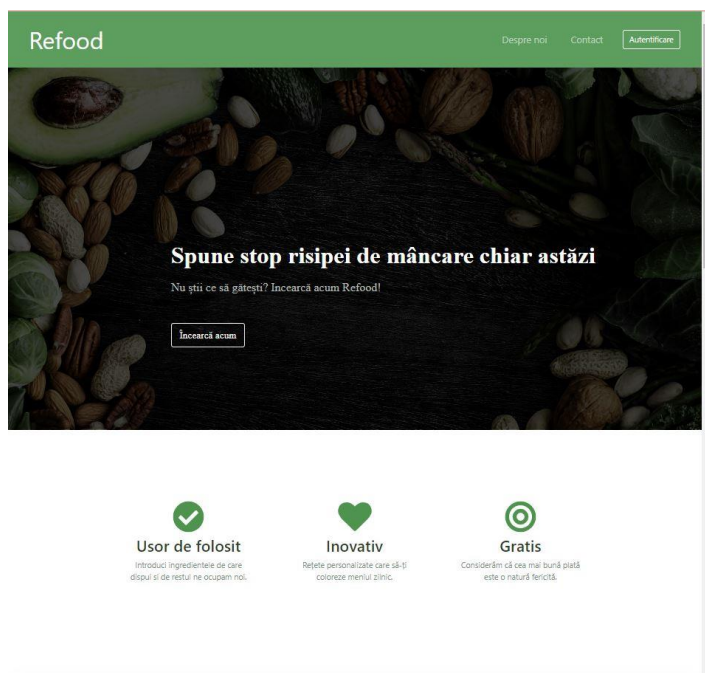


Figura 14: Pagina Home - Secțiunea numărul 1

Dacă utilizatorul apasă pe elementul „Despre noi” din partea dreaptă a navbar-ului este condus pe aceeași pagină, dar către secțiunea afișată în figura cu numărul 15. Această secțiune a fost adăugată cu scopul de a prezenta pe scurt aplicația pentru a atrage noi utilizatori.

Deoarece se cunoaște faptul că majoritatea oamenilor se decid greu în anumite zile cu privire la ce să gătească, această primă secțiune a pus accentul pe una dintre funcționalitățile principale ale aplicației, căutarea rețetelor după ingrediente.

În continuare, utilizatorul poate muta slide-ul în jos pentru a vizualiza a doua secțiune menită să descrie cea de-a doua funcționalitate specifică aplicației. Figura cu numărul 16 ilustrează această următoare secțiune.



Figura 15: Pagina Home - Secțiunea „Despre noi”

Butonul din stânga jos denumit „Explorează rețetele” trimite utilizatorul către meniul care prezintă toate rețetele. De aici, dacă utilizatorul nu are cont nu poate vizualiza detaliile de realizare a rețetelor și în cazul apăsării pe butonul „Citește mai mult” este redirecționat către pagina de autentificare care poate fi vizualizată în figura numărul 18.

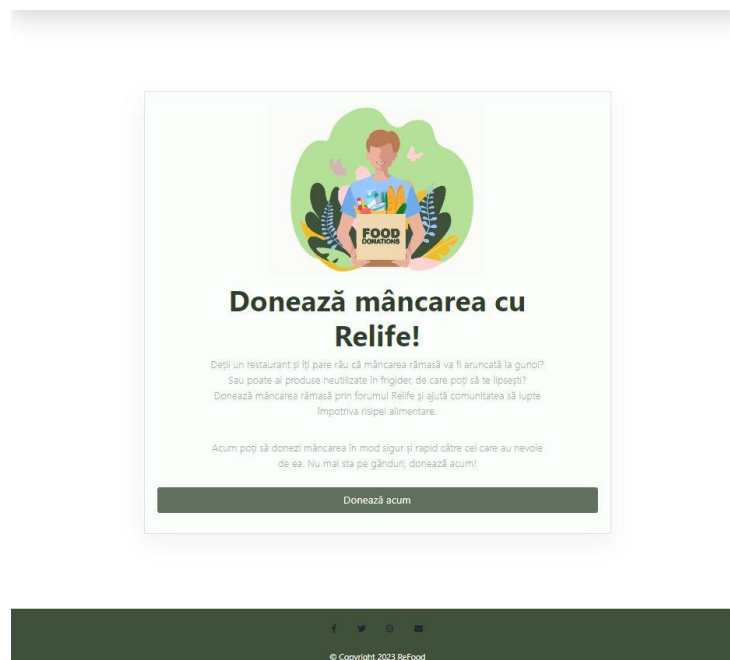


Figura 16: Pagina Home – A doua parte a secțiunii „Despre noi”

Această parte încurajează utilizatorii care dețin restaurante, producătorii de alimente sau persoanele fizice să doneze în funcție de posibilități folosind platforma Relife pusă la dispoziție de aplicația creată.

Al doilea element din cadrul navbar-ului este un buton de contact care schimbă pagina curentă și redirecționează user-ul către un formular pentru trimiterea unui mesaj.

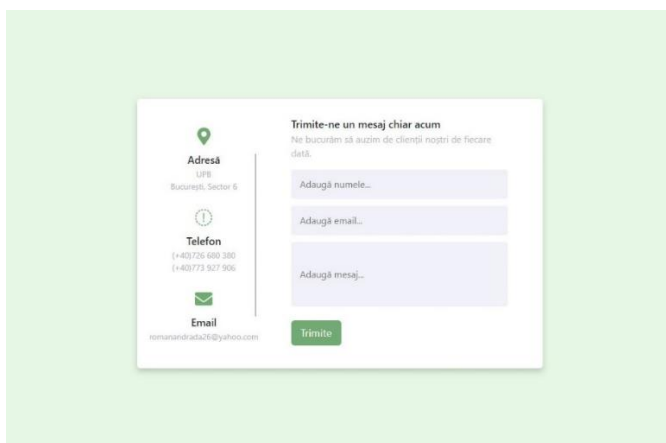
The image shows a contact page with a light green background. On the left, there is a contact information section with icons for location, phone, and email. The location is 'UPE Bucuresti, Sector 5'. The phone numbers are '+40726 699 380' and '+40773 507 906'. The email is 'romanandradu26@yahoo.com'. On the right, there is a section titled 'Trimite-ne un mesaj chiar acum' (Send us a message right now) with the text 'Ne bucurăm să auzim de clienții noștri de fiecare dată.' (We are happy to hear from our clients every day). Below this, there are three input fields: 'Adaugă numele...' (Add name...), 'Adaugă email...' (Add email...), and 'Adaugă mesaj...' (Add message...). At the bottom right of the form is a green button labeled 'Trimite' (Send).

Figura 17: Pagina de Contact

Dacă utilizatorul apasă butonul din dreapta navbar-ului denumit „Autentificare” sau pe butonul „Încearcă acum” din cadrul primei secțiuni din pagina Home, acesta este redirecționat către procesul de autentificare unde sunt solicitate email-ul sau numele de utilizator și parola pentru validarea accesului.

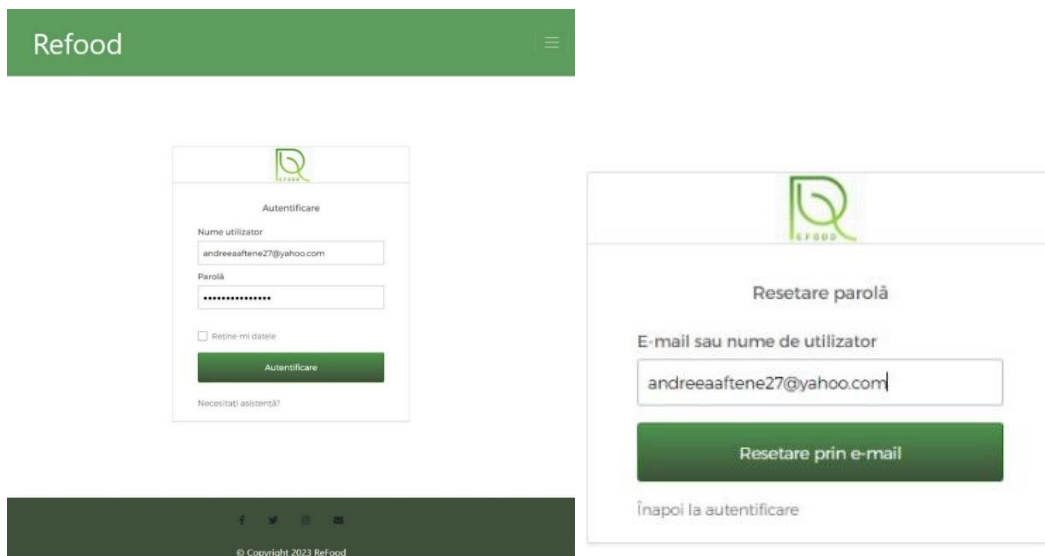
The image contains three screenshots. The top one is a green header bar with the 'Refood' logo on the left and a hamburger menu icon on the right. Below it are two screenshots of the authentication process. The left screenshot is the 'Autentificare' (Login) page, showing fields for 'Nume utilizator' (Username) with the value 'andreeaafte27@yahoo.com' and 'Parolă' (Password) with masked characters. There is a checkbox for 'Reține-mi datele' (Remember my data) and a green 'Autentificare' button. Below the button is a link 'Necesitați asistență?' (Need assistance?). The right screenshot is the 'Resetare parolă' (Reset password) page, showing a field for 'E-mail sau nume de utilizator' (Email or username) with the value 'andreeaafte27@yahoo.com' and a green 'Resetare prin e-mail' button. Below the button is a link 'Înapoi la autentificare' (Back to login). At the bottom of the entire image is a dark green footer bar with social media icons and the text '© Copyright 2023 Refood'.

