UNIVERSITATEA POLITEHNICA BUCUREŞTI
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECT DE DIPLOMĂ

Sistem Smart Mix pentru rețete și ingrediente

Ioan Constantin

Coordonator ştiinţific:

Prof. dr. ing. Nirvana Popescu

BUCUREŞTI

2020

CUPRINS

SINOPSIS

Tuturor ni se întâmplă să rămânem fără idei când vrem să preparăm ceva cu ingredientele pe care le avem, iar căutarea pe site-urile culinare durează mult, rezultatele nu sunt optime și cumpărarea celorlalte ingrediente necesare poate fi mai rapidă și mai convenabilă folosind acest proiect. Proiectul este o aplicație web in care utilizatorul introduce ingredientele pe care le are la dispoziție și aplicația filtrează după acestea si întoarce rețetele care conțin ingredientele cele mai apropiate de cele căutate de utilizator, iar pentru rețetele care mai necesită câteva ingrediente i se arată utilizatorului, folosind locația acestuia, cel mai scurt și cel mai ieftin drum (folosindu-se locațiile magazinelor si verificându-se dacă au acele ingrediente în stoc).

În plus, utilizatorul adaugă rețete în secțiunea de favorite și, utilizând această secțiune și paginile pe care utilizatorul le accesează frecvent, le trimite mesaje pe chat utilizatorilor care au aceleași preferințe culinare. Utilizatorului i se recomandă cele mai vizualizate și cele mai apreciate (după notă) rețete și există un tab cu rețete noi. De asemenea, când utilizatorul este lângă o cofetărie sau patiserie care are în stoc un produs pe care utilizatorul l-a adăugat la favorite i se trimite utilizatorului o notificare.

ABSTRACT

There is a common problem which appears when people try to prepare something only with the ingredients which are already bought. The search on the cooking websites takes a long time, the results aren't optimized and you can buy the other ingredients which are needed faster and cheaper using this project. The project is a web application, the user can enter the ingredients which are already taken and the application filters by these ingredients and returns the recipes which are taken from other sites. For the recipes which contain more ingredients than the existing ingredients, the user will see the shortest and the cheapest way to purchase these ingredients using his / her location and the location of the shops.

The user can also add recipes to the "Favourites" section and, using this section and the pages which are frequently visited by this user, he sends and receives messages to and from other users which have the same preferences as his / hers. The user can see the most visited and the most appreciated (by rating) recipes. When the user is close to a cake shop or to a pastry which has a product which is also in the user's "Favourites" section, the user receives a notification.

1 INTRODUCERE

1.1 Context

Conform stării actuale a pieței (subiect pe care l-am abordat mai jos într-un capitol separat în această lucrare), website-urile care conțin rețete sunt în continuă expansiune (website-urile din acest domeniu sunt din ce in ce mai diversificate și au un număr de vizitatori care continuă să crească). Problema principală este faptul că aceste website-uri nu oferă toate informațiile de care un utilizator are nevoie pentru a prepara acea rețetă.

Spre deosebire de aceste website-uri aplicația Smart Mix îi oferă utilizatorului toate detaliile, astfel încât acesta să poată prepară rețeta într-un timp mult mai scurt și îi oferă posibilitatea de a discuta cu utilizatori care au aceleași preferințe culinare ca și el (compatibilitatea culinară a utilizatorilor fiind stabilită după un algoritm care include secțiunile de favorite ale acestora și paginile cele mai accesate de către respectivii utilizatori).

Aplicația Smart Mix îi oferă utilizatorului cel mai scurt drum către magazinele care au în stoc produsele necesare pentru a prepara rețeta selectată de utilizator și cel mai ieftin drum (ținând cont de combustibilul pe care vehiculul utilizatorului îl consumă sau de mijlocul de transport pe care utilizatorul îl selectează în aplicație). De asemenea, aplicația actualizează în timp real stocurile magazinelor astfel încât dacă un produs se epuizează traseul utilizatorului va fi reconfigurat pentru a evita acel magazin.

1.2 Problema

Problemele pe care le rezolvă aplicația Smart Mix sunt reprezentate de faptul că oamenii nu cumpără mereu cele mai ieftine produse și nu găsesc cele mai scurte trasee pentru a cumpăra produsele necesare preparării rețetei preferate. În plus, în cazul website-urilor existente utilizatorii comunica între ei postând comentarii atașate fiecărei rețete. În cazul aplicației Smart Mix, algoritmul care stabilește compatibilitatea culinară a utilizatorilor este folosit pentru a facilita comunicarea publică și privată între utilizatori. Astfel, utilizatorii cu aceleași pasiuni culinare pot forma mai ușor o comunitate.

1.3 Obiective

Obiectivele acestui proiect sunt cele de a ghida utilizatorul prin toți pașii preparării unei rețete (inclusiv cumpărarea ingredientelor), de a îl asigura pe utilizator de faptul că alege cele mai convenabile opțiuni (atât din punct de vedere al timpului consumat, cât și din punct de vedere al banilor) și de a îi oferi utilizatorului o rețea de socializare prin care poate comunica cu ceilalți utilizatori în funcție de preferințele culinare ale acestora.

1.4 Soluția propusă

Pentru a găsi traseele cele mai scurte și cele mai ieftine se folosesc algoritmi pentru calculul drumului minim cu costurile fiind reprezentate de metri, respective de lei. Pentru respectarea distanțelor exacte este folosit Google Maps APi. Chat-ul între utilizatori este realizat folosind WebSockets, iar compatibilitatea culinară a utilizatorilor este stabilită folosind un algoritm care calculează numărul de pagini comune pe care cei 2 utilizatori le accesează frecvent (acestea sunt determinate cu ajutorul Google Analytics) si numărul de rețete comune din secțiunile de favorite ale celor 2 utilizatori.

Logarea se poate realiza folosind un cont care a fost anterior înregistrat (este folosită o baza de date MySQL care este legată de aplicație folosind MySQL JDBC Driver) sau utilizând contul de Linkedin (pentru ca dacă un utilizator este food blogger ceilalți utilizatori să poată vizualiza și profilul său profesional). În plus, utilizatorii pot vedea media notelor acordate fiecărei rețete (dacă sunt sub 10 note, media nu va fi afișată pentru că setul de date nu este suficient de mare), iar fiecare utilizator are atașată media notelor pe care le-a acordat (dacă utilizatorul a acordat mai mult de 10 note minime consecutive notele sale nu vor mai fi luate în considerare).

1.5 Rezultatele obținute

Plusurile acestui produs

Acest produs va îngloba toate rețetele de pe cele mai cunoscute site-uri culinare cu domeniul .ro și va avea 3 funcționalități în plus față de cele mai cunoscute site-uri din România și din străinătate:

1. Pentru rețetele care mai necesită câteva ingrediente în afara celor introduce de utilizator i se arată acestuia, folosind locația sa, cel mai scurt și cel mai ieftin drum pe

care trebuie să îl parcurgă pentru a le cumpăra (folosindu-se locațiile magazinelor si verificându-se dacă au acele ingrediente în stoc).

2. Utilizatorul va avea un rating de compatibilitate culinară cu un alt utilizator (rating care se calculează în funcție de secțiunile de favorite ale celor 2 utilizatori și de paginile pe care aceștia navighează frecvent), iar, bazându-se pe acest rating, utilizatorii cu aceleași preferințe culinare își pot trimite mesaje private unul altuia.
3. Când utilizatorul este lângă o cofetărie sau patiserie care are în stoc un produs pe care utilizatorul l-a adăugat la favorite i se trimite utilizatorului o notificare.
4. Afișarea mediei notelor acordate de fiecare utilizator (pentru a vedea cât de sever este utilizatorul în evaluarea rețetelor).
5. Afișarea celor mai ieftine rețete pe care le poate prepara utilizatorul bazându-se pe costul ingredientelor rămase de cumpărat.

1.6 Structura lucrării

În capitolul 3. Analiza cerințelor / Motivație sunt detaliate anumite funcționalități ale aplicației care trebuie să acopere ceea ce își doresc utilizatorii tintă ai aplicației (acestea sunt justificate prin prezentarea unor studii care arată ce își doresc utilizatorii website-urilor din acest domeniu). Capitolul 4. Studiu de piață / Metode existente conține subcapitolele 1. “Starea actuală a pieței” (în acest subcapitol sunt analizate website-urile din același domeniu din punct de vedere al funcționalităților oferite de acestea și al numărului de utilizatori frecvenți ai acestora), 2. “Descrierea arhitecturii generale a aplicației”, 3. “Alternative existente” (în acest subcapitol sunt prezentate alternativele și sunt justificate alegerile făcute în timpul implementării aplicației) si 4. “Algoritmi și arhitecturi asemănătoare din aplicații asemănătoare din articole științifice publicate”. În capitolul 5. Soluția propusă este explicată structura aplicației, sunt detaliați algoritmii folosiți și sunt adăugate mai multe diagrame pentru a ilustra relațiile dintre entități (În subcapitolul 4. “Algoritmi folosiți în aplicație”). În capitolul 6. Detalii de implementare sunt prezentate principalele părți din aplicație. În capitolul 7. Evaluare este analizată aplicația din punct de vedere al performanței, iar documentul se încheie cu capitolul 8. Concluzii.

2 MOTIVAȚIA ALEGERII TEMEI

Conform [1] în anul 2018 americanii plăteau în medie 20.37 dolari pentru un prânz livrat de un restaurant, 12.53 dolari pentru mâncare semi-preparată livrată acasă și doar 4.31 dolari pentru un prânz gătit acasă.

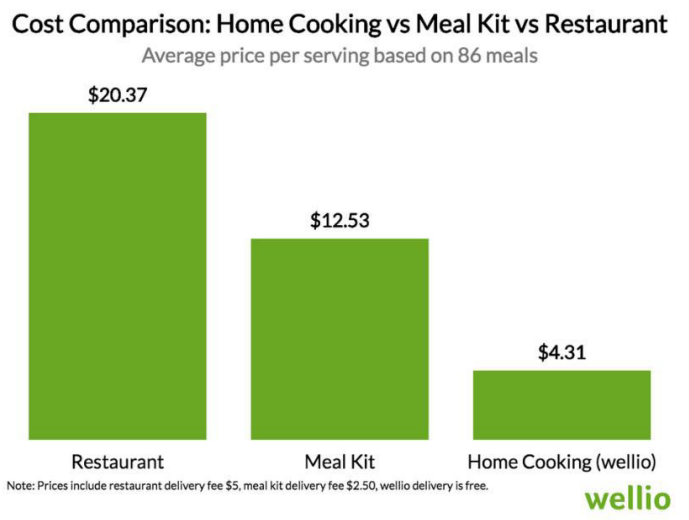


Figura 1. Cât plăteau americanii în anul 2018 pentru fiecare tip de prânz [1]

Conform [2] (sondajul a fost efectuat pe un eșantion de 502 persoane) în anul 2016 36% dintre americani găteau zilnic acasă, 50% dintre americani (și 58% dintre americanii angajați cu program full-time) găteau acasă intre 3 și 6 zile pe săptămână și doar 14% dintre americani găteau acasă mai puțin de 3 zile pe săptămână.

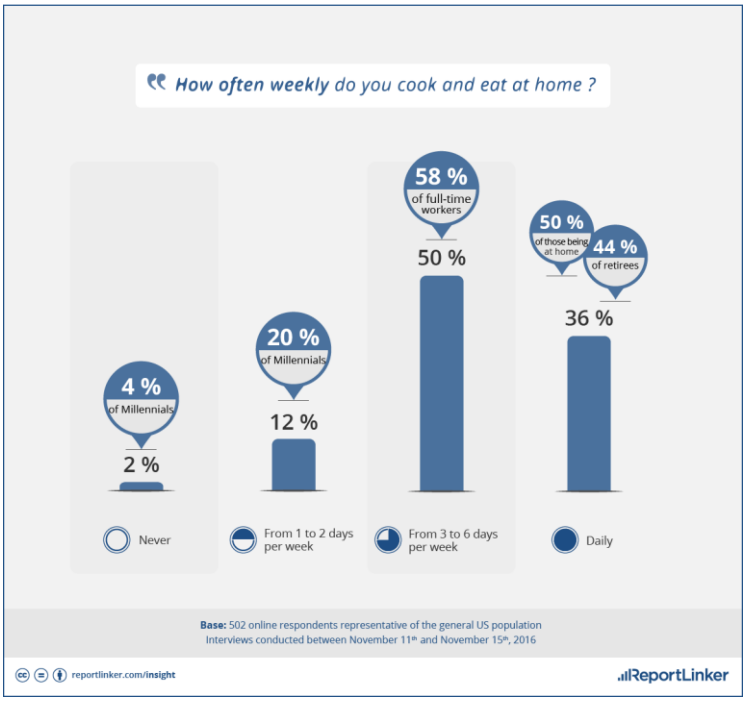


Figura 2. Câte zile pe saptămană găteau americanii acasă în anul 2016 [2]

Din corelarea celor 2 studii de mai sus se poate deduce că principalul motiv pentru care oamenii aleg să gătească acasă este prețul mai mic, ceea ce este arătat în partea secundară a sondajului al doilea. 31% dintre respondenți au ales acest motiv, 22% au ales gătitul pentru că este mai sănătos, 18% au spus că gătitul le oferă mai multa libertate de alegere, iar 13% au spus că gătesc din pasiune.

După analizarea acestor răspunsuri am introdus în aplicație funcționalități care să le garanteze utilizatorilor cel mai mic preț (prin afișarea celui mai ieftin drum pentru a cumpăra ingredientele rămase) și rețete sănătoase (exista secțiuni speciale pentru rețete fară gluten și cu conținut redus de zaharuri).

