UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Restaurant booking platform
RAPORT DE PROIECTARE ARHITECTURALĂ

Studenți:
Ciaușu Nicoleta 406
Dabu Alexandru 406
Dima Oana 406
Tudose Andrei 406

2023

Cuprins

1. Scopul aplicației

Descrierea tipului, misiunii și utilizatorilor aplicației (max. ½ de pagină)

"Restaurant booking platform" este o aplicație creată pentru efectuarea rezervărilor la restaurante. Aceasta conectează de la distanță utilizatorii cu personalul restaurantelor, simplificând astfel procesul de rezervare al meselor, prin eliminarea necesității de a apela sau vizita locația dorită. De asemenea, proprietarii restaurantelor au posibilitatea de a oferi diferite facilități utilizatorilor care și-au demonstrat loialitatea față de locațiile lor, cum ar fi oferte speciale, discounturi sau diferite programe de loialitate.

Aplicația se încadrează în categoria "booking apps" fiind o aplicație web, care poate fi accesată atât de pe desktop, cât și de pe smartphone sau tabletă, indiferent de sistemul de operare sau de tipul de browser utilizat.

Misiunea este să ușureze procesul rezervării unei mese la restaurantele preferate. Întregul proces devine mai accesibil pentru toate părțile implicate, deoarece sunt eliminate apelurile telefonice sau vizitele în persoană, rezervarea putând fi efectuată în orice moment al zilei, fără a fi implicate direct, în același timp, toate părțile participante.

Grupul țintă este format din persoanele care preferă să se bucure de o masă în oraș la restaurantul preferat și doresc să rezerve cât mai ușor o masă în prealabil. Aplicația este deosebit de populară în rândul persoanelor ocupate care nu au timp să sune sau să viziteze un restaurant pentru a face o rezervare. Aceasta poate fi utilizată și de către turiștii străini, care nu sunt familiarizați cu orașul sau locația și doresc să găsească un restaurant potrivit pentru nevoile și gusturile lor, prin consultarea meniului și al recenziilor.

2. Aria de acoperire a aplicației

Ce este și ce nu este aplicația (max. ½ de pagină)

"Restaurant booking platform" permite utilizatorilor să navigheze printr-o listă de restaurante, să vizualizeze meniurile, să citească recenzii de la alți utilizatori, dar și să evalueze și să facă rezervări în timp real. Utilizatorii pot selecta data, ora și masa de pe harta restaurantului, pentru care ulterior vor fi nevoiți să confirme prezența. De asemenea, platforma le oferă clienților și alte caracteristici suplimentare, cum ar fi programele de loialitate, reducerile și ofertele speciale.

Platforma nu constituie o aplicație care doar oferă informații despre restaurante, cum ar fi meniurile, locațiile și detaliile de contact ale acestora, cât aceasta a fost dezvoltată să permită în primul rând utilizatorilor să rezerve mese la restaurantul dorit, într-o anumită dată și la o anumită oră. În mod similar, nu este o aplicație care permite utilizatorilor să comande mâncare pentru livrare sau preluare. În plus, un alt aspect notabil, este faptul că deși este o aplicație pentru rezervarea locurilor, platforma nu a fost concepută pentru rezervarea întregului restaurant pentru un anumit eveniment.

3. Tacticile arhitecturale

Pentru fiecare dintre scenariile de calitate selectate și prezentate în raportul de analiză arhitecturală, se va alege și descrie tactica sau tacticile (soluțiile, metodele de

implementare) arhitecturale adecvate.

QS1. Extensibilitate

Pentru îndeplinirea cerințelor de extensibilitate vom aplica următoarele tactici arhitecturale:

- Design orientat pe componente: această tehnică facilitează extensibilitatea prin permiterea unei dezvoltări modulare și flexibile, unde adăugarea sau modificarea unei componente nu va afecta întregul sistem. (ex: utilizarea design patterns, concepte OOP, pachetelor). Utilizarea Spring Boot pentru BE permite utilizarea tuturor conceptelor enumerate, modularizarea codului și reutilizarea componentelor, deoarece este un framework de Java, un limbaj orientat pe obiecte.
- Arhitectură bazată pe evenimente: această abordare se bazează pe producerea și consumul de evenimente în cadrul aplicației. Componentele aplicației devin independente și interacționează prin intermediul evenimentelor, ceea ce le permite să se extindă și să se modifice în mod independent.
- REST APIs: un sistem care implementeaza acest tip de API are la bază apeluri de tip HTTP. Acestea oferă un nivel ridicat de interoperabilitate și permite adăugarea și modificarea funcționalităților prin intermediul REST APIs. Utilizarea Spring Boot pentru implementarea BE facilitează dezvoltarea unei astfel de arhitecturi.