Figura 18: Pagina de logare și opțiunea „Parolă uitată”

Datele adăugate pot fi reținute de browser pentru viitoare autentificări, metodă implementată prin Okta. La fel, în cazul butonului dropdown „Necesitați asistență?”, la apăsare apare un buton denumit „Parolă uitată” care trimite către adresa de mail specificată un mesaj de resetare a parolei.



Pagina de autentificare furnizată de Okta este predefinită din punct de vedere al design-ului, însă se poate modifica folosind cod CSS. De asemenea, oferă suport pentru diverse limbi. Aplicația creată folosește limba română datorită setărilor de preferințe din browser.

În caz că logarea se finalizează cu succes, utilizatorul este redirecționat către pagina care conține meniul lateral de unde pot fi accesate toate funcționalitățile aplicației. Fiecare element regăsit în partea stângă, din cadrul sided navbar-ului, modifică doar partea gri a conținutul acestei pagini.

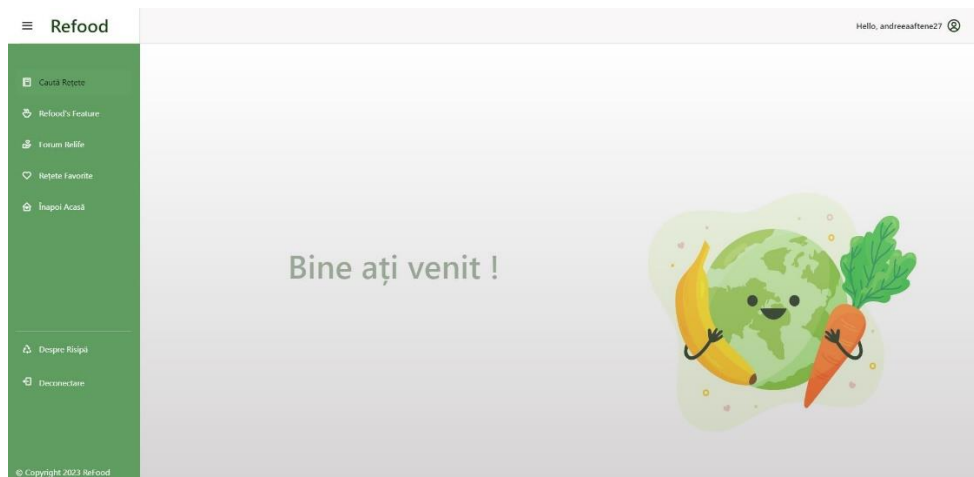


Figura 19: Pagina Menu de întâmpinare a utilizatorului logat

De aici, utilizatorul poate alege ce opțiuni din aplicație să încerce. În partea de sus a dashboard-ului, primele 4 butoane redirecționează utilizatorul către funcționalitățile cheie ale site-ului. În subcapitolul următor, aceste patru elemente semnificative din aplicație vor fi prezentate vizual prin poze și va fi descris în detaliu modul de implementare și funcționare. De asemenea, este implementat și un buton denumit „Înapoi acasă” cu ajutorul căruia se poate reajunge la pagina de Home prezentată anterior.

În partea de jos, avem opțiunea de vizualizare a unui conținut informativ cu privire la risipa alimentară ce se poate observa în figura 20, iar următorul buton se ocupă cu deconectarea utilizatorului din platformă care-l redirecționează automat către Home.

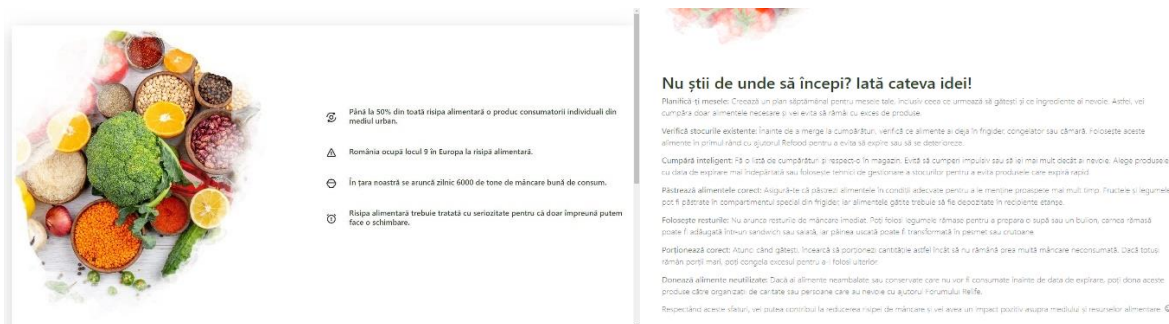


Figura 20: Opțiunea „Despre Risipă”

## Descriere componentelor cheie frontend

În acest subcapitol, se vor prezenta paginile ce includ funcționalitățile: de vizualizare a rețetelor și a detaliilor acestora, de căutare a rețetelor după ingrediente, de afișare a favoritelor și de observare și postare a unui anunț.

### Pagina de căutare a rețetelor

Pagina de afișare a tuturor rețetelor preluate din server, este o pagină de tip responsive care poate fi observată în figura cu numărul 21. Aceasta paginează rezultatele obținute și poate face sortarea rețetelor după categorie sau căutarea după titlu.

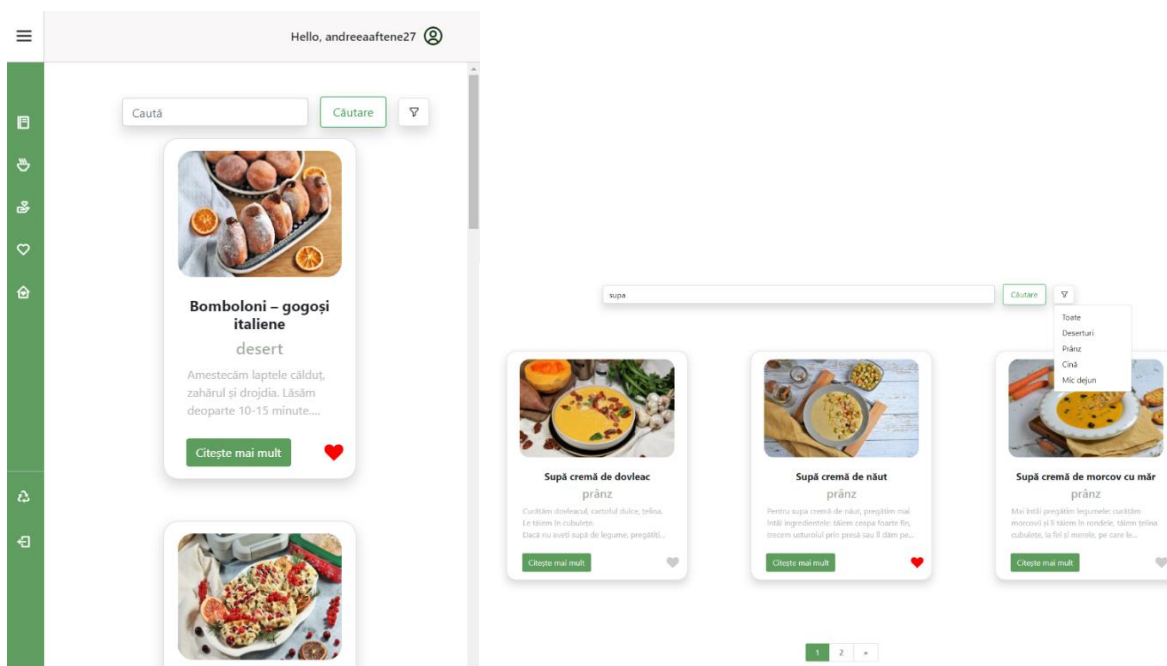


Figura 21: Opțiunea „Caută rețete”

Pentru stocarea datelor necesare acestei pagini în serviciul client am implementat clasa „Recipe” cu atributele obligatorii: id, denumire, categorie, realizare și atributele opționale: portii, timp pregătire, dificultate, img1 și img2. A fost adăugat un constructor pentru inițializarea tuturor proprietăților unui obiect nou creat.