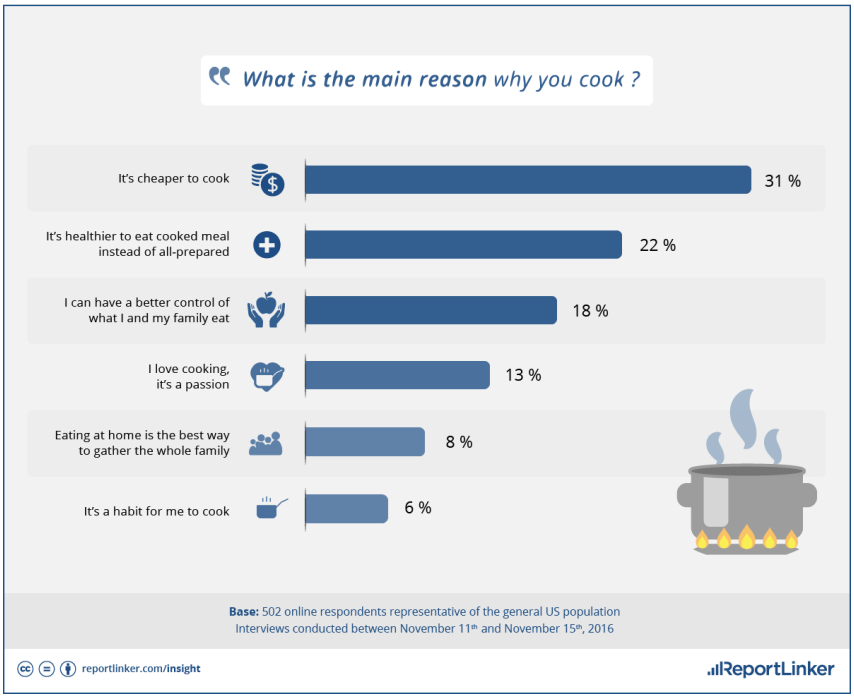


Figura 3. Motivele pentru care gateau americanii acasă în anul 2016 [2]

O altă întrebare din acest sondaj care a influențat funcționalitățile aplicației este "De unde te inspiri atunci când gătești?". 28% dintre respondenți au spus că se inspiră din rețetele de familie, 20% au afirmat că prepară cu ingredientele pe care le au deja acasă, 17% citesc website-uri de gătit, iar 12% gătesc aceeași rețetă mereu.

Aplicația Smart Mix acoperă 3 dintre cele 4 categorii de utilizatori de mai sus (prima categorie nu este acoperită de Smart Mix, utilizatorii din cea de-a doua categorie pot introduce ingredientele și Smart Mix filtrează după ele și utilizatorii din a patra categorie pot adăuga în secțiunea de favorite o rețetă pe care o prepară mereu pentru a o accesa mai ușor).

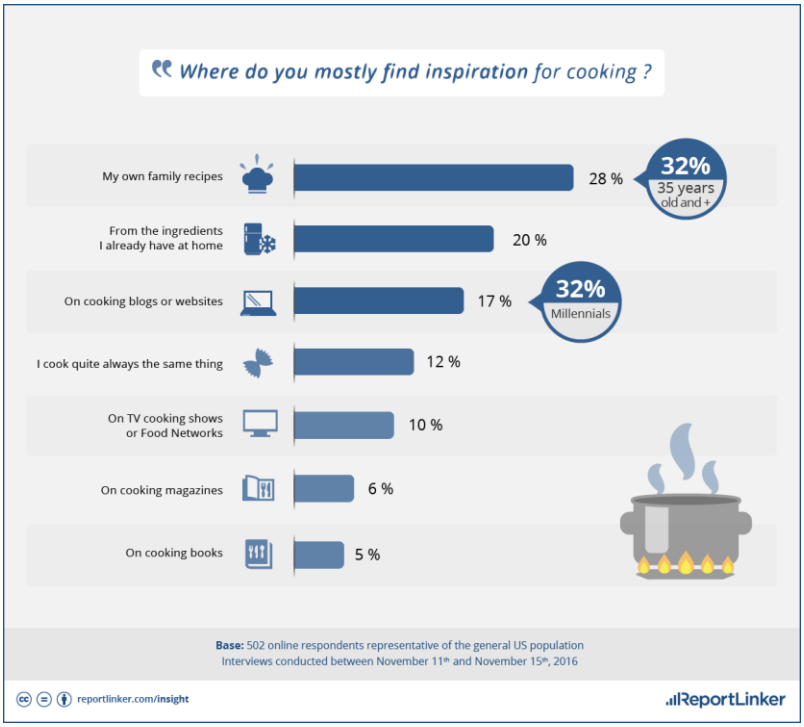


Figura 4. De unde se inspirau americanii atunci cand găteau acasă în anul 2016 [2]

O altă întrebare din sondajul [2] este “Cât de bine vă pricepeți la gătit?”. 65% dintre americani au afirmat că se consideră la un nivel intermediar (70% dintre cei care gătesc între 3 și 6 zile pe săptămana), 23% au spus că sunt experți (28% dintre cei care folosesc rețete proprii), iar 12% au spus că sunt începători (36% dintre cei care gătesc una sau două zile pe saptămână). De aici reiese faptul că respondenții care se consideră experți (23%) nu utilizează prea mult website-urile de gătit, în timp ce restul (77%) folosesc mai mult website-urile de gătit și, fiind începători sau intermediari, au nevoie de un ghid detaliat care să le exemplifice toți pașii, ceea ce oferă Smart Mix.

Conform [3] în anul 2015 bărbații și femeile găteau în medie 43, respective 70 de minute pe zi. Aceeași întrebare a fost pusă și în sondajul de mai sus și 50% dintre respondenți au ales varianta “intre 31 de minute și o oră”, 23% petrec mai mult de o oră gătind, iar doar 3% petrec mai puțin de 15 minute.

Din aceste sondaje reiese faptul că utilizatorii nu au suficient timp să caute în detaliu toate ingredientele. Din această cauză aplicația Smart Mix vine în ajutorul utilizatorilor și, cu ajutorul algoritmilor folosiți, îi ajută pe utilizatori să cumpere ingredientele din locurile cele mai convenabile pentru ei în funcție de ce opțiuni au selectat aceștia în aplicație.

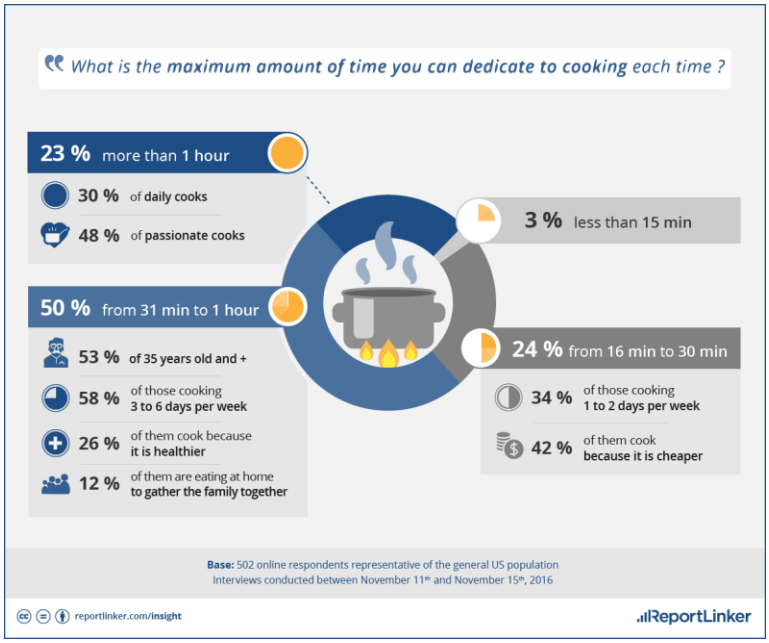


Figura 5. "Care este perioada maximă de timp pe care o dedicați gătitului de fiecare dată?" (Sondaj realizat în SUA în anul 2016) [2]

3 SOLUȚII EXISTENTE

1. Starea actuală a pieței

Conform site-ului trafic.ro cele mai vizitate 3 site-uri culinare cu domeniul .ro în luna noiembrie 2019 au fost www.gustos.ro (1.531.112 de vizite), www.petitchef.ro (398.559 de vizite) și www.retete-gustoase.ro (53.502 de vizite), iar pe glob, conform site-ului ebizmba.com, cele mai vizitate 3 site-uri cu rețete sunt (ordonate după media vizitatorilor unici pe lună) allrecipes.com (~ 25.000.000), foodnetwork.co.uk (~ 23.500.000) și www.food.com (~ 15.500.000).

Site-ul www.gustos.ro este cel mai vizitat site de rețete culinare din România și le oferă utilizatorilor posibilitatea de a posta rețete care sunt grupate pe mai multe categorii. Există și o secțiune de sfaturi culinare.

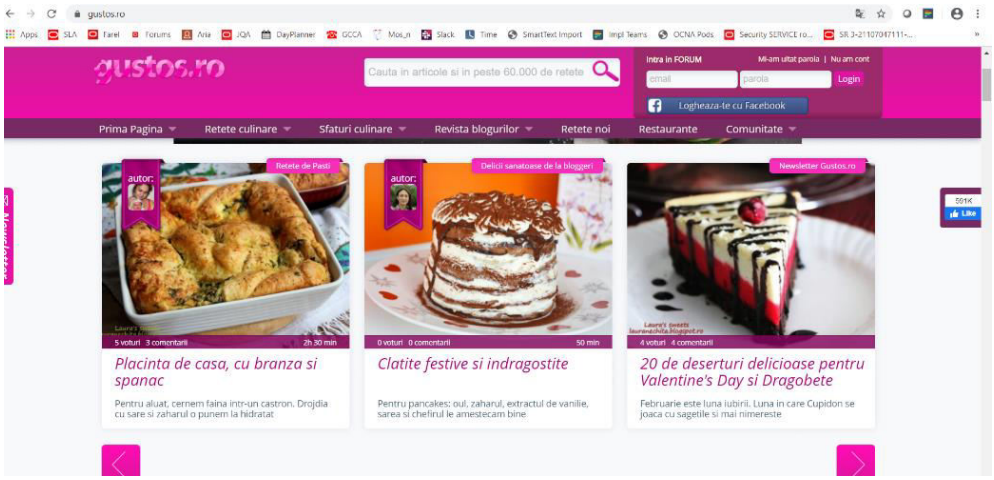


Figura 6. Imagine preluată de pe site-ul www.gustos.ro [4]

Site-ul petitchef.ro le oferă utilizatorilor posibilitatea de a posta rețete (inclusiv de a posta videoclipuri în care prepară acele rețete), are o secțiune numită meniul zilei care înglobează de obicei 3 rețete și utilizatorii pot salva o rețeta în secțiunea “Cartea mea de rețete” și îi pot acorda o notă fiecărei rețete.

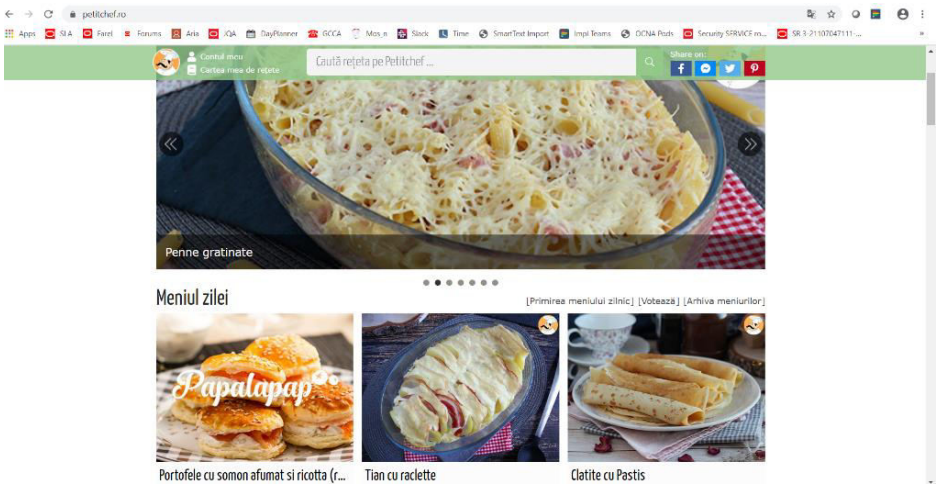


Figura 7. Imagine preluată de pe site-ul www.petitchef.ro [5]

Site-ul allrecipes.com este cel mai vizitat site de rețete culinare din lume. În afară de funcționalitățile pe care le au și cele 2 site-uri descrise mai sus allrecipes.com are o secțiune specială pentru utilizatori în care aceștia sunt învățați să gătească (secțiunea “Cooking school”), oferă posibilitatea căutării filtrate folosind ingredientele și are mai multe filtre bazate pe stilul gastronomic pe care îl are fiecare utilizator (“Quick & Easy”, “Low Calorie”, “Gluten Free” etc.).

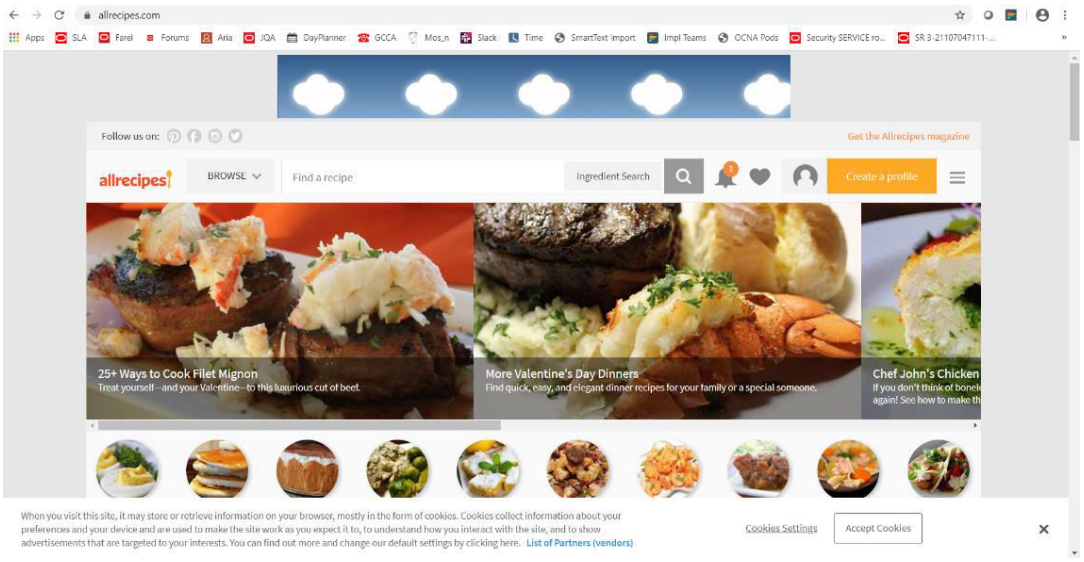


Figura 8. Imagine preluată de pe site-ul www.allrecipes.com [6]

În afară de funcționalitățile oferite de site-ul allrecipes.com site-ul food.com arată în partea dreapta a paginii de start ultimele comentarii adăugate de utilizatori (împreună cu rețeta la care a fost adăugat comentariul) și activitatea recentă de pe site (spre exemplu utilizatorul X îl urmărește pe utilizatorul Y) și are o secțiune cu rețete aflate în trending.