QS2. Testabilitate

Pentru a facilita testarea, aplicația va fi scrisă având în vedere următoarele aspecte:

- Design inteligibil al arhitecturii codului, urmărind standarde din industrie: REST API, n-layer architecture. Acestea fac testarea mai ușoară deoarece funcționalitățile sunt mai ușor de înțeles de către cel care testează.
- Componente de cod modulare și reutilizabile, al cărui scop să fie bine definit și focusat pe o singură responsabilitate. În acest mod, o componentă poate fi testată comprehensiv o singură dată și este suficient, chiar dacă ea va fi folosită în mai multe fluxuri de acțiuni.
- Arhitectură pe straturi (layered), care va permite abstractizarea dependințelor unei componente, pentru ca aceasta să fie testată în izolare. Spre exemplu, sursa de date (baza de date SQL) va putea fi substituită cu un mock în vederea testării.
- Mediul în care va rula aplicația va fi unul reprezentativ, similar cu cel în care va fi utilizată aplicația în producție, pentru a oferi indicatori precișsi de performanță și utilizabilitate.
- Definirea de cazuri de eroare cât mai precise pe server și întoarcerea de răspunsuri HTTP detaliate la întâlnirea acestor cazuri. Astfel, la testare se va putea identifica exact dacă aplicația are comportamentul așteptat la apelarea oricărui endpoint cu oricare date de input.

QS3. Performanța

Pentru îndeplinirea cerințelor de performanța vom aplica următoarele tactici arhitecturale:

- Vom controla fluxul de resurse: În cadrul implementării, atât pe BE și FE, vom face numai operații strict necesare pentru furnizarea unui răspuns, iar aceste operații vor folosi principii de codare agreate de tehnologiile folosite astfel încât operațiile să fie făcute cât

mai eficient. Pe FE în special se va lua în considerare reducerea numărului de apeluri pentru reconstruirea elementelor, astfel încât impactul computațional pentru afișarea componentelor să fie minim.
- Vom gestiona resursele într-un mod constructiv, acolo unde este posibil vom încerca să păstrăm un comportament asincron asupra operațiilor executate, astfel încât să ne asigurăm că într-un anumit interval de timp ne folosim cât mai inteligent de resursele de care dispunem. În ceea ce privește datele de care dispunem, în cazul în care putem păstra în memorie anumite instanțe care ne ajută să procesăm mai repede unele operații, vom compara costurile de păstrare în memorie cu potențialele costuri dacă facem operațiile fără a integra acest aspect și vom alege varianta mai eficientă.
- În cazul asincron menționat mai sus, va trebui să avem în vedere modul în care gestionăm prioritățile de execuție a operațiilor. Vom încerca să menținem un sistem în care operațiile non prioritare nu vor bloca stiva de apeluri.

QS4. Utilizabilitate

Tacticile arhitecturale folosite în asigurarea îndeplinirii cerintei de calitate Utilizabilitate in cadrul dezvoltării aplicației vor fi:

- La design time:
  - **Incremental design process**: Pe parcursul întregii dezvoltări se vor aplica tehnici de iterare succesiva si prototipare pentru a genera interfața potrivită aplicației
- La run time:
  - **Feedback constant**:
    - Vom implementa un sistem de notificări de tip toast (mesaje scurte care apar în colțul paginii) care va atentiona utilizatorii atunci cand este nevoie.
    - Vom folosi componente vizuale(butoane, drawers etc) cu animatiile specifice paradigmelor de design modern, a.i. utilizatorul sa poata avea o intelegere intuitiva asupra modului in care se interactioneaza cu aplicatia.
  - **Responsive design**:
    - Vom apela atât la biblioteci care usureaza lucrul cu pagini ce trebuie randate pe mai multe platforme (precum ChakraUI), cât și la proprietăți ale CSS-ului care permit randarea diferită în funcție de criterii precum rezoluția ecranului utilizatorului.
-

4. Specificațiile de proiectare

> Specificațiile de analiză sub formă de user stories se vor descompune în specificații mai fine care să descrie tacticile arhitecturale adoptate mai sus pentru fiecare dintre scenariile de calitate.