Există două foldere în frontend numite „ShowRecipesAfterLogin” și „ShowRecipesPage”. Ambele au aceeași funcționalitate de afișare a rețetelor, însă prima le afișează cum este ilustrat în figura 21, iar cea de-a doua apare când un utilizator nelogat apasă butonul „Explorează rețetele” din prima parte a secțiunii „Despre noi”. A doua opțiune nu prezintă sidebar-ul și nici opțiunea de adăugare a rețetelor la favorite, din lipsa conectării user-ului la cont.

Ambele directoare conțin pagina principală „ShowRecipes.tsx” care se ocupă cu afișarea tuturor rețetelor și un alt fișier creat în subdirectorul „elements” denumit „FindRecipes.tsx” care preia datele câte unei rețete pe rând pentru crearea fiecărui card în parte.

În fișierele principale care fac posibilă afișarea, există un `useEffect` care conține funcția „`fetchRecipes`”. Aceasta include o cerere HTTP care așteaptă răspunsul de la url-ul specificat „`/api/recipes`”, îl stochează într-o variabilă și folosind metoda `.json()` parsează datele într-un obiect JSON care are proprietățile următoare: `_embedded.recipes` (include toate elementele rețetă), `page.totalElements` (conține numărul total de elemente) și `page.totalPages` (stochează numărul paginilor). Se va inițializa un tablou gol denumit „`Recipesload`” de tip „`Recipe`” unde se vor adăuga obiectele preluate prin proprietățile răspunsului JSON. În continuare, se va actualiza variabila de stare „`recipes`” cu conținutul array-ului „`Recipesload`” folosind funcția declarată „`setRecipe`”. ( `const [recipes, setRecipes] = useState<Recipe[]>([]);` ) Acest `useEffect` se va executa la fiecare modificare a url-ului sau a paginii curente. În mod normal, `recipes` va conține doar 6 elemente de tip rețetă, deoarece atâtea elemente au fost setate să fie conținute pe câte o pagină folosind variabila de tipul `useState` denumită „`nrRecipesPg`”. Aceasta specifică proprietatea `size` din cadrul url-ului și, pentru figura 21, a fost modificată la 3 elemente pe pagină cu scopul surprinderii esențialului în aceeași imagine. Tot în acest fișier, independent de `useEffect`, există alte două funcții apelate la căutarea după titlu sau după categorie.

Utilizatorul introduce o denumire de rețetă, pe baza căreia să se execute căutarea, în interiorul unui element de tip `input` care conține atributul `onChange` ce apelează funcția de modificare a stării variabilei `find` pentru a corespunde cu `input`-ul utilizatorului. Dacă se apasă, în continuare, pe butonul denumit „Căutare” se va declanșa, folosind `onClick`, funcția `findDenumireChange()`. Această funcție setează pagina curentă la 1 și verifică dacă elementul `find` nu este gol pentru a modifica variabila url adăugându-i calea „`/search/findByDenumireContaining`” și specificând denumirea conținută în variabila `find`, numărul de pagini și numărul de rețete afișate într-o pagină. Dacă nu există nicio rețetă care să conțină textul căutat de utilizator, se va afișa un mesaj de eroare.

Pentru sortarea în funcție de categorie, a fost implementat un buton de tipul `dropdown` care să conțină lista cu toate categoriile existente. Când un element din listă este apăsat, prin proprietatea `onClick`, se va apela funcția `findCategorieChange('categorie')`. Cuvântul „`categorie`” va fi înlocuit cu numele elementului apăsat. Astfel, funcția implementată primește deja categoria după care se face sortarea drept parametru și se va seta pagina curentă la 1, apoi se va schimba calea url-ului cu „`/search/findByCategorie`”. Se vor specifica, de asemenea, categoria conținută în parametrul funcției, numărul de pagini și numărul de rețete afișate într-o pagină. În cazul în care se alege opțiunea toate, nu se va schimba adresa url-ului.

Paginarea, în cadrul acestei funcționalități, este implementată în frontend folosind o componentă separată ce primește drept parametrii: `currentPg`, `totalPg` și `pagPg`. În acest fișier, se creează un array `pageNr` care va conține toate numerele de pagină care pot fi afișate în funcție de `currentPg` și `totalPg`. Se vor afișa cele două butoane de navigare, către pagina din stânga și pagina din dreapta, doar dacă pagina curentă nu este 1 sau ultima pagină. Afișarea noilor pagini și a paginii active curente se face prin actualizarea variabilei în funcție de apăsarea controalelor de navigare.

Fișierele „`FindRecipes.tsx`” primesc ca parametru un obiect de tip rețetă și îl utilizează sub forma unui `props` care va conține rețeta curentă de afișat. Deci acestea sunt responsabile pentru

afișarea unui sigur card care conține doar o rețetă. Această componentă se randează pe fiecare element mapat din cadrul tabloului „recipes” ce conține toate rețetele de afișat. În interiorul card-urilor, se poate observa butonul în formă de inimă ce specifică dacă rețeta este sau nu printre favoritele utilizatorului. Prin urmare, în fișierele „FindRecipes.tsx” se regăsește un `useEffect`. În funcția de efect asincronă se verifică, pentru început, dacă există elementul `authState` și dacă proprietatea `.isAuthenticated` este `true`. Doar în acest caz, se creează o metodă de tip GET către adresa „api/favorites/search/findRecipesByUserEmail” prin care se specifică email-ul utilizatorului după care se face căutarea. Răspunsul acestei cereri este un tablou de elemente de tip „Favorite” stocate în variabila de stare „favorites”, folosind funcția „setFavorites”. Această variabilă va conține toate rețetele favorite ale utilizatorului curent. În cadrul funcției „isRecipeInFavorite”, pentru fiecare element din array-ul de favorite, se verifică dacă denumirea favoritei este aceeași cu denumirea rețetei curente trimise prin props. Funcția returnează elementul „isMatch”, inițializat cu `false`, care devine `true` în momentul găsirii rețetei curente în totalul de rețete favorite ale user-ului. După finalizare, valoarea returnată este stocată într-o variabilă de stare „isFavorite”, prin funcția de actualizare a stării „setIsFavorite”. Acest `useEffect` este recalculat la fiecare modificare a tabloului „favorites” și a variabilei „authState”. Procesul descris anterior este folosit doar pentru determinarea favoritelor unui utilizator, însă sunt implementate și două funcții de adăugare și ștergere a favoritelor care vor fi descrise în continuare.

Funcția asincronă de adăugare a unei favorite se numește „returnRecipe” și are drept parametru id-ul rețetei de adăugat. În cadrul acesteia, se realizează o cerere PUT cu opțiunea headers `—> Authorization: `Bearer accessToken( accesat prin variabila authState )`` la adresa „api/recipes/secure/return”. Se așteaptă răspunsul folosind `await` și la finalizare se stochează rezultatul care determină dacă adăugarea a avut succes sau nu.


Pentru ștergerea unei favorite, am folosit o metodă similară care presupune implementarea unei funcții asincrone denumite „deleteRecipe” ce primește drept parametru „recipeId” ( id-ul rețetei curente la care s-a executat apăsarea butonului inimă). Funcția creează o cerere de tip DELETE autorizată prin opțiunea headers care transmite token-ul autentificării curente. Se așteaptă rezultatul `await fetch( url-ul endpoint-ului + opțiunile cererii)`. Adresa unde se trimite cererea este stocată într-o variabilă denumită `url` cu valoarea „api/favorites/secure/delete”. În cazul ambelor metode este necesar să existe specificată în calea cererii id-ul rețetei curente de adăugat sau șters. În final, se reține răspunsul în variabila „returnResponse”. Dacă aceasta este nulă, înseamnă că ștergerea a eșuat și o componentă de încărcare a paginii apare în interfața cu utilizatorul pentru a îl avertiza că pentru un moment serviciul nu este disponibil.

De asemenea, tot în acest fișier, există implementată și funcția denumită „handleClickFavorite” care este apelată la apăsarea butonului inimă, folosind proprietatea `onClick`. Aceasta verifică dacă variabila „isFavorite” este `true` ( ceea ce înseamnă că rețeta este deja inclusă în favoritele utilizatorului, rezultând că se dorește ștergerea acesteia din cadrul favoritelor) și în acest caz, apelează `await „deleteRecipe(props.recipe.id)”` și execută „setIsFavorite” cu valoarea opusă a variabilei „isFavorite”. În caz contrar, rețeta nu se află în favoritele user-ului și se dorește adăugarea. Aceasta se realizează prin apelul funcției „returnRecipe” și setarea variabilei „isFavorite” la opusul valorii curente.