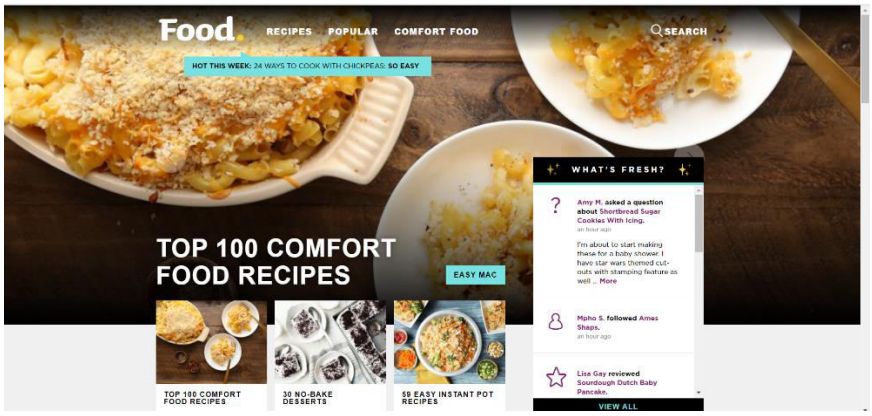


Figura 9. Imagine preluată de pe site-ul www.food.com [7]

Aplicația noastră înglobează toate funcționalitățile site-urilor descrise mai sus și oferă niște funcționalități suplimentare foarte importante : afișarea celor mai ieftine rețete pe care le poate prepara utilizatorul bazându-se pe costul ingredientelor lipsă, cel mai scurt și cel mai ieftin drum pentru ca utilizatorul să cumpere ingredientele pentru o rețetă aleasă de acesta, stabilirea compatibilității culinare între utilizatori și posibilitatea ca utilizatorii să vorbească pe un chat privat, afișarea mediei notelor acordate de fiecare utilizator (pentru a vedea cât de sever este utilizatorul în evaluarea rețetelor; uneori există utilizatori care

acordă 5 stele oricărei rețete și aceștia nu evaluează corect) și primirea unei notificări în momentul în care utilizatorul trece pe lângă o cofetărie care are în stoc un produs aflat în lista sa de favorite.

Conform site-ului nielsen.com 86 de milioane de americani au vizitat site-uri culinare în luna noiembrie 2013, iar în România nu este atât de bine dezvoltată aceasta categorie de site-uri, ceea ce înseamnă că există o oportunitate pe care proiectul o va exploata.

BRAND/WEBSITE	UNIQUE AUDIENCE	AVERAGE TIME SPENT (HH:MM:SS)
All Food and cooking sites	86,350,000	0:25:30
Scripps Networks Food Websites	22,433,000	0:10:37
Allrecipes	17,397,000	0:06:38
About.com Food	9.249,000	0:02:56
Kraft Foods	8,110,000	0:05:38
Betty Crocker	7,554,000	0:05:19
Pizza Hut	6,198,000	0:09:22
Starbucks Coffee	6,164,000	0:08:26
Pillsbury	5,268,000	0:05:45
MyRecipes.com Network	5,167,000	0:07:10
Cooks.com	5,005,000	0:03:51

Figura 10 Topul utilizatorilor unici și durata medie a timpului petrecut pe fiecare site [8]

2. **Algoritmi și arhitecturi din aplicații asemănătoare**

În articolul [9] este prezentată problema tuturor aplicațiilor de taxi precum “Uber” sau aplicațiilor de hărți precum “Here we go”, “Google Maps” etc. în găsirea celui mai scurt drum pentru șoferi (problema cu care se confruntă și aplicația noastră). Diferența este faptul că față de aplicația noastră aceste aplicații pot avea inclusiv trasee care depășesc 1000 de noduri. Acest articol oferă o comparatie a algoritmilor bazată pe numărul de noduri pentru care un algoritm este mai eficient decât celălalt și propune o separare a unei arii în mai multe arii mai mici și apoi găsirea soluției ariei mari prin compunerea soluțiilor ariilor mai mici (o abordare de tip Divide et impera). Algoritmii comparați sunt Algoritmul lui Dijkstra, Algoritmul A* (un algoritm mai general decât cel al lui Dijkstra) și Algoritmul Floyd-Warshall.

```
// A* finds a path from start to goal.
// h is the heuristic function. h(n) estimates the cost to reach goal from node n.
function A_Star(start, goal, h)
    // The set of discovered nodes that may need to be (re-)expanded.
    // Initially, only the start node is known.
    openSet := {start}

    // For node n, cameFrom[n] is the node immediately preceding it on the cheapest path from start to n currently known.
    cameFrom := an empty map

    // For node n, gScore[n] is the cost of the cheapest path from start to n currently known.
    gScore := map with default value of Infinity
    gScore[start] := 0

    // For node n, fScore[n] := gScore[n] + h(n).
    fScore := map with default value of Infinity
    fScore[start] := h(start)

    while openSet is not empty
        current := the node in openSet having the lowest fScore[] value
        if current = goal
            return reconstruct_path(cameFrom, current)

        openSet.Remove(current)
        for each neighbor of current
            // d(current,neighbor) is the weight of the edge from current to neighbor
            // tentative_gScore is the distance from start to the neighbor through current
            tentative_gScore := gScore[current] + d(current, neighbor)
            if tentative_gScore < gScore[neighbor]
                // This path to neighbor is better than any previous one. Record it!
                cameFrom[neighbor] := current
                gScore[neighbor] := tentative_gScore
                fScore[neighbor] := gScore[neighbor] + h(neighbor)
                if neighbor not in openSet
                    openSet.add(neighbor)

    // Open set is empty but goal was never reached
    return failure
```

Figura 11 Algoritmul A * scris în pseudocod [10]

În plus , articolul propune o extensie a algoritmului Floyd-Warshall care este folositoare în cazul rețelelor statice cu arie largă.

$d_{min} := \infty$
for all exit nodes j of SG_{start} **do**
  $d_{exit} = dist(SG_{start}, i, j)$
  for all entry nodes l of SG_{dest} **do**
    $d_{main} := dist(Main, j, l)$; $d_{entry} := dist(SG_{dest}, l, k)$; $d_{total} := d_{exit} + d_{main} + d_{entry}$
    if $d_{total} < d_{min}$ **then** $d_{min} := d_{total}$; $j_{best} := j$; $l_{best} := l$

Figura 12 În loc de abordarea folosind Floyd Warshall este recomandată această abordare folosind subgrafuri (abordare de tip Divide et impera) [9]

În articolul [11] este prezentată exact problema cu care se confruntă aplicația noastră și este oferită o rezolvare care, conform acestui articol, este mai bună în anumite cazuri decât soluția unui algoritm “point-to-point”. Așadar , în acest articol este prezentată soluția cea mai rapidă de găsire a unui drum care are o sursă și o destinație fixate și un număr predefinit de puncte intermediare care sunt identificate prin poziția lor geografică. Spre deosebire de Dijkstra acest algoritm prezentat în articolul [11] permite să existe mai multe etichete pe același nod, fiecare dintre aceste etichete fiind adăugată la o anumită iterație (această iteratia va fi memorată).

Atunci când la final va fi aleasă o anumită iterație se va lua de pe fiecare nod eticheta coresponzatoare acelei iteratii și astfel se va realiza o reconstituire a celui mai scurt drum.

În articolul [12] este prezentată o îmbunătățire a algoritmului lui Dijkstra pe care îl vom folosi pentru a calcula cel mai scurt drum pe care utilizatorul trebuie să îl parcurgă pentru a cumpără ingredientele rămase . Conform articolului, algoritmul lui Dijkstra poate intre intr-un ciclu infinit în cazuri limită și algoritmul îmbunătățit al lui Dijkstra funcționează precum în exemplul următor:

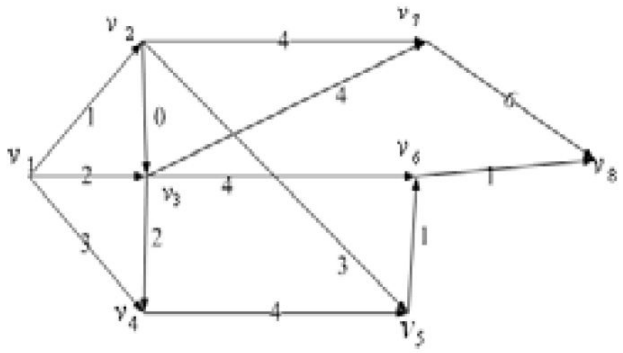


Figura 13. Exemplul de graf care urmează să fie rezolvat cu algoritmul prezentat în articolul [12]

R	V1	V2	V3	V4
1	0	1/V1	2/V1	3/V1
2		1/V1	1/V2	3/V1
3			1/V2	3/V1,V3
4				3/V1,V3

Figura 14. Calculul tabelului folosind algoritmul prezentat în articolul [12]

Observăm că cele mai scurte drumuri cu sursa v1 sunt (v1, v2, v3, v6, v8) cu costul 6, (v1, v2, v3, v7) cu costul 5 și (v1, v2, v3, v4) cu costul 3.

În articolul [13] este analizată din mai multe privințe o aplicație care se bazează pe arhitectura Model View Controller. La începutul articolului sunt prezentate dependințele dintre Model, View, Controller și User.Apoi sunt amintite avantajele

acestei arhitecturi: scalabilitatea este mai mare, facilitează mentenanța, design-ul este mai clar, coeziunea este mai puternică etc.

În plus , clasifică obiectele în obiecte entitate (entity objects), obiecte limită (boundary objects) și obiecte de control (control objects). Conform analizei prezentate în articol în cazul arhitecturii MVC sunt identificate următoarele reguli:

I. Actorii (userii) pot să interacționeze doar cu obiectele limită (boundary objects).

II. Obiectele limită (boundary objects) pot interacționa doar cu controller-ele și cu actorii.

III. Obiectele entitate (entity objects) interacționează doar cu controller-ele.

IV. Controller-ele pot interacționa doar cu obiectele limită (boundary objects), obiectele entitate (entity objects), cu alte controller-e, dar nu și cu actorii.

În articolul [14] este prezentat felul în care funcționează o aplicație bazată pe arhitectură Spring MVC. Așadar, Spring conține features care sunt organizate în 20 de module grupate în Core Container , Dată Access / Integration, Web, Aspect Oriented Programming, Instrumentation, Messaging și Text.

Cele mai mari avantaje oferite de Spring sunt separarea dependințelor și injectarea în aplicație la runtime. Aspected oriented programming oferă separarea modulelor de logică principal de business și lipirea lor în aplicație la compilare, la încărcarea claselor sau la runtime.

4 SOLUȚIA PROPUSĂ

Arhitectura aplicației

Diagrama de arhitectură

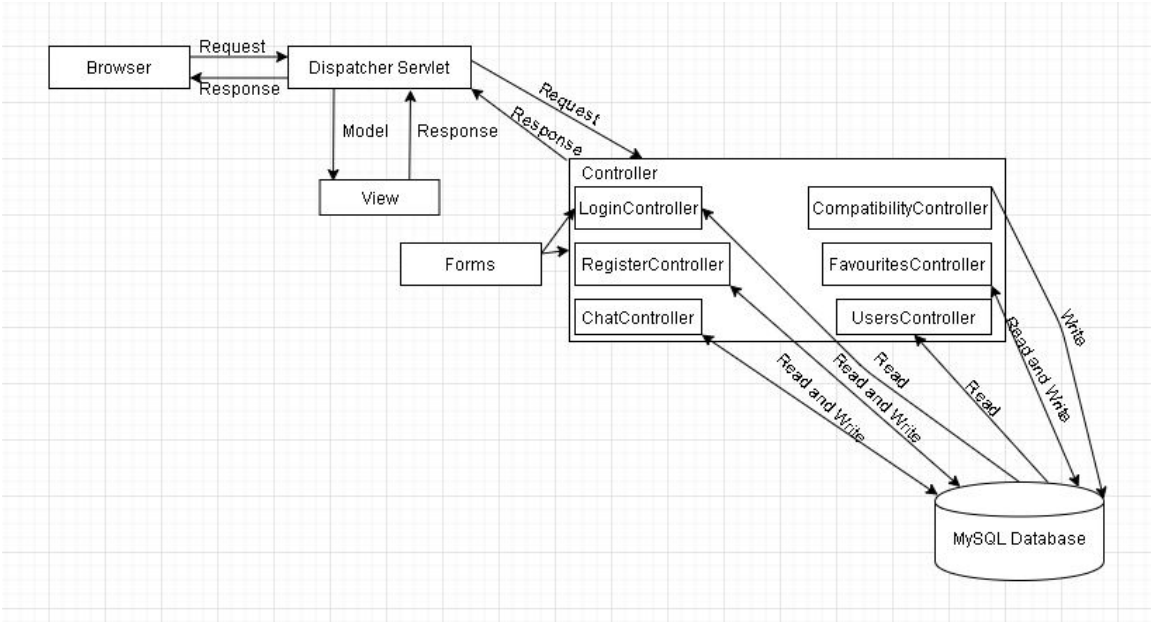


Figura 15 Diagrama de arhitectură

Arhitectura generală a aplicației este arhitectura Spring MVC. Conform articolului [15] care oferă o descriere a arhitecturii Model-View-Controller modelul este responsabil pentru extragerea și validarea datelor. De cele mai multe ori modelele sunt slabe (thin model application) și pot fi reutilizate în alte aplicații , cea mai mare parte a activității unui utilizator fiind păstrată în controllere.

View-ul este responsabil pentru interfața cu utilizatorul însemnând practice design-ul aplicației (butoane, etichete etc.). Faptul că în aplicație este separat View-ul de logică aplicației este important deoarece previne modificarea logicii aplicației atunci când se fac modificări doar în design-ul acesteia.