## Pagina de afișare a detaliilor rețetelor

Pornind de la pagina ilustrată în figura 21, dacă utilizatorul apasă pe butonul „Citește mai mult” este redirecționat către pagina observată în imaginea cu numărul 22. Această pagină se ocupă cu prezentarea detaliilor rețetei (numărul de porții raportat la cantitățile afișate, gradul de dificultate și timpul de gătire), a modului său de realizare pe pași și a ingredientelor folosite, care apar scrise și ilustrate printr-o imagine alăturată. Aici, utilizatorului poate lăsa un review dacă nu a lăsat deja pentru această rețetă, care constă într-un rating și un text opțional sau poate observa în partea de jos a paginii toate comentariile postate de către alți utilizatori care au încercat rețeta.

Bomboloni – gogoși italiene



Tip rețetă: desert  
Ingrediente pentru 15 bucati  
Grad de dificultate: mediu  
Timp de realizare: 1h


Ne dorim să aflăm parerea ta!  
Dorți să lăsați un comentariu? \*

5 ★★★★★

Discuție

Opțional

Postează



Ingrediente:

- 500g făină
- 60g zahăr
- 100ml lapte
- 45g drojdie uscată
- 500g ulei
- 3 ouă
- 1/2 linguriță extract de vanilie
- cremă de ciocolată
- 1plic zahăr pudră vanilinat

MOD DE PREPARARE

- Amestecăm laptele cald, zahărul și drojdia. Lăsam dospirea 10-15 minute. Intr-un bol încălzitor punem făina, facem o gropiță în mijloc, turnăm amestecul de lapte.
- Putăm mișcăm cu acelorul de frământare sau frământăm cu mâna. Adăugăm treptat ouale bătute, vanilia, coaja de portocală.
- Frământăm până 10-20 min sau până avem un aluat elastic, care se desprinde de pereții vasului și de mâini.
- Mulțum aluatul într-un bol curat, uns cu puțin ulei. Acoperim cu folie alimentară și lăsam la dospit 2 ore sau până își dublează volumul.
- Lucrăm pe masa unsă cu puțin ulei. Întindem aluatul într-o foaie mai groasă, de circa 2 cm. Deslușăm ferma rețutată. Frământăm aluatul rămas și îl întindem din nou, deslușăm ferma până la epuizarea aluatului.
- Măsim ferma gogoșii pe hârtie de copt (hârtie de hârtie de copt în gălărie) și lăsam să dospescă alte 30 min. De ce folosim hârtie de copt? Aluatul este mai ușor de manipulat fără a cădea forma gogoșilor și ne ajută să le ținem în baie de ulei, fără pericol de a ne arde.
- Între timp încălzim uleiul. Cum verificăm că e gata pentru prăjit? Introducem o scobitoare/figușcă de lemn în ulei și, dacă se fac mișcări de ulei în jurul ei, baia de ulei este pregătită pentru gogoși.
- Prăjim gogoșile la foc mediu, întorcându-le des, pentru a se rumeni uniform pe ambele părți. Ținem că sunt gata când sunt frumose rumenite și rămân doar o mică dungă de aluat mai deschis la culoare pe mijlocul lor.
- Le scoatem pe hârtie absorbantă, cu ajutorul unei paletă. Le lăsam să se scurgă bine de uleiul în exces, apoi le trecem prin zahăr pudră/făină și, opțional, le scurgem cu ciocolată cu ajutorul unei seringi de pastă.

Secțiune de comentarii:

alexbarbulescu19@yahoo.com  
June 10, 2023  
Am încercat această rețetă și este delicioasă. Căi una din fazele mele!

andrardulescu3@yahoo.com  
June 10, 2023  
O rețetă puțin dificilă, dar interesantă. Mă rugăm să fi încercat când aveți suficient timp la dispoziție :)

Figura 22: Pagina ce conține detaliile unei rețete

Pentru această componentă, a fost creat un director separat care conține fișierul principal „DetailsRecipe.tsx” și folderul „elements” unde se află patru fișiere necesare manipulării comentariilor.

La apăsarea butonului denumit „Citește mai mult” este utilizat un element de tip Link ce redirecționează aplicația folosind proprietatea „to” către adresa componentei de afișare a detaliilor menționându-se id-ul rețetei pe care s-a executat click-ul. Acest id este preluat în fișierul principal din pathname-ul ferestrei și este creată o variabilă „idRecipe” care stochează rețeta curentă. În cadrul fișierului principal, există 3 useEffect-uri. Un useEffect este folosit pentru crearea a două cereri. Prima funcție de efect așteaptă un răspuns de la endpoint-ul „api/recipes/\${idRecipe}”, apoi preia răspunsul JSON primit care conține detaliile despre rețeta curentă și îl stochează într-o variabilă de stare de tip „Recipe”. Acest obiect este folosit pentru afișarea titlului, pozelor, informațiilor de pregătire (figura 22 colțul din dreapta sus) și modului de preparare. Același useEffect se utilizează pentru preluarea ingredientelor rețetei curente. Astfel, cea de-a doua funcție „findAllIngredients” accesează adresa endpoint-ului creat în

backend „/api/recipes/\${idRecipe}/ingredients”. Răspunsul este stocat prin aceeași metodă descrisă anterior, în variabila „ingredients”, un tablou de element de tip „Ingredients”.

Vizualizarea ingredientelor drept listă cu care se poate interacționa, prin tăierea acelor deținute, este o funcție implementată în cadrul fișierului „DetailsRecipe.tsx”. Pentru început a fost creată o variabilă de stare de tip tablou de date number denumită „selectedIngredients”. Aceasta va fi goală la inițializare și pe parcurs va stoca id-urile ingredientelor ce sunt selectate de utilizator. Am implementat în fișierul „App.css” un stil ce colorează fundalul cercului din dreptul ingredientului în culoarea verde și adaugă un text-decoration de tipul line-through. Acest stil a fost atașat className-ului „ticked”. La afișarea ingredientelor se folosește variabila creată anterior care conține ingredientele rețetei curente. Array-ul se mapează pentru iterarea prin fiecare ingredient, iar pentru fiecare iterație se creează un element nou în listă și se verifică dacă id-ul ingredientului curent se regăsește în tabloul „selectedIngredient”. Dacă acesta există, atunci se adaugă la className-ul deja atașat și „ticked”. Elementul „li” apelează la apăsare, prin onClick, funcția „Tick(ingredient.id)”. Această funcție verifică dacă id-ul ingredientului este inclus în array-ul „selectedIngredient”. Dacă acesta este deja inclus, atunci se elimină apariția lui în tablou, iar în caz contrar se adaugă id-ul în „selectedIngredient”. Ceea ce înseamnă că un element este tăiat la prima apăsare și revine la normal la următoarea.

Al doilea useEffect, conține o funcție ce creează o cerere către adresa „/api/comments/search/findByRecipeId?recipeId=\${idRecipe}” și apoi stochează răspunsul într-o variabilă de stare de tip tablou de elemente „Comment”. Procesul este același în cazul tuturor manipularilor de răspunsuri primite de la backend. Se așteaptă răspunsul și se stochează într-o variabilă. Dacă răspunsul nu are proprietatea ok atunci se aruncă o eroare, în caz contrar, se continuă execuția. Se transformă răspunsul într-un răspuns de tip JSON și se accesează și salvează proprietatea „\_embedded.comments”. Se parcurge răspunsul și pentru fiecare element din răspuns sunt preluate atributele și adăugate( prin metoda push) într-un array de obiecte „Comment” care va fi folosit pentru setarea variabilei de stare finale ce va conține toate comentariile, cu ajutorul funcției „setComments”. Pentru fiecare funcție de efect, dacă apare o eroare, se setează variabila de încărcare a datelor la false și se afișează mesajul de eroare în browser, specificând funcția în cauză.

Ultima funcție de efect este folosită pentru a afla dacă utilizatorul curent a lăsat deja comentariu la rețeta pentru care sunt afișate detaliile. În interiorul funcției, se verifică dacă utilizatorul este deja logat folosind o variabilă „authState” inițializată prin hook-ul useOktaAuth(), furnizat de serviciul de autentificare integrat în aplicație, Okta. Se verifică dacă elementul nu este null și dacă proprietatea isAuthenticated există, iar în acest caz, se execută o cerere GET care transmite în headers —> Authorization, token-ul de autentificare al utilizatorului curent. Se așteaptă primirea înapoi a răspunsului și apoi, se setează variabila de stare inițializată false, folosind funcția „setIsCommLeft(răspuns)”. După cum știm din cadrul subcapitolului „Serviciul Comentarii”, răspunsul provenit de la server returnează true dacă a găsit utilizatorul prin comentariile lăuate la rețeta curentă și false în caz contrar. Acest useEffect este recalculat la fiecare modificare a variabilei authState. De fiecare dată când variabila „isCommLeft” se va modifica, va determina recalcularea tuturor comentariilor preluate de la server prin re-executarea useEffect-ului cu numărul doi. În card-ul din pagina curentă, colțul din dreapta sus,

este afișată sub informațiile despre rețetă, o funcție numită „canLeaveComm” care verifică variabila „authState” pentru a determina dacă user-ul este deja logat și variabila „isCommLeft” setată în useEffect. Dacă ambele sunt adevărate, înseamnă ca utilizatorul a lăsat deja un review și un mesaj corespunzător este afișat, iar dacă user-ul este autentificat, dar variabila care determină dacă a fost lăsat comentariu este falsă, atunci se randează componenta „LeaveComment”.

Această componentă apelează la rândul ei, funcția cu numele „submitComment” ce conține parametrii „rating” și „textComentariu” setate folosind form-uri. Orice metodă de crearea a unei noi înregistrări, implică crearea unui nou model de request. Am implementat clasa „CommentRequest” cu atributele rating, textComentariu și recipeId. Se va instanția un obiect al acestei clase și se vor adăuga valorile atributelor primite ca parametrii și variabila „recipeId” din fișierul principal. Metoda de POST este astfel realizată printr-o funcție asincronă ce realizează o cerere către adresa „api/comments/secure”, specificând proprietățile headers și body. În body, este trimis obiectul creat „commentRequest”. Se așteaptă și verifică răspunsul și dacă aceste etape sunt validate se setează variabila de stare „isCommLeft” la true folosind funcția specifică de actualizare a stării.

#### Pagina de căutare după ingrediente

Utilizatorul poate accesa conținutul acestei pagini dacă apasă pe opțiunea „Refood’s Feature” din cadrul sided navbar-ului. Aici, utilizatorul poate introduce toate ingredientele, care pot avea un termen de valabilitate apropiat, și pe care dorește să le folosească în gătirea unei rețete. Algoritmul de căutare, afișează rețetele în ordinea numărului de apariții a ingredientelor menționate. Se vor afișa primele, rețetele care conțin toate ingredientele, urmând apoi rețetele ce conțin numărul total de ingrediente -1 ș.a.m.d, până la rețetele ce conțin un singur ingredient căutat. Și folosind această pagină se pot accesa detaliile unei rețete și se pot observa favoritele setate de utilizator, modul de interacționare fiind același descris în subcapitolele anterioare.

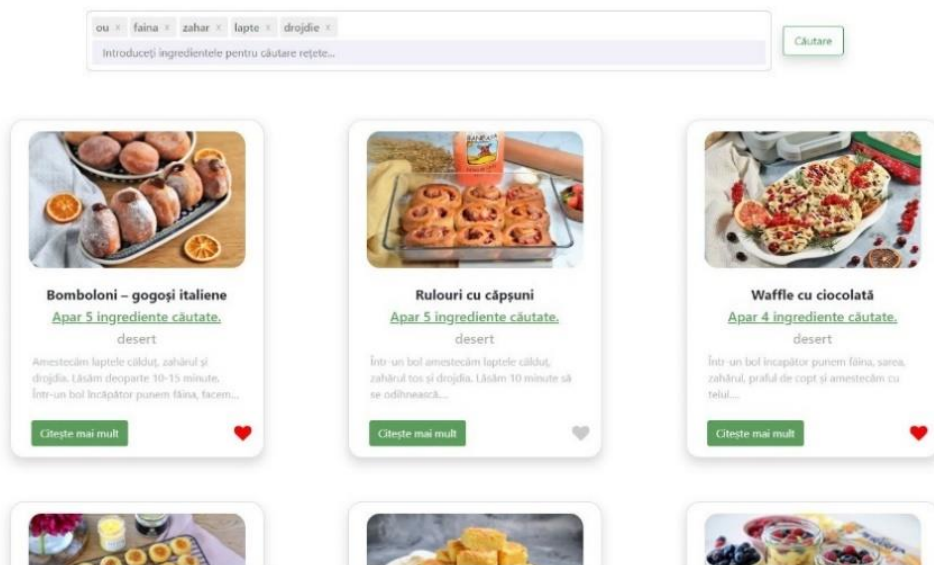


Figura 23: Opțiunea „Refood’s Feature”



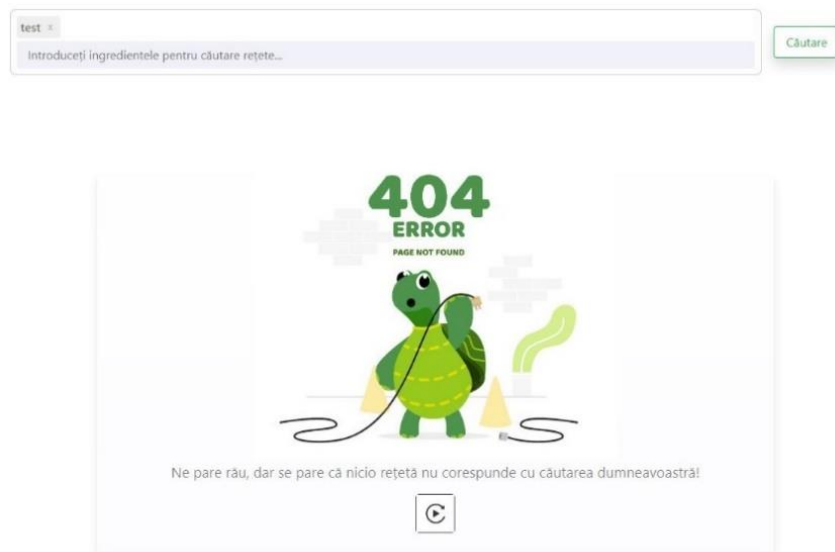


Figura 24: Rezultatul căutării după ingrediente inexistente

Pentru a începe procesul, un user trebuie să introducă un ingredient dorit ( se pot introduce texte și cu, și fără diacritice, deoarece nu vor impacta rezultatul algoritmului ) și după finalizarea fiecărui cuvânt se va da enter pentru a stoca ingredientul. De asemenea, după stocare, acesta apare în partea de sus a text field-ului pentru a demonstra validarea procesului de adăugare ingredient. Ingredientele adăugate pot fi șterse individual oricând până la apăsarea butonului de căutare care pornește algoritmul. Ștergerea acestora este posibilă și după afișarea rezultatului, însă nu va modifica rețetele afișate în urma noilor ingrediente, decât după o nouă apăsare a butonului denumit „Căutare”. Dacă ingredientele căutate au fost scrise greșit sau nu există în baza de date, atunci se va afișa un mesaj de eroare ce poate fi vizualizat în figura cu numărul 24. Există un buton de replay care permite utilizatorului reîncărcarea paginii pentru restartarea funcționalității de căutare după ingrediente.

Pentru această componentă a fost creat folderul „RefoodFeaturePage” ce conține fișierul principal și directorul „elements” care, la rândul lui, include fișierul „FindRecipeHashMap.tsx”. În cadrul fișierului principal, sunt implementate trei useEffect-uri care formulează următoarele cereri. Folosind funcția „fetchRecipes”, s-a accesat endpoint-ul cu adresa „api/recipes” pentru preluarea și afișarea tuturor rețetelor paginate. Această metodă este identică cu cea prezentată în cadrul secțiunii „Pagina de căutare a rețetelor”. De asemenea, răspunsul este folosit în același mod, în interiorul return-ului ce conține HTML-ul componentei, se mapează tabloul rezultat „recipes” și pentru fiecare element se accesează fișierul „FindRecipe” cu parametru setat la rețeta curentă. Pentru căutarea ingredientelor introduse de utilizator, este necesară o variabilă care să includă toate elementele de tip rețetă. Deoarece prima metodă descrisă oferă suport pentru paginarea datelor, variabila „recipes” va conține, în orice moment al rulării, doar rețetele prezente în pagina curentă. În schimb, pentru efectuarea algoritmului de căutare a ingredientelor, trebuie verificat fiecare ingredient al fiecărei rețete cu fiecare ingredient introdus



de user. Prin urmare, a fost eliminată paginarea în a doua cerere și astfel am stocat în variabila de stare de tip tablou de elemente „Recipe”, numită „totalRecipes”, toate rețetele după care se va realiza căutarea. Acest `useEffect` se recalculează la fiecare modificare a variabilei de stare „`isSearchFinished`”.

În continuare, voi prezenta implementarea elementului de tipul input. Acesta conține următoarele proprietăți: `onChange` care execută funcția „`handleInputChange`” și `onKeyPress` care apelează „`handleKeyPress`”. Prima stochează valoarea introdusă de la tastatură în variabila de stare „`inputValue`”, folosind funcția de actualizare a stării, iar cea de-a doua detectează când user-ul a apăsăst tasta `enter` și, în acel moment, salvează cuvântul format în `inputValue` (iar la final îl resetează la un string gol) în variabila de stare, de tip tablou de string-uri, numită „`selectedWords`”. În cadrul html-ului, este mapat fiecare element al array-ului „`selectedWords`” și este creată caseta în care este introdus cuvântul după `enter`. În interiorul casetei ce conține cuvântul, există un buton `X`, ce folosește proprietatea `onClick` prin apelarea funcției „`handleRemoveWord`” cu parametrul setat prin index-ul cuvântului. Se execută funcția de căutare după index a cuvântului în tabloul „`selectedWords`” și se execută ștergerea acestuia.