Controller-ul este responsabil pentru gestionarea evenimentelor (provocate de un utilizator care interacționează cu aplicația sau de un process de sistem), stabilește formatul răspunsului și preia informațiile de la Model.

Am ales să folosesc Java Spring pentru că, având Inversion of Control (în loc să instanțiăm un obiect al unei clase în altă clasă facem un constructor care are acel obiect ca parametru), nu există prea multă dependințe in cod, aplicația este modularizată, se foloseste abstractizarea (un utilizator este lăsat să facă doar anumite operații – spre exemplu, un utilizator va fi lăsat să trimită mesaje doar altor utilizatori cu care are compatibilitate culinară, nu tuturor utilizatorilor), aplicația este sigură din punct de

vedere al securității (putem elimina cât mai multe request-uri http din semnăturile metodelor, putem folosi SessionBean pentru a lua doar o dată informațiile de pe sesiune, va fi câte un controller pentru fiecare parte a aplicației) și se pot face teste unitare pentru a verifica fiecare parte a aplicației.

Fiecare pagină are un Controller, pentru Login și Register am definit 2 clase Java care conțin setterii și getterii pentru cele 2 Form-uri de pe aceste pagini și fiecare pagină are câte o pagină HTML asociata care se află în resources -> templates. Pentru design am folosit CSS și pentru legarea între Java și HTML am folosit Thymeleaf. În ceea ce privește legarea cu baza de date MySQL se folosește MySQL JDBC Driver.

În ceea ce privește relația dintre Controllere și baza de date, după cum am arătat în figura de mai sus LoginController și RegisterController citesc din baza de date pentru a verifica dacă utilizatorul este înregistrat, în caz contrar RegisterController scrie în baza de date introducând noul utilizator. ChatController citește din baza de date mesajele precedente ale utilizatorilor și, odată ce un utilizator scrie un mesaj, îl scrie pe acesta în baza de date. FavouritesController citește din baza de date rețetele adăugate de utilizator la secțiunea de favorite și scrie în baza de date atunci când un utilizator adaugă o rețetă la favorite. CompatibilityController scrie în baza de date odată ce au fost îndeplinite condițiile ca doi utilizatori sa aibă aceleași preferințe culinare.

Diagrama bazei de date

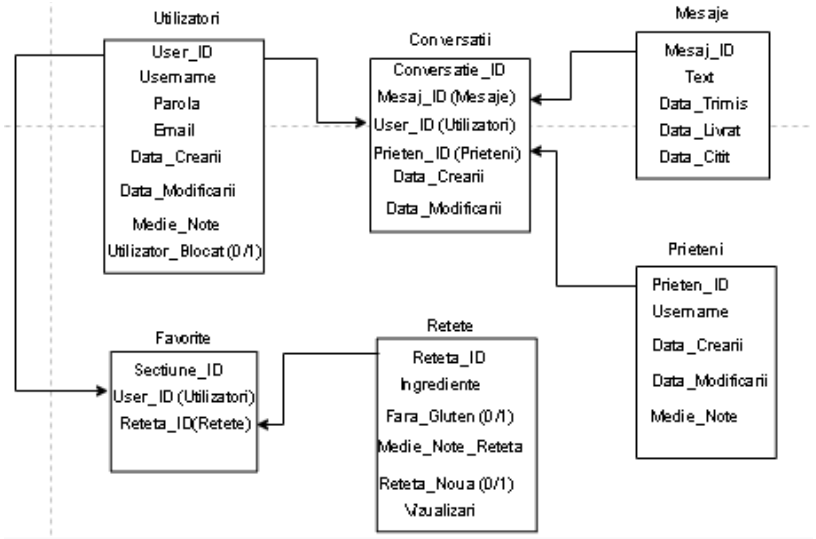


Figura 16. Diagrama bazei de date

Tabela Conversații are cheia primară Conversație_ID, iar Mesaj_ID, User_ID Prieten_ID sunt chei străine (cheile primare din tabelele Mesaje, Utilizatori și Prieteni). Tabela Favorite are cheia primară Secțiune_ID, iar User_ID și Rețetă_ID sunt chei străine (cheile primare din tabelele Utilizatori și Rețete). Relațiile dintre tabelele Conversații și Mesaje, Favorite și Rețete și Utilizatori și Prieteni sunt one-to-many. Restul relațiilor din baza de date sunt one-to-one.

Diagrama de clase

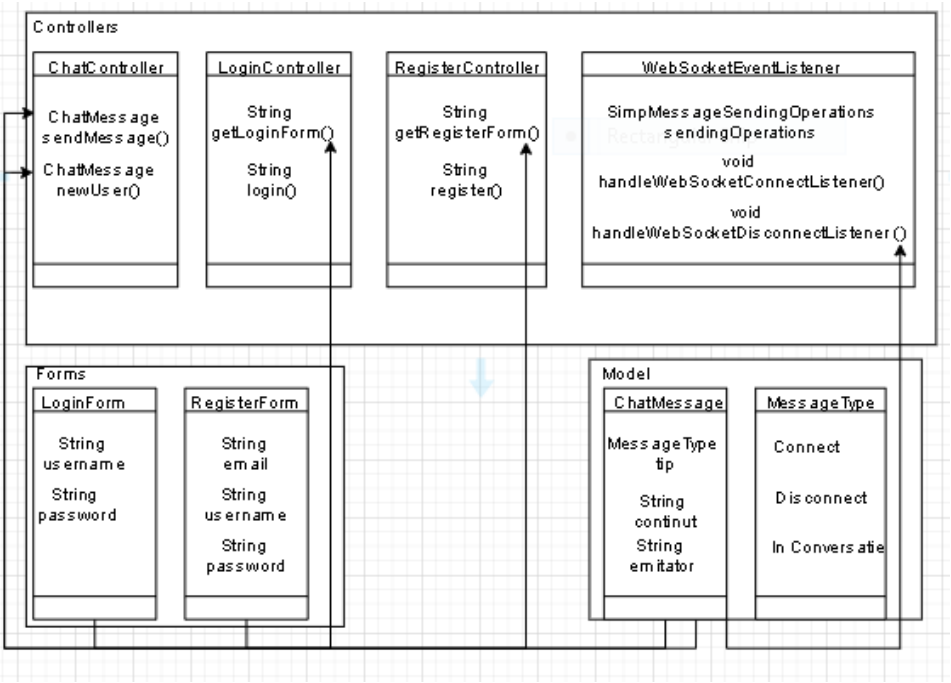


Figura 17. Diagrama de clase

În această diagramă am adăugat cele mai importante Controllere, împreună cu metodele fiecăreia și am evidențiat relațiile dintre metodele claselor Java din Controller și clasele Java din Forms și Model.

Diagrama de context

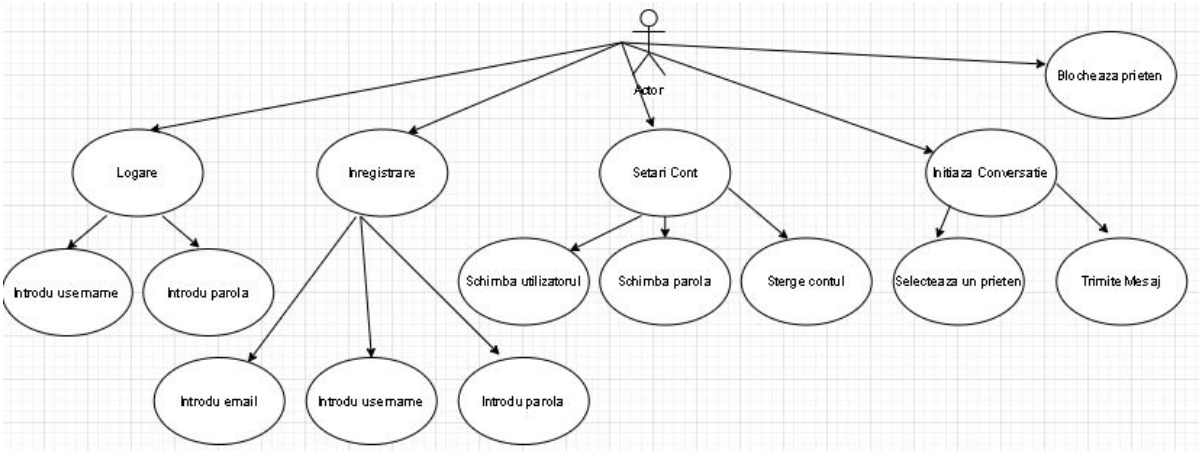


Figura 18. Diagrama de context (partea 1)

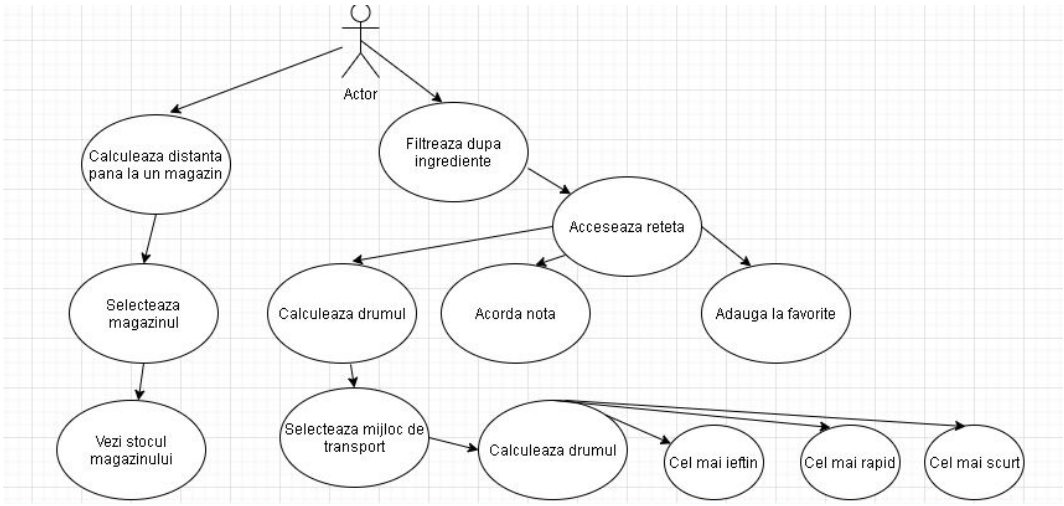


Figura 19. Diagrama de context (partea a 2-a)

Tehnologii utilizate

Felul în care trebuie dezvoltată o aplicație web folosind Spring MVC este explicat succint în articolul [16].

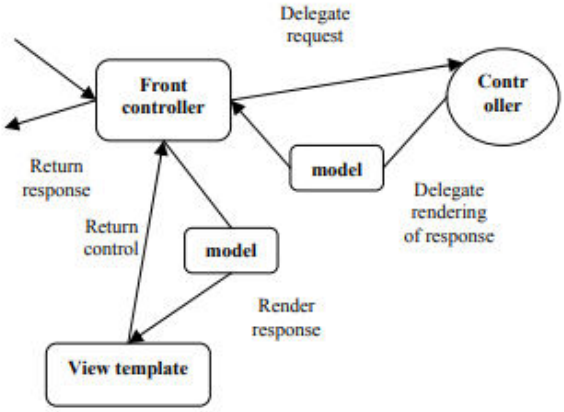


Figura 20 Arhitectura Spring [16]

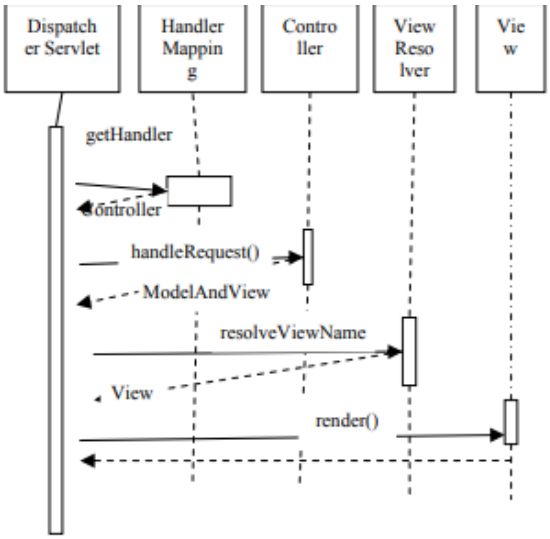


Figura 21 Diagrama de secvență a unei aplicatii dezvoltate folosind Spring [16]

Articolul [17] arată cum să dezvolți rapid o aplicație bazată pe arhitectură MVC. Articolul susține faptul că în cazul aplicațiilor web cel mai des sunt folosite CSS și HTML pentru a prezența informația utilizatorului (acestea sunt folosite și în aplicația noastră) și JavaScript (este, de asemenea, folosit în aplicația noastră).

Noua structură MVC propusă în acest articol pentru dezvoltarea mai rapidă a aplicațiilor bazate pe MVC este:

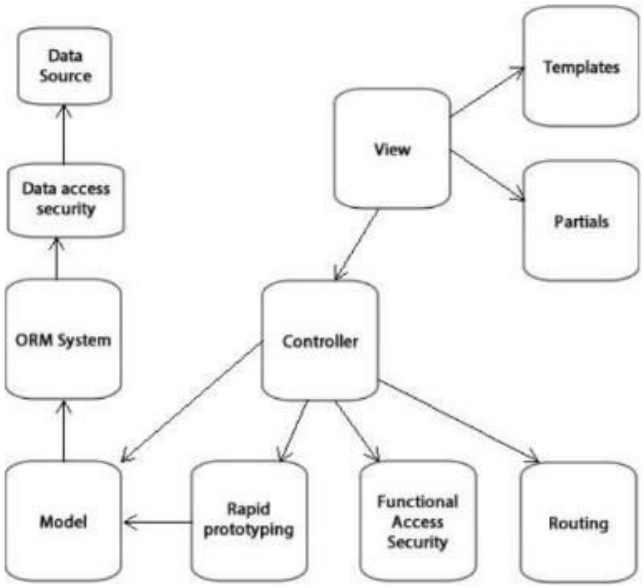


Figura 22 Structura MVC pentru dezvoltarea rapidă a unei aplicații [17]

O alternativă existentă pentru dezvoltarea aplicației este framework-ul Ionic cu AngularJS, dar nu este recomandat folosirea acestuia pentru că, având mult cod, nu va fi la fel de bine modularizat precum folosind Spring si va fi din ce în ce mai greu de adăugat feature-uri pe măsură ce aplicația va evolua.