La apăsarea butonului de căutare se începe procesul, prin apelul funcției „`handleSearch`”, accesată folosind proprietatea `onClick`. În interiorul „`handleSearch`”, funcție asincronă, se setează variabila „`isSearchClicked`” la `true` și folosind un `for` cu `k` de la 1 la numărul total de rețete (variabila „`totalRecipes`”) se apelează funcția „`handleCompareRecipeSelectedWords(k)`” și se așteaptă finalizarea acesteia prin utilizarea `await`. Funcția respectivă creează o cerere către endpoint-ul de preluare a tuturor ingredientelor rețetei curente „`recipeIndex`”, descrisă în secțiunea „Pagina de afișare a detaliilor rețetelor”. După preluarea răspunsului, rezultatul se stochează în tabloul „`ingredients`”. Se setează variabila apariții la 0 și se execută un `for` ce iterează prin fiecare cuvânt din „`selectedWords`”, în care există un alt `for` care parcurge fiecare element din tabloul „`ingredients`”. Pentru fiecare combinație cuvânt-ingredient se verifică dacă cele două string-uri sunt egale folosind funcția „`areEqualStrings`”. În cadrul acesteia, se preiau cele două variabile ca parametri și sunt transformate prin eliminarea literelor mari și a diacriticilor. La final, se verifică dacă string-ul modificat al cuvântului introdus este inclus în string-ul obținut al ingredientului rețetei și se returnează `true` pentru o condiție îndeplinită. În cele două `for-uri`, în momentul când `if-ul` este `true`, se crește numărul aparițiilor cu 1, se setează, în variabila „`hashMap`”, o intrare cu cheia egală cu „`recipeIndex`”, valoarea variabilei „`aparitii`” și se execută `break`. În continuare, algoritmul trece la verificarea următorului cuvânt introdus de utilizator cu toate ingredientele rețetei curente. După execuția `for-urilor`, în `hashMap` la cheia ce specifică rețeta curentă se va găsi numărul total de ingrediente conținute de rețetă ce corespund cu ingredientele căutate de utilizator. Se revine în funcția „`handleSearch`” și se execută acest proces pentru fiecare rețetă în parte. La final-ul `for-ului` ce apelează funcția de comparare, variabila „`hashMap`”, va conține pentru fiecare rețetă, deci la fiecare cheie, numărul de apariții total al ingredientelor căutate în ingredientele rețetei. În continuare în funcția „`handleSearch`”, se sortează numeric „`hashMap`” în funcție de apariții și se stochează rezultatul în „`sortedMap`”. În momentul de față, se cunosc id-urile rețetelor ce conțin toate ingredientele căutate de utilizator, apoi acelea ce conțin toate -1 ingrediente, ș.a.m.d, până la rețetele ce conțin doar unul din ingredientele specificate. La final, este setată variabila „`newTotalRecipes`” cu size-ul mapei și „`isSearchFinished`” la `true`, pentru a specifica finalizarea procesului de căutare.

Al treilea `useEffect` implementat este recalculat la fiecare schimbare a variabilei „`isSearchFinished`”. Acesta constă în preluarea doar acelor rețete regăsite în variabila „`sortedMap`” și afișarea lor exact în ordinea apariției. Se iterează prin fiecare element al mapei și se verifică fiecare rețetă din tabloul „`allRecipes`”, pentru identificarea id-urilor rețetelor care corespund cu cheia rețetei curente din mapă. La final, noile rețete de afișat sunt stocate în variabila „`filteredRecipes`”.

Conținutul acestei pagini se modifică în trei cazuri. Primul conținut este accesat când variabila „`isSearchClicked`” este `true` (butonul de căutare a fost apăsat), variabila „`newTotalRecipes`” este mai mare decât 0 și variabila „`replay`” este `false`. Codul HTML, mapează array-ul format în urma căutării, „`filteredRecipes`” și pentru fiecare rețetă apelează „`FindRecipesHashMap.tsx`” cu cei doi parametri: obiectul curent rețetă și valoarea variabilei „`aparitii`” corespunzătoare în mapă. Fișierul de afișare a unei rețete este foarte asemănător cu cel prezentat anterior, însă adăugă în plus, în interiorul cardului, numărul de ingrediente conținute de rețetă care corespund cu ingredientele căutate de utilizator. În caz că, numărul de apariții este 1 se afișează textul următor: „Apare un ingredient căutat.” Acest conținut poate fi observat în cadrul figurii cu numărul 23.

Al doilea caz în care se modifică pagina, apare când: variabila care determină starea butonului de căutare este `true` (buton apăsat), variabila „`isSearchFinished`” este `true` (proces căutare finalizat), variabila „`newTotalRecipes`” este zero (nu a fost găsită nicio rețetă care să corespundă cu ingredientele căutate) și variabila „`replay`” este tot falsă. Acest conținut se poate observa în figura cu numărul 24. Aici există un buton cu proprietate `onClick` ce apelează funcția „`handleReplay`” care setează variabila „`replay`” la `true`.

În final, dacă pentru primele două cazuri nu au fost îndeplinite condițiile, atunci se va afișa cazul trei. Acest conținut reprezintă atât starea de început, cât și starea accesată după `replay`. Conținutul este implementat exact ca în cazul opțiunii „Caută rețete” care poate fi observată în figura 21. În cadrul HTML-ului, se mapează tabloul „`recipes`” și pentru fiecare rețetă se apelează componenta „`FindRecipe`”.

#### Pagina de favorite

Această pagina a fost creată doar pentru o vizualizare mai grupată a rețetelor favorite. Paginile „Caută rețete” și „Refood’s Feature” sunt singurele ce pot fi folosite pentru gestionarea favoritelor ( adăugare sau ștergere ), însă modul de afișare foarte dispersat nu este accesibil pentru observarea tuturor acestor favorite.

Rețetele dumneavoastră favorite:



Figura 25: Opțiunea „Rețete Favorite”

Această pagina are propriul folder denumit „FavoritePage” în care există doar fișierul principal ce creează conținutul acestei pagini, „FavoriteShow”. Conținutul paginii, când user-ul curent nu are favorite adăugate încă (adică nu există variabila „favorites” sau „favorites.length” este 0), permite apăsarea butonului „Adaugă Favorite” ce îl redirecționează către meniul de rețete observat în figura 21, unde va putea să observe toate rețetele și să-și aleagă favorite.



Figura 26: Pagina când utilizatorul nu are nicio favorită

În caz contrar, în codul HTML ce afișează conținutul paginii, se mapează tabloul „favorites” și pentru fiecare element se creează un card unic care afișează numele, categoria și butonul de accesare a detaliilor rețetei. Array-ul „favorites” este populat în singurul useEffect din acest fișier care se recalculează la fiecare schimbare a paginii curente și a variabilei „authState”. Funcția de efect verifică dacă utilizatorul este autentificat și, în caz afirmativ al condiției, se creează o cerere către adresa endpoint-ului „api/favorites/search/findRecipesByEmail”, specificându-se token-ul utilizatorului, pagina curentă și size-ul (numărul de favorite afișat de pagină).

## Pagina forumului Relife

Folosind această pagină, utilizatorii pot crea noi anunțuri cu donații și pot observa toate postările create de toți userii. În cadrul opțiunii „Vezi Forum”, utilizatorul curent poate șterge doar anunțurile făcute de el și nu poate vizualiza detaliile de preluare ale anunțurilor care nu îi aparțin. Această pagină își va îndeplini funcționalitatea când aplicația va fi scoasă pe piață și se vor căuta parteneri sau centre de donații care să vizualizeze toate anunțurile, să preia donațiile din locurile de preluare specificate și să le distribuie.

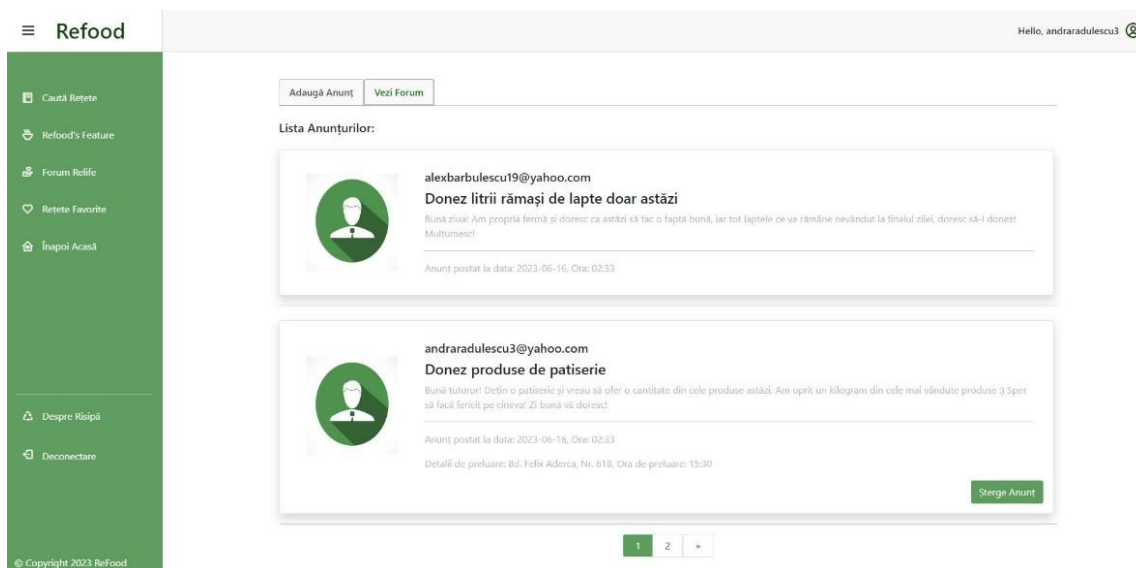


Figura 27: Opțiunea „Vezi Forum”

Pagina a fost creată folosind folderul „ForumRelifePage” ce conține fișierul principal care afișează taburile de navigare din partea de sus a paginii. „Adaugă Anunț” și „Vezi Forum” sunt două butoane care setează „isSeeForumClicked” la false, respectiv true folosind proprietatea onClick. În funcție de această variabilă se afișează fie conținutul fișierului „AddForum.tsx”, fie „SeeForum.tsx” care se regăsesc în folderul „elements”. La accesarea paginii, butonul „Vezi Forum” este setat activ și variabila corespunzătoare apăsării acestuia este true pentru a se afișa toate anunțurile din cadrul forumului.

## Opțiunea de vizualizare a donațiilor

Folosind fișierul „SeeForum”, sunt implementate două useEffect-uri pentru preluarea datelor necesare acestei opțiuni. Prima funcție de efect, denumită „fetchDonations”, creează o cerere către adresa endpoint-ului „api/donations”. Se așteaptă răspunsul serverului și se stochează rezultatul folosind metodele explicate anterior în acest capitol. Cu ajutorul acestei cereri, sunt accesate toate anunțurile de donații existente în format ce permite paginare și care populează array-ul „donations”, folosind funcția de actualizare de stare. Dacă transferul de date între server și client s-a realizat cu succes, se verifică variabila „isDelete”. În caz că este true, se setează pagina curentă la 1 și se reinițializează „isDelete” la false. Acest useEffect se recalculează la fiecare modificare a paginii curente și a variabilei „isDelete”.

A doua funcție de stare, numită „fetchUserDonations”, accesează toate donațiile postate de utilizatorul curent. Înainte de crearea cererii, se verifică variabila „authState” și proprietatea acesteia .isAuthenticated să nu fie nule. În urma îndeplinirii condiției, se execută request-ul către adresa „api/donations/search/findDonationsByEmail” prin specificarea în url a token-ului de autentificare. După preluarea răspunsului, datele rezultate sunt stocate în tabloul „userDonations”. Acest proces este recalculat automat la fiecare schimbare a variabilelor „donations” și „authState”.

A treia cerere de date, se realizează folosind funcția asincronă „deleteDonation” ce conține, drept parametru, id-ul donației care trebuie ștearsă la cererea utilizatorului curent. Se implementează o cerere DELETE către endpoint-ul „api/donations/secure/delete”, specificând în cadrul url-ului, id-ul postării și în headers token-ul de autentificare al utilizatorului care a executat ștergerea. Butonul de ștergere din cadrul fiecărui anunț este afișat folosind funcția „isDonationMade”. Cu ajutorul acesteia, se verifică fiecare donație din totalitatea postărilor cu fiecare element conținut de tabloul „userDonation”. Dacă se găsește o potrivire între cele două id-uri, înseamnă că donația în cauză a fost creată de utilizatorul curent, deci i se permite ștergerea acesteia. Butonul folosește proprietatea onClick pentru apelarea „handleClickErase” cu parametrul id-ul donației care a îndeplinit condiția anterioară. În continuare, această funcție apelează și așteaptă finalizarea execuției „deleteDonation”, iar la final, setează „isDelete” true. Conținutul paginii pentru această opțiune, dacă nu se găsește niciun anunț postat, afișează următorul mesaj ce poate fi observat în figura de mai jos.

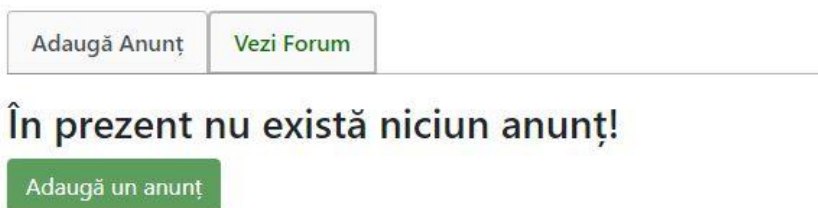


Figura 28: Pagina când nu există donații

#### Opțiunea de adăugare a unei donații

În cadrul acestei opțiuni, se realizează procesul de postare a unui nou anunț care poate fi vizualizat în timp real în forum. Pentru adăugarea unui nou anunț și formularea unei cereri corespunzătoare, a fost necesară crearea unei clase „DonationRequest” ce conține atributele setate folosind interfața: titlu, detaliiPreluare și descriere.

Fișierul „AddForum.tsx” implementează funcția asincronă „submitDonation” care creează o cerere către endpoint-ul cu calea „api/donations/secure/create”. Aici, se verifică dacă utilizatorul este autentificat și dacă titlul și detaliile de preluare pentru noul anunț au fost introduse( doar aceste elemente sunt obligatorii, descrierea fiind opțională ). Condiția validă implică specificarea opțiunilor cererii, prin trimiterea în headers a token-ului de autentificare și includerea în body a obiectului „donationRequest” creat la introducerea informațiilor anunțului.

Se așteaptă răspunsul și, dacă este valid, se resetează variabilele care au stocat atributele noului anunț și se execută `setInvalid(false)`, `setValid(true)`. În caz contrar, ultimele două variabile sunt setate în mod opus.

Codul HTML, verifică dacă variabila „valid” este true și afișează mesajul de succes al adăugării anunțului: „Anunț adăugat cu succes !!!”. În schimb, dacă variabila „invalid” este true, se afișează mesajul ce poate fi observat în figura numărul 29.

Figura 29: Opțiunea „Adaugă Anunț”

## Evaluarea lucrării

În urma finalizării procesului de implementare a aplicației, este necesară desfășurarea unei evaluări a rezultatelor dobândite. Evaluarea este o etapă necesară, cu ajutorul căreia se identifică performanța și calitatea obținute din punct de vedere al funcționării aplicației, al aspectului interfeței și al experienței oferite utilizatorului. Un alt scop al evaluării este determinarea posibilelor limitări sau puncte slabe ale implementării cu scopul remedierii acestora în viitoarele dezvoltări. În cadrul acestui capitol, se vor evalua și se vor expune toate aspectele menționate anterior pentru fiecare funcționalitate implementată în parte.

### Funcționalitatea de căutare a rețetelor

Prima funcționalitate implementată servește drept meniu virtual pentru utilizator și poate fi observată în figura 17. Punctele forte ale căutării rețetelor după titlu sunt următoarele: se poate realiza folosind secvențe care sunt conținute în interiorul titlului și nu necesită folosirea diacriticelor. Limitările descoperite sunt referitoare la numărul mic de rețete adăugate în baza

de date și setul redus de caracteristici al unei rețete. O posibilă îmbunătățire constă în adăugarea mai multor informații despre rețete, pe baza cărora se vor putea introduce noi metode de filtrare. Spre exemplu, rețetele puteau fi sortate după regiunea de proveniență sau cantitatea de calorii pe porție. De asemenea, acestea puteau să fie afișate în funcție de gradul de dificultate sau numărul de ingrediente. Un filtru foarte important care trebuia luat în considerare, dar nu a fost implementat, este specificarea rețetelor vegane sau vegetariene pentru a include cât mai multe din regimurile alimentare existente.

#### Funcționalitatea de afișare a detaliilor rețetelor

În cadrul acestei funcționalități, pentru o experiență completă a utilizatorului la încercarea unei rețete, se puteau adăuga imagini cu procesul de gătit a rețetei care să corespundă cu rezultatul fiecărui pas din modul de realizare al acesteia. Fiecare pas de gătit putea avea asociat un timp ce putea fi pornit și oprit în mod interactiv. Spre exemplu, utilizatorul putea pune tava în cuptor și pornea cronometru pentru timpul de execuție al pasului din aplicație pentru a îl notifica la finalizarea pasului. O alternativă la această funcționalitate este afișarea unui video în pagină care să demonstreze cum trebuie executată gătirea. În plus, rețetele puteau avea cantități ale ingredientelor care să se modifice în funcție de numărul de porții dorit. Astfel, utilizatorii ar fi putut alege opțiunea dorită și potrivită de porții pentru a vedea cantitățile corespunzătoare de ingrediente necesare. De asemenea, pentru componenta de creare a unui nou comentariu din cadrul acestei pagini se putea adăuga metodă de ștergere.