Pentru partea de backend alternativele sunt Python sau Ruby. Pentru Python dezavantajele sunt faptul că este mai lent decât Java (pentru că este un limbaj compilat, cu toate că Java consumă mai multă memorie decât Python), faptul că are limitări când vine vorba de accesul la baza de date si faptul că erorile de runtime sunt mai des întâlnite în Python decât în Java.

Dezavantajele lui Ruby sunt faptul că nu este flexilbil, faptul că are un timp destul de scăzut la runtime si faptul că o greseală poate fi depistată mai greu decât in Java (debug-ul este mai anevoios).

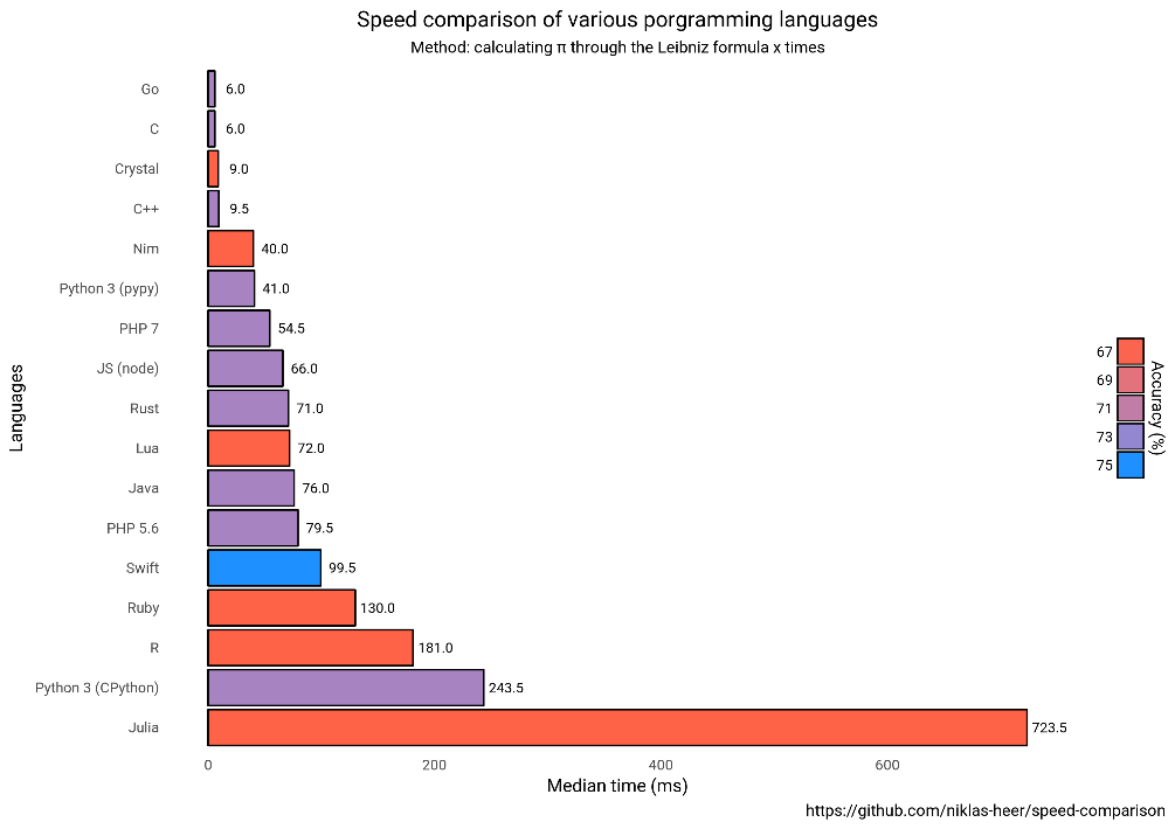


Figura 23 Comparația timpilor pentru cele mai cunoscute limbaje de programare [18]

Algoritmi folosiți în aplicație

Pentru a determina cel mai scurt drum (în acest caz avem doar grafuri complete, adică avem muchie între oricare 2 noduri ale grafului) si cel mai ieftin drum (în cazul celui mai ieftin drum vom seta un maximum pentru distanța până la fiecare magazin, astfel că după ce vom calcula cele mai mici prețuri pentru fiecare ingredient ne va rămâne un graf ponderat căruia trebuie să îi aflăm arborele minim de acoperire) pe care utilizatorul trebuie să le parcurgă pentru a cumpăra produsele rămase de la magazinele din apropiere vom folosi:

I. Algoritmi de calcul ai celui mai scurt drum (notăm cu V numărul de vârfuri ale grafului si cu E numărul de muchii ale grafului). Vom folosi algoritmii pentru o singură sursă (aceasta fiind utilizatorul).

a) Algoritmul lui Dijkstra [19]

Are complexitatea temporală egală cu O(|V|lg|V| + |E|) (pentru cazul în care folosim un heap Fibonacci - se foloseste mai ales în grafurile rare – un

graf este rar dacă numărul de muchii este mult mai mic decât numărul de vârfuri ridicat la pătrat).

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:           // only v that are still in Q
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

Figura 24 Algoritmul lui Dijkstra scris în pseudocod [19]

b) Algoritmul Bellman Ford

Are complexitatea temporală egală cu O(|E|*|V|). Este similar cu Dijkstra, singura diferență este că Bellman-Ford funcționează pentru muchii de cost negativ, iar Dijkstra nu face acest lucru, dar nu este cazul nostru, noi vom folosi Dijkstra.

II. Algoritmi pentru calculul arborelui minim de acoperire al grafului ponderat

a) Algoritmul lui Kruskal

Are complexitatea temporală egală cu O(E * log V).

b) Algoritmul lui Prim

Are complexitatea temporală egală O(V ^ 2).

În cazul în care mai rămâne un singur produs într-un magazin și un alt om cumpără acel produs în timp ce utilizatorul se indreaptă către magazin, utilizatorul va fi atenționat printr-o notificare (stocul magazinului va fi actualizat periodic) și se va reconfigura traseul utilizatorului.

In cazul alegerii algoritmului pentru arborele minim de acoperire algoritmul lui Kruskal și algoritmul lui Prim au același rezultat, dar principala diferență constă în nodul care

urmează să fie adăugat în arborele minim de acoperire la fiecare pas: Prim reține mereu o component conexă (o component conexă are proprietatea că poți ajunge dintr-un nod în oricare alt nod al ei) și caută următorul nod pe care să îl adauge în acea componentă conexă, iar Kruskal nu reține o component conexă, ci o pădure (o mulțime de arbori) și adaugă o muchie la ea doar dacă acea muchie conectează 2 arbori diferiți.

Noi vom alege să folosim Prim pentru că luăm în considerare cazul limită explicat mai sus (dacă se reconfigurează traseul pentru că un produs a fost epuizat chiar înainte să ajungă utilizatorul în magazin ne va fi de folos component conexă curentă, pentru că putem adăuga noduri la ea; dacă am folosi Kruskal va trebui sa reluăm toți pașii, iar acest lucru va consuma mult timp).

```
procedure PRIM(G:graf; var T:MultimeArce);
{Date de intrare: graful ponderat G=(N,A)}
{Date de ieșire: arborele de acoperire minim T al grafului G, T:MultimeArce}
begin
 *inițializează mulțimea arcelor arborelui T ca fiind mulțimea vidă
 *inițializează mulțimea nodurilor arborelui, U, prin adăugarea unui nod arbitrar de pornire
 while (*nu s-au adăugat mulțimii U toate nodurile mulțimii N)
 begin
  *găsește arcul (u,v) cu costul cel mai redus care conectează pe U cu N\U și care are capetele u în U și pe v în N\U
  *adaugă arcul (u,v) la arborele de acoperire T
  *adaugă nodul v la mulțimea U a nodurilor arborelui T
 end;
end;
```

Figura 25 Pseudocodul algoritmului lui Prim [20]

Pentru sortarile din aplicația noastră vom folosi java.util.Collections.sort, care utilizează Mergesort. În articolul [21] este analizată performanța algoritmului Mergesort comparativ cu cea a algoritmului Quicksort, care este considerat unul dintre cei mai rapizi algoritmi de sortare.

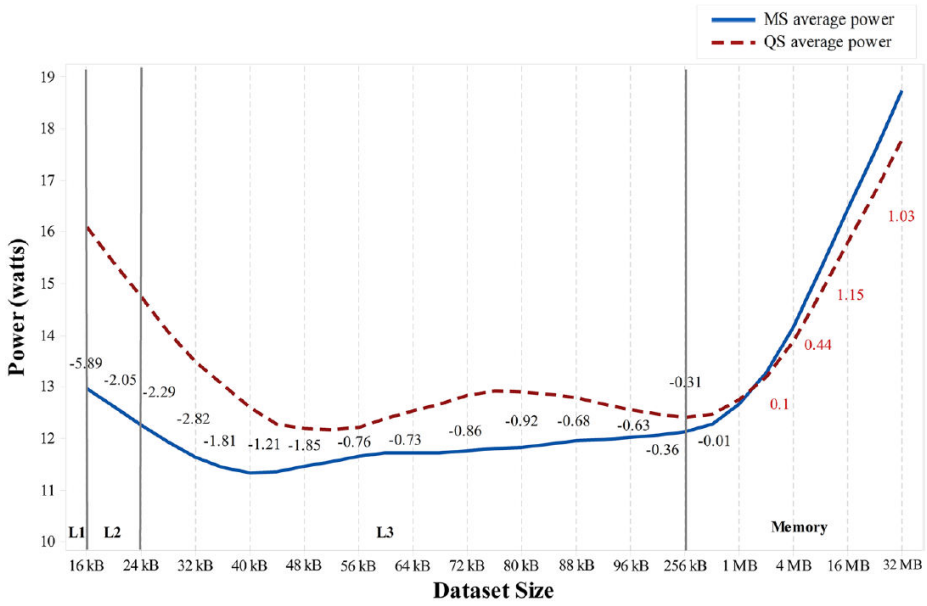


Figura 26 Puterea medie consumată pentru Mergesort și Quicksort [21]

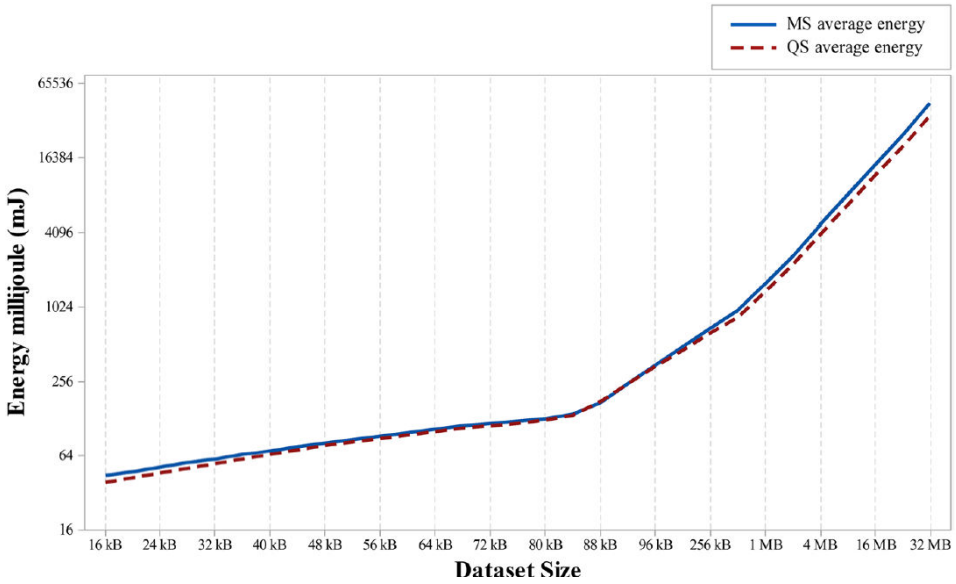


Figura 27 Energia medie consumată pentru Mergesort și Quicksort [21]

5 DETALII DE IMPLEMENTARE

Funcționalitățile principale ale aplicației

Logarea și înregistrarea

Pagina de login este prezentată în figura de mai jos. După introducerea credentialelor aplicația caută în baza de date daca există utilizatorul respectiv și dacă acelui utilizator îi corespunde parola respectivă. În cazul în care perechea (utilizator, parolă) nu

există în baza de date va returna un mesaj de eroare și nu îi va permite logarea utilizatorului precum este arătat în figura de mai jos:

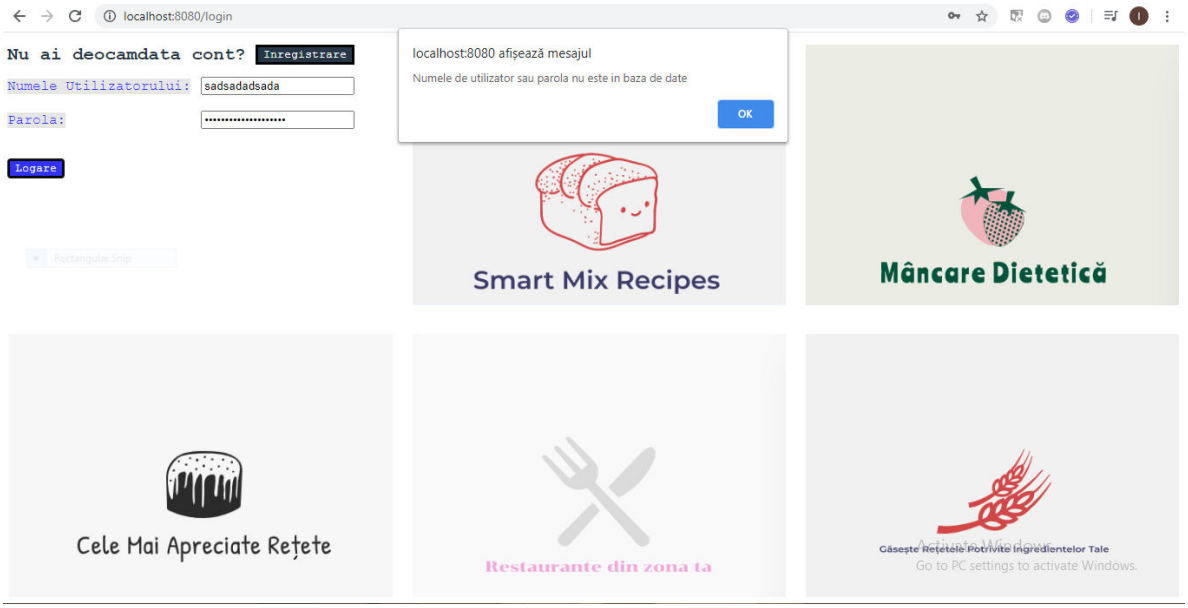


Figura 28. Pagina de login și afișarea mesajului pentru credențiale greșite

Dacă utilizatorul dorește să își creeze un cont va apăsa butonul “Înregistrare” de pe pagina de login și va fi redirecționat spre pagina de înregistrare. În cazul în care el va încerca să înregistreze un utilizator deja existent aplicația va căuta în baza de date, dar nu îi va permite înregistrarea și îl va atenționa cu un mesaj:

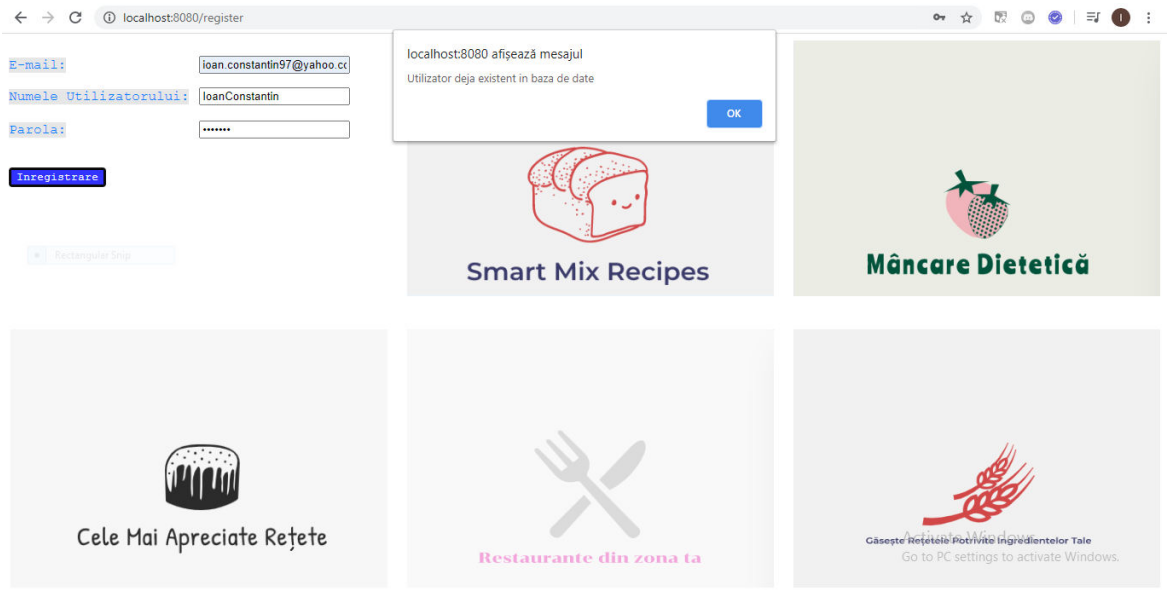


Figura 29. Pagina de register și afișarea mesajului pentru utilizator deja existent

În cazul în care utilizatorul se înregistrează cu un nume de utilizator care nu există deja în baza de date este adăugat în baza de date rândul utilizatorului respectiv și acesta este logat și îi este arătată pagina cu cele mai apreciate rețete:

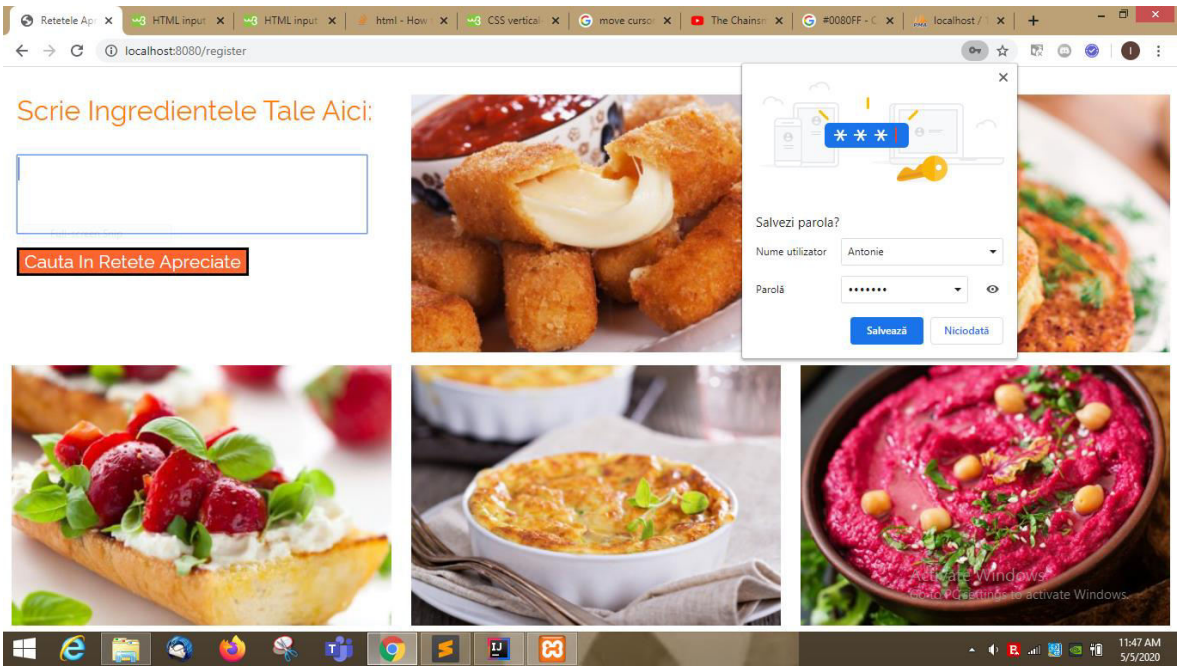


Figura 30. Utilizator înregistrat cu succes

Pentru verificarea credențialelor de login am făcut un select pe toată tabela users definită de mine în MySQL și apoi am parcurs cu un loop înregistrările din tabelă. La clasa Register Controller se execută un query care modifică baza de date. Întâi trebuie făcut un select care ia toate înregistrările din tabela cu utilizatori, iar apoi se face un insert doar daca atunci când am parcurs cu un loop toate înregistrările nu găsim un nume de utilizator echivalent cu numele introdus în browser (daca se întâmplă acest lucru afișăm mesajul “Utilizator Deja Existent”. În Register HTML am aranjat pozele și am editat font-urile pentru form și am folosit th:if din Thymeleaf pentru a afișa mesajul “Utilizator Existent” când este cazul.

Optiunea “Rețete Apreciate”

Pe pagina “Rețete Apreciate” inițial sunt afișate pozele cu fiecare rețetă și un formular pentru căutare după ingrediente. În cazul în care facem hover cu mouse-ul pe o poză cu o rețetă se va afișa textul acelei rețete. Utilizatorul va introduce un șir de caractere

cu ingredientele după care dorește să efectueze filtrarea și îi sunt afișate rețetele care au trecut de filtrare și numărul lor.

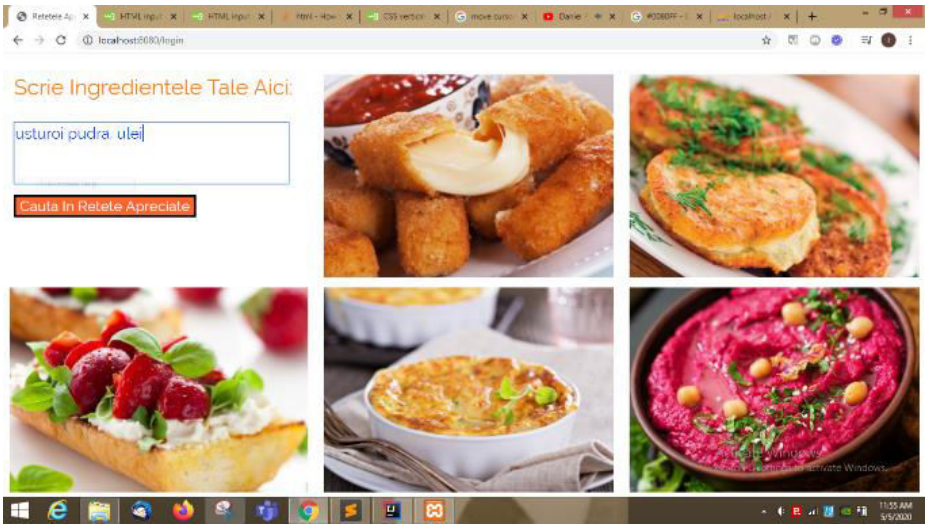


Figura 31. “Rețete Apreciate” inițial

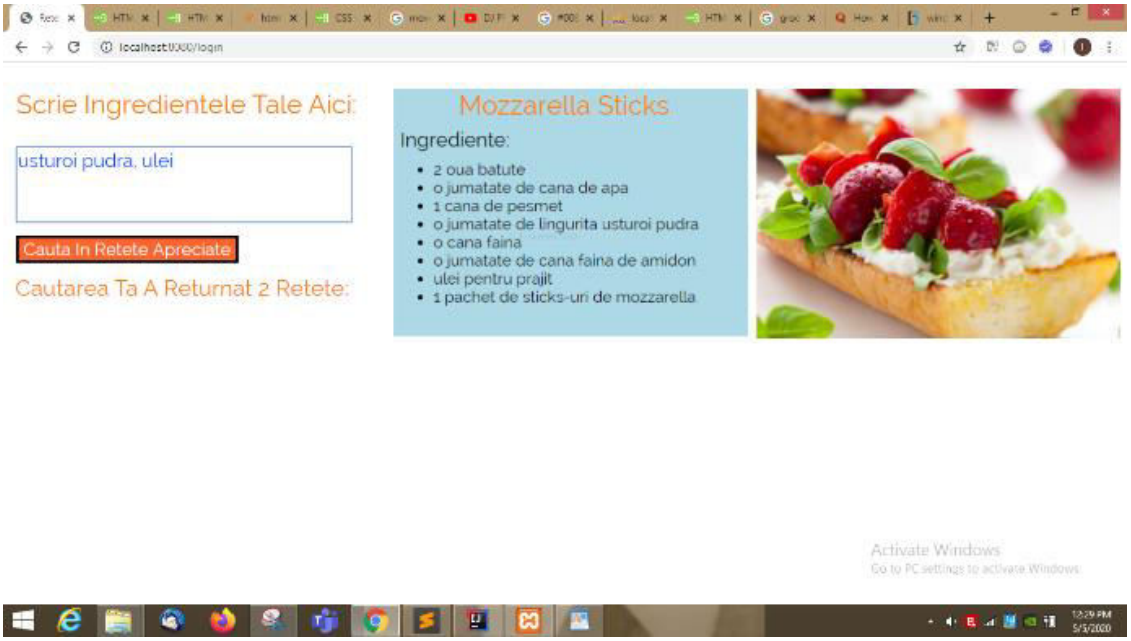


Figura 32. “Rețete Apreciate după filtrare ingrediente și mouse hover”

Integrarea Google Maps API

Întâi am creat o cheie pentru API pe google cloud, apoi am scris un script în care am centrat harta în punctul cu locația mea și apoi am adăugat niste markere pentru locațiile câtorva magazine din București.

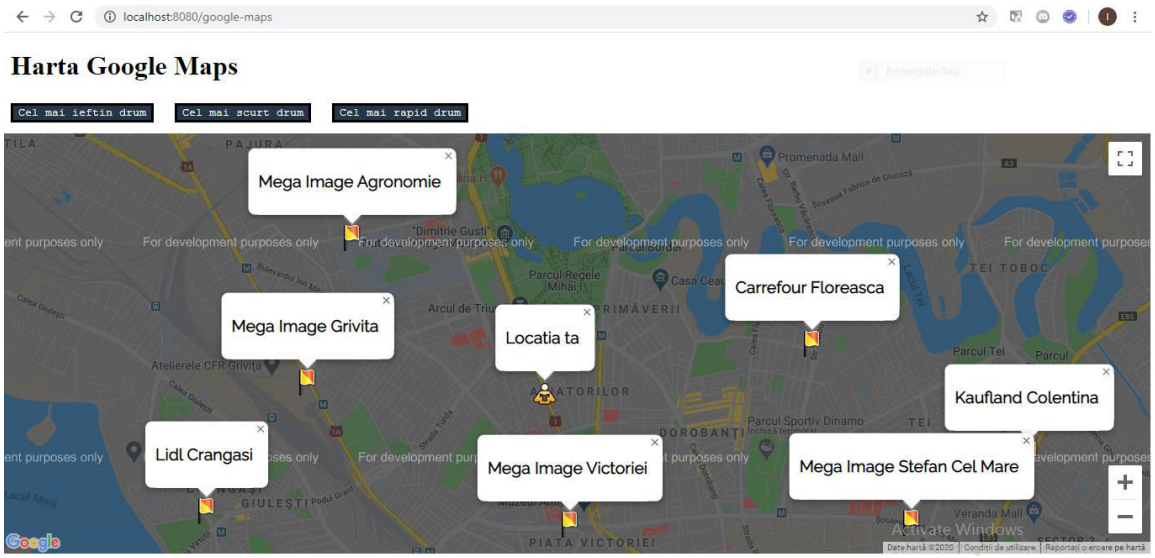


Figura 33. Magazinele din apropierea utilizatorului

Distanța dintre un utilizator și un magazin

Pentru a reuși să îi furnizăm utilizatorului cel mai scurt drum (în minute și în kilometri) și cel mai ieftin drum pentru a cumpăra ingredientele de care are nevoie pentru o anumită rețetă trebuie inițial să calculăm distanța dintre locația sa și fiecare magazin.

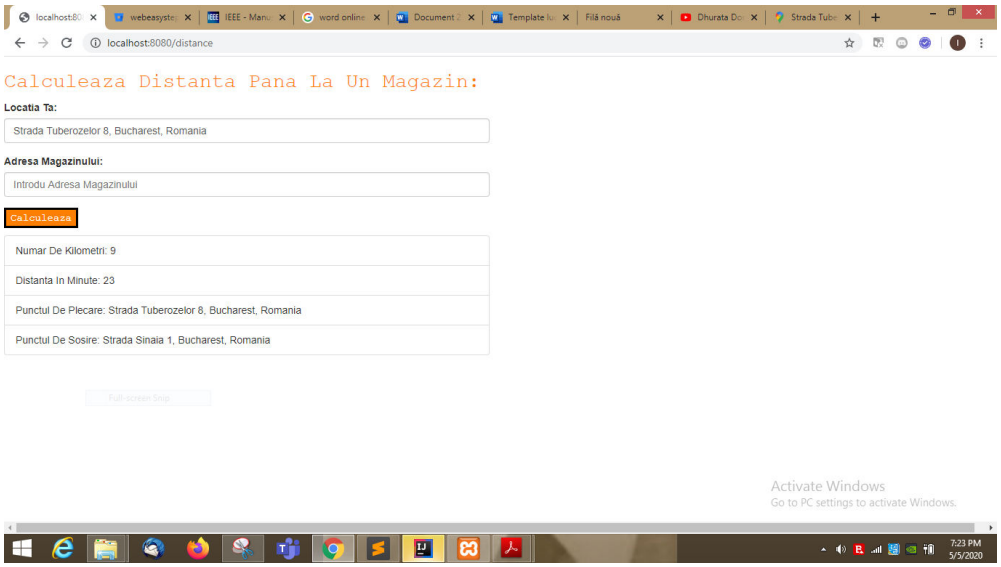


Figura 34. Calculul distanței până la un magazin

Localizarea utilizatorilor si a magazinelor

Pentru a determina cu exactitate pozițiiile utilizatorilor si pe cele ale magazinelor, cofetăriilor si patiseriilor vom integra Google Maps API. Am găsit o

carte în care este explicat cum trebuie integrat acest API si metodele si constructorii sunt explicate si explicati atât ca functionalitate, cât și ca utilizare [22].

De exemplu va fi de ajutor constructorul LatLng (lat : number, lng : number, noWrap ? boolean). Dacă noWrap este false atunci se va verifica dacă latitudinea introdusă este între -90 și 90 de grade și dacă longitudinea introdusă este între -180 și 180 de grade. Dacă nu sunt în aceste intervale vor fi automat modificate pentru a aparține intervalelor . Dacă noWrap este true atunci nu se mai verifică condiția de mai sus și se iau latitudinea și longitudinea așa cum au fost introduse.

Chat între utilizatori

Mai jos este prezentat un exemplu de chat privat între 2 utilizatori:

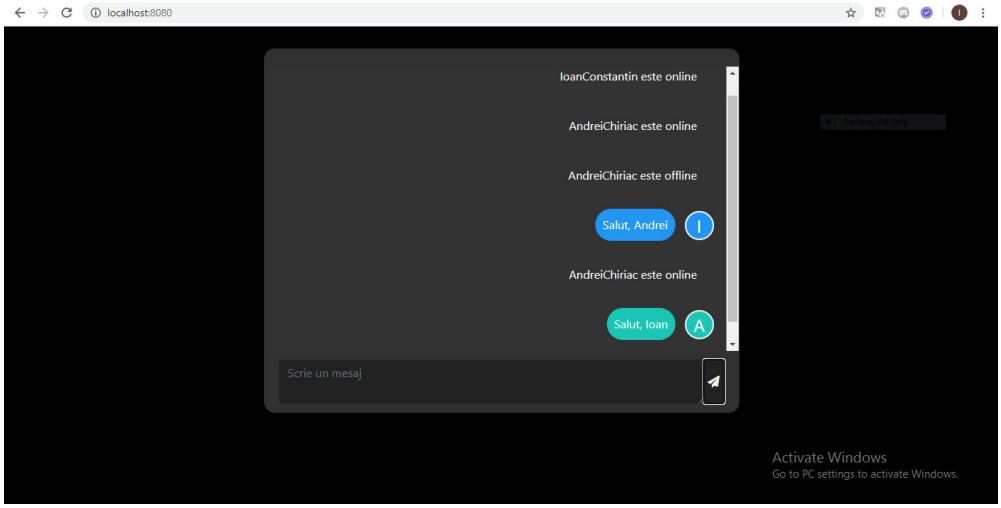


Figura 35. Chat-ul privit din perspectiva utilizatorului IoanConstantin

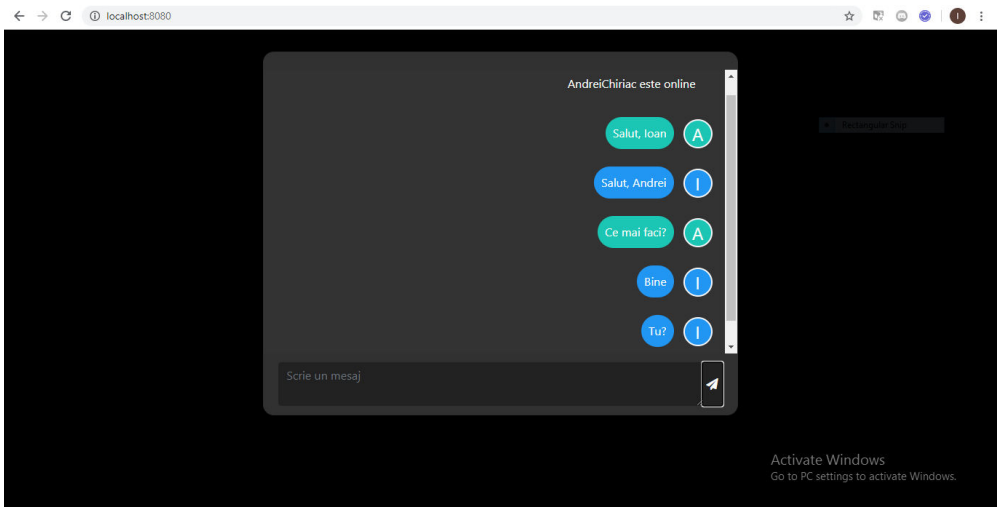


Figura 36. Chat-ul privit din perspectiva utilizatorului AndreiChiriac

Pentru realizarea acestui chat am urmat tutorialul de la [23]. Am adăugat plugin-ul Lombok în pom.xml în Maven. Apoi în clasa WebSocketMessageConfig se adaugă endpoint-urile și am folosit un message broker pentru a intermedia comunicarea dintre cei 2 utilizatori. Un mesaj are atributele conținut, emițător și tip (tipul poate fi conectat, deconectat sau în conversație). Se utilizează anotațiile @Getter și @Setter specifice Lombok in locul setter-ilor și getter-ilor clasici.

Scenarii de utilizare

Utilizatorul IoanConstantin se află pe pagina home a aplicației SmartMix în momentul în care este informat de faptul că un alt utilizator are aceleasi preferințe culinare:

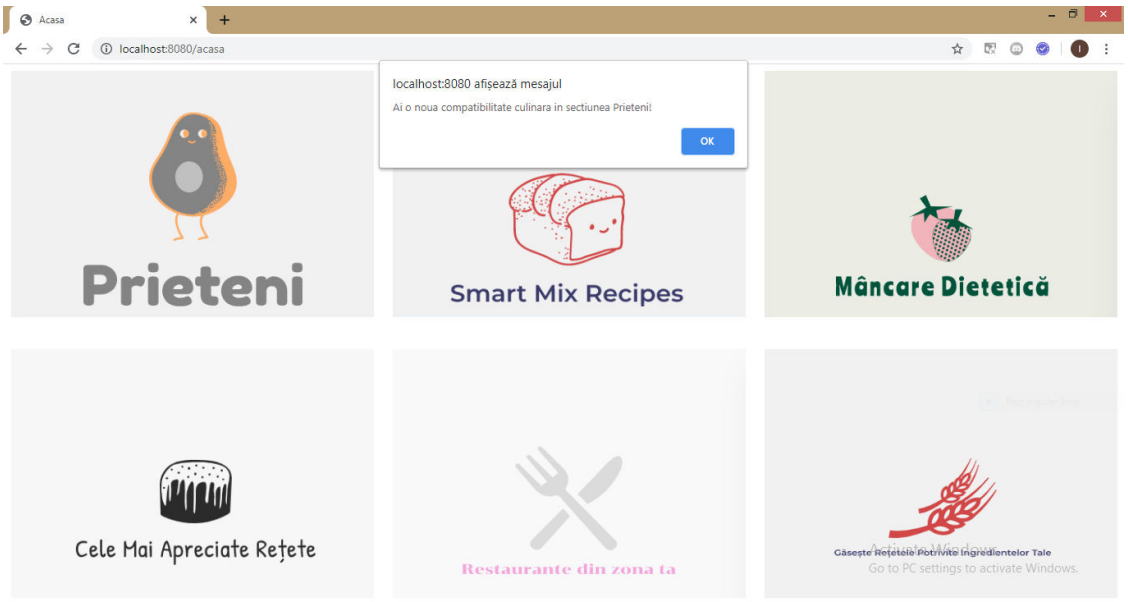


Figura 37. Utilizatorul IoanConstantin este informat că există un alt utilizator cu aceleași preferințe culinare

IoanConstantin intră în conversație cu utilizatorul respectiv și află că acel utilizator citește rețeta “Mozzarella Sticks”:

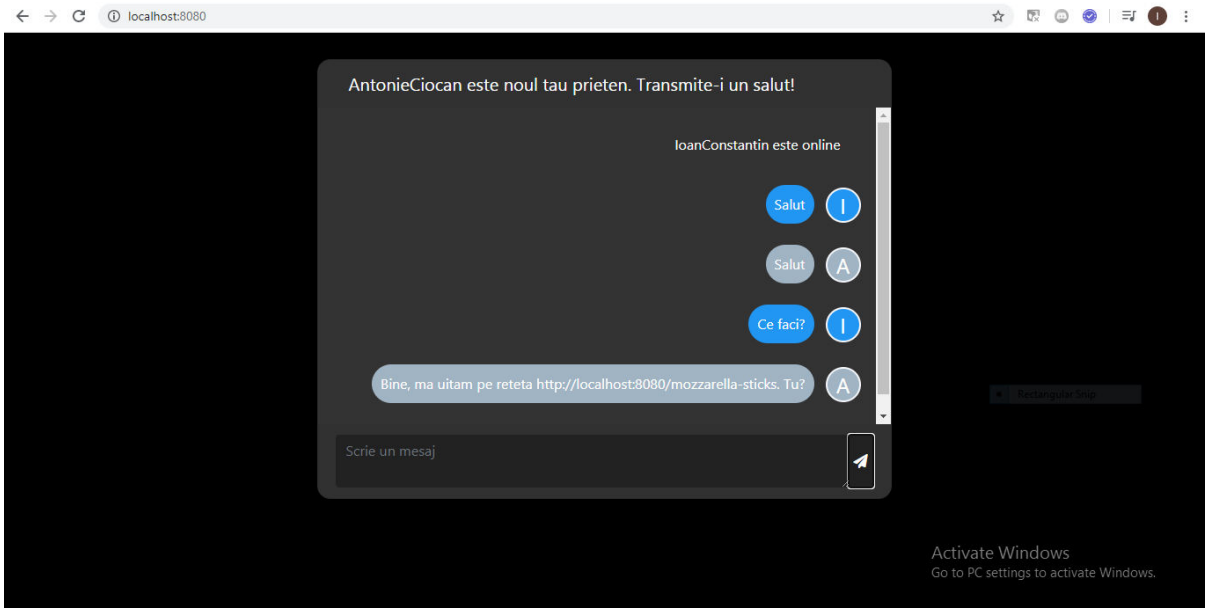
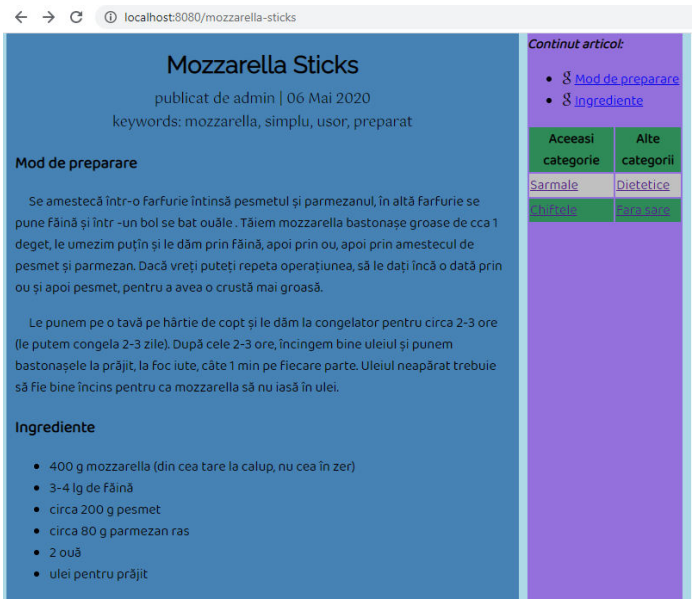
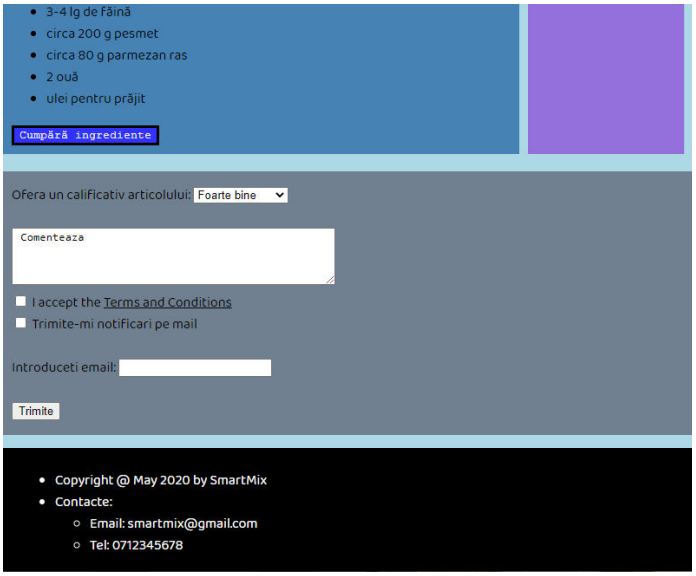