#### Funcționalitatea de căutare după ingrediente

Cu ajutorul acestei funcționalități, utilizatorilor le sunt sugerate în timp real rețete care conțin ingredientele introduse de aceștia. Printre atuurile algoritmului de căutare implementat în aplicație se regăsesc adaptabilitatea procesului la cuvinte fără diacritice, timpul scurt de execuție al procesului și afișarea instantanee a rezultatelor. Unul dintre punctele slabe în cazul căutării este faptul că utilizatorii nu pot selecta și cantitatea deținută din ingredientul introdus. Să presupunem scenariul în care un consumator deține 600 de grame de căpșuni și dorește să le folosească într-o rețetă de desert pentru a nu se strica. Folosind aplicația, acesta ar afla că există trei rețete care folosesc acest ingredient, însă după vizualizarea detaliilor, observă faptul că pentru cantitatea necesară a ingredientului din toate rețetele rezultate este necesar un kilogram. Prin urmare, consumatorul este nevoit să achiziționeze în plus ingrediente și se reduce, astfel, eficiența aplicației de a reduce risipa de alimente. În cadrul algoritmului implementat pentru căutare, o îmbunătățire majoră ar fi constat în afișarea cu prioritate a rețetelor ce conțin cele mai multe din ingredientele căutate cu dată de expirare mică. Pentru a se îndeplini obiectivul de evitare a risipei, trebuie să se asigure folosirea cu prioritate a alimentelor perisabile.

#### Funcționalitatea de vizualizare a forumului Relife

Anunțurile din cadrul aplicației sunt afișate într-un format ușor de urmărit care susține confidențialitatea datelor prin ascunderea adreselor de preluare a donațiilor. O posibilă îmbunătățire a procesului este implementarea unui cont de administrator care să vizualizeze aceste anunțuri și să poată observa, pentru fiecare postare, adresa de preluare. Administratorii ar putea fi voluntari sau angajați ai firmelor partenere care preiau donațiile de la locațiile specificate de utilizatori în anunțurile efectuate și le livrează către centre și agenții de donații.



De asemenea, în cazul acestei funcționalități, nu s-a luat în calcul posibilitatea în care donatorii nu vor să ofere o adresă de preluare, deoarece doresc să aducă donația singuri către centrele de donații. Prin urmare, se putea adăuga o pagină de informare cu privire la adresele centrelor de donații sau organizațiilor unde se pot realiza donații de alimente.

## Retrospectiva și Concluziile Lucrării

În acest capitol, se vor restrange ideile dezvoltate pe parcursul lucrării cu scopul formulării unei sinteze ce păstrează esența procesului desfășurat pentru crearea aplicației web. Se vor trece în revistă răspunsurile la următoarele întrebări: Ce reprezintă aplicația creată? În urma implementării s-au obținut rezultatele dorite din punct de vedere al cerințelor și al performanțelor? Ce cunoștințe au fost necesare să fie dobândite pe parcursul proiectului sau să fie antrenate pentru implementarea aplicației? Ce impact are aplicația dezvoltată în domeniu evitării sau reducerii risipei alimentare și prin ce se remarcă față de soluțiile deja existente pe piață?

Aplicația implementată, Refood, este o aplicație de gătit menită să reducă risipa alimentară prin căutarea în timp real a rețetelor după ingredientele deținute de utilizator și introduse de acesta în cadrul aplicației. Aceasta prezintă și funcționalitatea de căutare a rețetelor după categorie sau după titlu și adăugarea acestora la favorite. În cadrul platformei, se pot realiza donații de mâncare de la utilizatori și este pusă la dispoziție o pagină de informare asupra risipei alimentare și metodelor sau practicilor de reducere a acesteia.

Dacă analizăm obiectivele propuse în cadrul primului capitol și funcționalitățile obținute în urma implementării, se poate observa că aplicația creată urmărește setul de cerințe specificate păstrând scopul principal de reducere a risipei alimentare. Totuși, urmărind evaluarea realizată în capitolul anterior se poate afirma faptul că, în cazul unei lansări, anumite părți limitate din cadrul funcționalităților riscă să nu-și atingă potențialul maxim fără adăugarea îmbunătățirilor descoperite.

Crearea acestei aplicații a necesitat dezvoltarea cunoștințelor legate de modelele arhitecturale existente și înțelegerea procesului de comunicare între componente cu scopul proiectării propriiei arhitecturi. În ceea ce privește limbajele și tehnologiile folosite pentru crearea aplicației web din cadrul acestei lucrări, am urmărit dobândirea de noi competențe prin utilizarea unor limbaje necunoscute.

Având în vedere toate datele expuse în această lucrare și ținând cont de soluțiile existente în domeniu dezvoltate în al doilea capitol, dar și de funcționalitățile obținute ale aplicației proiectate și dezvoltate, se poate afirma faptul că platforma web Refood reprezintă o inovație pe piața actuală ce se remarcă prin îmbinarea caracteristicii de căutare a rețetelor în timp real după ingrediente cu procesul online de realizare a donațiilor alimentare.

## Bibliografie

1. Angelique Chrisafis, French law forbids food waste by supermarkets, 2016 02 04, Link: <https://www.theguardian.com/world/2016/feb/04/french-law-forbids-food-waste-by-supermarkets>, Accesat la data: 05 2023
2. Bezkoder, JPA One To Many example with Hibernate and Spring Boot, 2023 06 16, Link: <https://www.bezkoder.com/jpa-one-to-many/>, Accesat la data: 06 2023
3. CSS Introduction, Link: [https://www.w3schools.com/css/css\\_intro.asp](https://www.w3schools.com/css/css_intro.asp), Accesat la data: 06 2023
4. Chirsanova, Aurica, and Dumitru Calcatiniuc. "The impact of food waste and ways to minimize IT." Journal of Social Sciences 1.4 (2021): 128-139.
5. Gabriel Sescu, Banca pentru Alimente, Link: <https://bancapentrualimente.ro/contact/>, Accesat la data: 05 2023
6. Get started with Bootstrap, Link: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>, Accesat la data: 06 2023
7. Getting Started, Link: <https://legacy.reactjs.org/docs/getting-started.html>, Accesat la data: 06 2023
8. Gutierrez, Felipe, Gutierrez, and Anglin. Pro Spring Boot 2. Apress, 2019.
9. Holmberg, Oliver. "Migrating from JavaScript to TypeScript and its advantages." (2023).
10. JavaScript HTML DOM, Link: [https://www.w3schools.com/js/js\\_htmldom.asp](https://www.w3schools.com/js/js_htmldom.asp), Accesat la data: 05 2023
11. Love Food Hate Waste, Link: <https://www.lovefoodhatewaste.com/>, Accesat la data: 05 2023
12. Ministerul Agriculturii și Dezvoltării Rurale, 29 septembrie - Ziua Internațională de conștientizare asupra risipei alimentare , Link: <https://www.madr.ro/risipa-alimentara/29-septembrie-ziua-internationala-de-constientizare-asupra-risipei-alimentare.html/>, Accesat la data: 05 2023
13. Okta Help Center May 9, 2023, Link: [https://support.okta.com/help/s/article/what-is-okta?language=en\\_US](https://support.okta.com/help/s/article/what-is-okta?language=en_US), Accesat la data: 05 2023
14. Pereira, Anil L., Mehdi Raoufi, and Jerrod C. Frost. "Using MySQL and JDBC in new teaching methods for undergraduate database systems courses." Data Engineering and

Management: Second International Conference, ICDEM 2010, Tiruchirappalli, India, July 29-31, 2010. Revised Selected Papers. Springer Berlin Heidelberg, 2012.

15. Priyesh Patel, What exactly is Node.js?, Link: <https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/>, Accesat la data: 06 2023

16. Rawat, Prateek, and Archana N. Mahajan. "ReactJS: A modern web development framework." International Journal of Innovative Science and Research Technology 5.11 (2020): 698-702.

17. Start a New React Project, Link: <https://react.dev/learn/start-a-new-react-project>, Accesat la data: 06 2023

18. Teo's Kitchen, Link: <https://teoskitchen.ro/>, Accesat la data: 05 2023

19. Too Good To Go, Link: <https://www.toogoodtogo.com/>, Accesat la data: 05 2023

20. UNEP Planetary Action 2021, Link: <https://www.unep.org/annualreport/2021/index.php>, Accesat la data: 05 2023

21. Walls, Craig. Spring Boot in action. Simon and Schuster, 2015.