Figura 38. Conversația dintre IoanConstantin și AntonieCiocan

IoanConstantin accesează linkul trimis de AntonieCiocan și îi este afișată pagina respectivă:





Figurile 39 și 40. Pagina rețetei “Mozzarella Sticks”

Apoi utilizatorul IoanConstantin apasă pe butonul “Cumpără ingrediente” de pe pagina de mai sus și selectează opțiunea “Cel mai ieftin drum” de pe pagina de mai jos. Îi sunt apoi afișate magazinele la care trebuie sa meargă în ordine, ingredientele pe care trebuie să le cumpere de la fiecare magazin și prețurile acestor ingrediente:

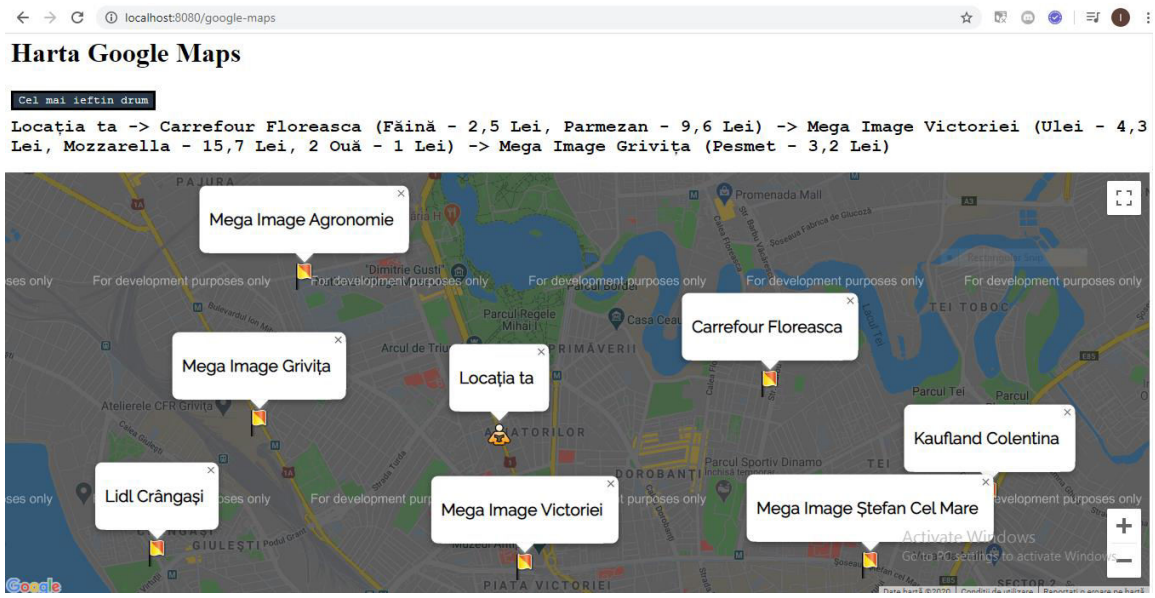


Figura 41. Cel mai ieftin drum exemplificat (ordinea magazinelor, ingrediente, prețuri)

6 EVALUARE

După o medie de 10 rulări ale aplicației aceasta pornește în 7,463 secunde. Din punct de vedere al scalabilității adăugarea de module se poate realiza cu ușurință datorită separării

dependințelor și injectării lor în aplicație la runtime. Astfel, logica curentă a aplicației nu este impactată la adăugarea ulterioară a acestor module.

Am rulat 4 versiuni intermediare salvate ale aplicației: prima versiune nu are nici legarea cu baza de date, nici chat-ul dintre utilizatori, nici calculul drumurilor; a doua versiune are doar logarea, înregistrarea și conexiunea cu baza de date; a treia variantă are în plus față de a doua chat-ul dintre utilizatori și secțiunea favorite, iar a patra variantă este aplicația întreagă incluzând calculul drumurilor. Timpii de rulare ai celor 4 variante au fost 2.18, 3.92, 4.59, 7.46 secunde. Graficul cu timpii medii obținuți după câte 10 rulări ale fiecare variante este ilustrat în figura următoare:

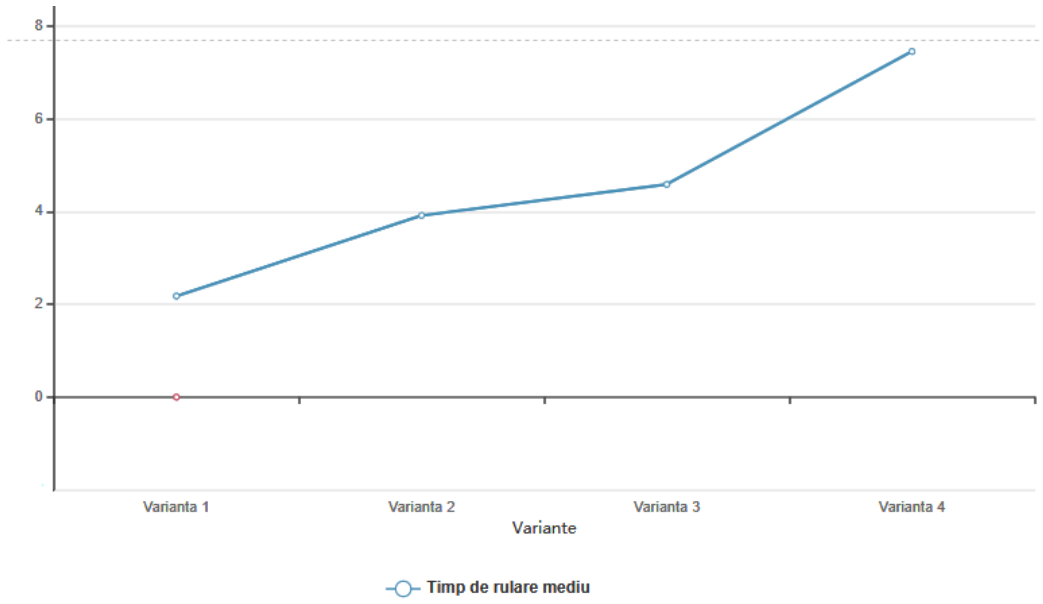


Figura 42. Timpii de rulare medii ai celor 4 variante exemplificate mai sus

În ceea ce privește uzabilitatea aplicației am arătat aplicația unor persoane care nu sunt familiarizate cu acest tip de aplicații și aceștia mi-au dat un feedback legat de părțile mai neintuitive ale aplicației pe care le-am modificat ulterior. Totuși, numărul de persoane fiind redus, nu pot formula o statistică, însă problemele principale semnalate țineau de așezarea neintuitivă în pagină și de anumite denumiri ambigue.

Conform [24] există mai multe variante prin care se pot îmbunătăți performanțele aplicațiilor. Una dintre cele mai bune variante este folosirea unui pool de conexiuni. Aplicația folosește astfel doar conexiunea de care are nevoie la un anumit moment după care trimite din nou această conexiune în pool.

Alte metode specificate în articolul [24] sunt: în ceea ce privește legarea cu baza de date trebuie făcute doar join-uri absolut necesare; în ceea ce privește Spring Security trebuie șters cât mai des cache-ul de autentificare; de asemenea, trebuie șterse fișierele temporare, iar în plus față de thread-urile server-ului Tomcat poate fi folosit Executor Framework care ajută prin executarea unor call-uri asincrone.

Având în vedere că unul dintre indicia de performanță menționați în articolul de mai sus este cât de repede comunică aplicația cu baza de date, am măsurat timpii pentru 4 variante. Un utilizator comunică pe chat cu unul, respectiv doi utilizatori și fiecare conversație conține zece, respectiv douăzeci de mesaje. Se măsoară în cât timp se umplu tabelele din baza de date specifice acelor conversații (adică în cât timp se adaugă în aceste tabele numărul de rânduri corespunzător numărului de mesaje trimis de utilizatori). Timpii obținuți sunt 0.02 ms pentru 1 conversație și 10 mesaje (varianta 1), 0.05 ms pentru 1 conversație și 20 de mesaje (varianta 2), 0.09 ms pentru două conversații și câte 10 mesaje (varianta 3) și 0.21 ms pentru două conversații și câte 20 de mesaje (varianta 4). Graficul este ilustrat în figura următoare:

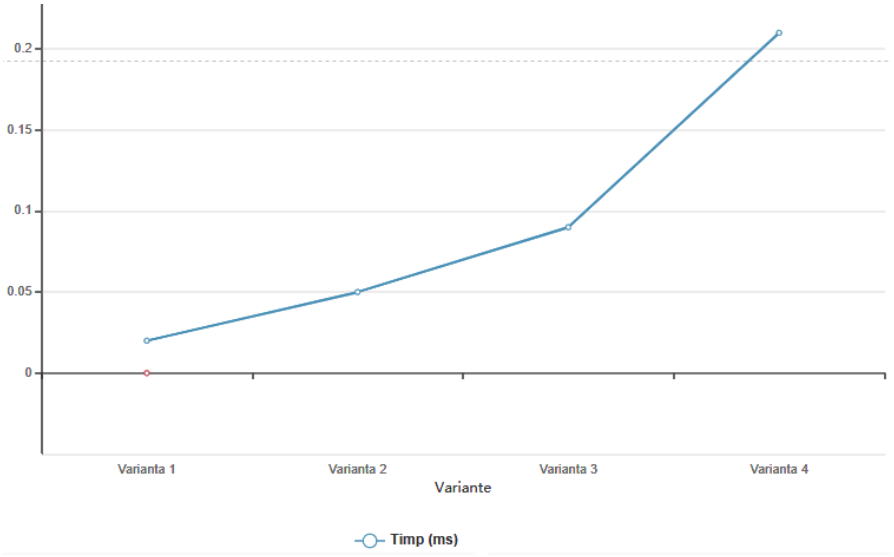


Figura 43. Timpii în care se updatează baza de date pentru cele 4 variante de mai sus

7 BIBLIOGRAFIE

[1] Sondaj cu întrebarea "Câți bani se economisesc cu gătitul în detrimentul mersului la restaurant sau comandării de mâncare?" realizat în SUA în 2018, https://www.forbes.com/sites/priceonomics/2018/07/10/heres-how-much-money-do-you-save-by-cooking-at-home/#4506dd735e54, accesat pe 30.05.2020

[2] Sondaj cu mai multe întrebări despre gătit realizat în SUA în 2016, https://www.reportlinker.com/insight/americans-cooking-habits.html, accesat pe 14.06.2020

[3] Sondaj cu întrebarea "Câte minute gătiți în fiecare zi?"realizat în SUA în 2015, https://www.bls.gov/tus/charts/household.htm, accesat pe 02.06.2020

[4] www.gustos.ro, accesat pe 07.05.2020

[5] www.petitchef.ro, accesat pe 07.05.2020

[6] www.allrecipes.com, accesat pe 23.05.2020

[7] www.food.com, accesat pe 28.05.2020

[8]Numărul de vizitatori americani ai website-urilor de gătit, https://www.nielsen.com/us/en/insights/article/2014/recipe-for-success-86-million-americans-visited-food-and-cooking-websites/, accesat pe 08.01.2020

[9] Gutenschwager, Kai & Radtke, Axel & Volker, Sven & Zeller, Georg. (2012). The shortest path: Comparison of different approaches and implementations for the automatic routing of vehicles. Proceedings - Winter Simulation Conference. 1-12. 10.1109/WSC.2012.6465023.

[10] Algoritmul A*, https://en.wikipedia.org/wiki/A*_search_algorithm, accesat pe 15.04.2020

[11] S. Demeyer, P. Audenaert, and M. Pickavet, “On Determining the Shortest Path through a Number of Intermediate Points”, Ghent University, Department of Information Technology (INTEC), 2012

[12] Ji-Xian Xiao and Fang-Ling Lu, "An improvement of the shortest path algorithm based on Dijkstra algorithm," *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, Singapore, 2010, pp. 383-385.

[13] Selfa, Diana & Carrillo, Maya & Boone, Ma. (2006). A Database and Web Application Based on MVC Architecture. 48 - 48.

[14] Naik, Poornima & Naik, Girish. (2017). Struts, Hibernate and Spring Integration – A Case Study. International Journal on Recent and Innovation Trends in Computing and Communication. 5. 261-268.

[15] Pop, Dragos-Paul & Altar Samuel, Adam. (2014). Designing an MVC Model for Rapid Web Application Development. Procedia Engineering. 69. 10.1016/j.proeng.2014.03.106.

[16] Gupta, Praveen & Praveen, & Govil, M.. (2010). Spring Web MVC Framework for rapid open source J2EE application development: a case. International Journal of Engineering Science and Technology. 2.

[17] Pop, Dragos-Paul & Altar Samuel, Adam. (2014). Designing an MVC Model for Rapid Web Application Development. Procedia Engineering. 69. 10.1016/j.proeng.2014.03.106.

[18] Comparația timpilor pentru cele mai cunoscute limbaje de programare, https://github.com/niklas-heer/speed-comparison, accesat pe 23.01.2020

[19] Algoritmul lui Dijkstra, https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm, accesat pe 01.04.2020

[20]Algoritmul lui Prim, https://forum.softpedia.com/topic/514186-algoritmul-lui-prim, accesat pe 02.04.2020

[21] Aljabri, Naif & Al-Hashimi, Muhammad & Saleh, Mostafa & Abulnaja, Osama. (2019). Investigating power efficiency of mergesort. The Journal of Supercomputing. 10.1007/s11227-019-02850-5.

[22] Svennerberg, Gabriel. 2010. *Beginning Google Maps API 3*.

[23] Java Websockets Chat, https://codeyogi.co.uk/2020/01/11/java-websockets-chat-app, accesat pe 16.05.2020

[24] Îmbunătățiri de performanță pentru aplicațiile Spring, http://www.jcombat.com/spring/performance-tuning-of-spring-based-application, accesat pe 22.06.2020