QS1. Extensibilitate

Pentru îndeplinirea cerințelor de Extensibilitate am avut în vedere aplicarea unor tehnici arhitecturale care să faciliteze dezvoltarea ulterioară. Cele mai notabile modificări au fost aplicate în cadrul următoarelor aspecte din user stories:

- US1. În sistemul de alegere al unuia dintre restaurantele listate, userul își dorește să găsească cât mai repede ceea ce-și dorește folosind anumite criterii personalizate. Pentru aceasta va fi nevoie de atașarea unui sistem de recomandări și filtre pe lista de restaurante, astfel fiind nevoie de extinderea funcționalității inițiale, de bază. Ținând cont încă de la structurarea designului că nevoile clienților se vor schimba și va trebui să adăugăm noi features pe parcurs, am ales să urmărim cu atenție următoarele aspecte:
    - Dezvoltarea BE utilizând Spring Boot și structurarea aplicației în entități, servicii și controlere va permite adăugarea foarte ușoară a unui nou API de filtrare și/sau sortare în funcție de anumite criterii, Nu va fi nevoie de adăugarea unei noi clase sau crearea unui nou pachet, ci în controlerul care randează lista inițială de restaurante vor fi inserate funcțiile noi de filtrare.
    - Implementarea unui sistem de recomandare va presupune folosirea conceptelor de machine learning și a limbajului Python pentru construirea unui model de prezicere bazat pe un anumit set de date de antrenare. Modularitatea aplicației atât pe BE, cât și pe FE, cât și utilizarea limbajelor populare pentru acestea vor permite foarte ușor integrarea și executarea unui astfel de model extern.

QS2. Testabilitate

Pentru îndeplinirea cerințelor de Testabilitate am avut în vedere aplicarea unor tehnici arhitecturale care să faciliteze interacțiunea dintre utilizator și aplicație. Cele mai notabile modificări au fost aplicate în cadrul următoarelor aspecte din user stories:

- US3. Cerința funcțională de confirmare prin email / SMS se va implementa având în vedere ca serviciile externe de email / SMS să poată fi decuplate de logica de business care le apelează, pentru a se putea observa inputul care se va trimite către aceste servicii; astfel, se va putea testa logica de trimitere de email / SMS atât unitar cât și în integrare cu serviciul ales.
- US5. Strategia de autentificare și autorizare trebuie proiectată astfel încât la automatizarea testelor, dovada autorizării să poată fi obținută programatic de către software-ul care rulează testele. Astfel, funcționalitățile se vor putea testa end-to-end atât cu și fără autentificare în prealabil și se vor identifica erori de autorizare.

QS3. Performanța

Pentru îndeplinirea cerințelor de Performanță am avut în vedere aplicarea unor tehnici arhitecturale care să faciliteze interacțiunea dintre utilizator și aplicație. Cele mai notabile modificări au fost aplicate în cadrul următoarelor aspecte din user stories:

- US1. În ceea ce privește listarea restaurantelor s-au avut în vedere următoarele:
    - pe BE s-a folosit un query eficient pentru extragerea datelor relevante, fără să se adauge operații suplimentare irelevante. Pentru reducerea dimensiunii payload-ului s-a făcut și un sistem de paginare, astfel încât să selectăm doar parțial restaurantele
    - pe FE s-au randat în pagină cardurile folosindu-ne de sistemul de paginare, astfel încât să nu încărcăm pagina cu prea multe restaurante care să fi îngreunat experiența de utilizare. O altă strategie, folosită în special pentru mobil, ar fi fost integrarea sistemului de infinite scrolling, sau prin cel de lazy loading, în care

încărcăm în pagină numai cardurile de care avem nevoie și distrugem cardurile care se află în afara ecranului
    - Pentru afișarea în sine a cardurilor de restaurant s-a avut în vedere afișarea unor informații restrânse, astfel încât să reducem dimensiunea răspunsului primit de pe server
- US2. În ceea ce privește realizarea unei rezervări, au fost luate în considerare următoarele:
    - pentru a nu îngreuna experiența utilizatorului în cazul evenimentelor concurente, am încercat să păstrăm un comportament asincron, astfel încât să nu se formeze cozi pentru rezervări. Atunci când un utilizator selectează niște mese pentru rezervare, acea instanță va rămâne rezervată pentru utilizator, încât să nu se ajungă la un race condition.
    - pentru randarea meselor s-a comparat costul de procesare împreună cu costul de stocare a unor informații complexe de amplasament ale elementelor grafice în pagină. Pentru a nu extinde sistemul cu o componenta de tip storage pentru imagini, s-a decis stocarea în baza de date a coordonatelor și configurației meselor pentru structura restaurantului, astfel încât să poată fi calculată la afișare în pagina de rezervare. Calcularea amplasamentului grafic se calculează doar o singură dată pentru a nu îngreuna memoria. Elementele pot fi stocate pe timpul execuției procesului de rezervare.

QS4. Utilizabilitate

Din perspectiva tacticilor arhitecturale folosite pentru a asigura îndeplinirea cerintei Utilizabilitate, se remarca următoarele aspecte din user story-urile stabilite în raportul arhitectural:

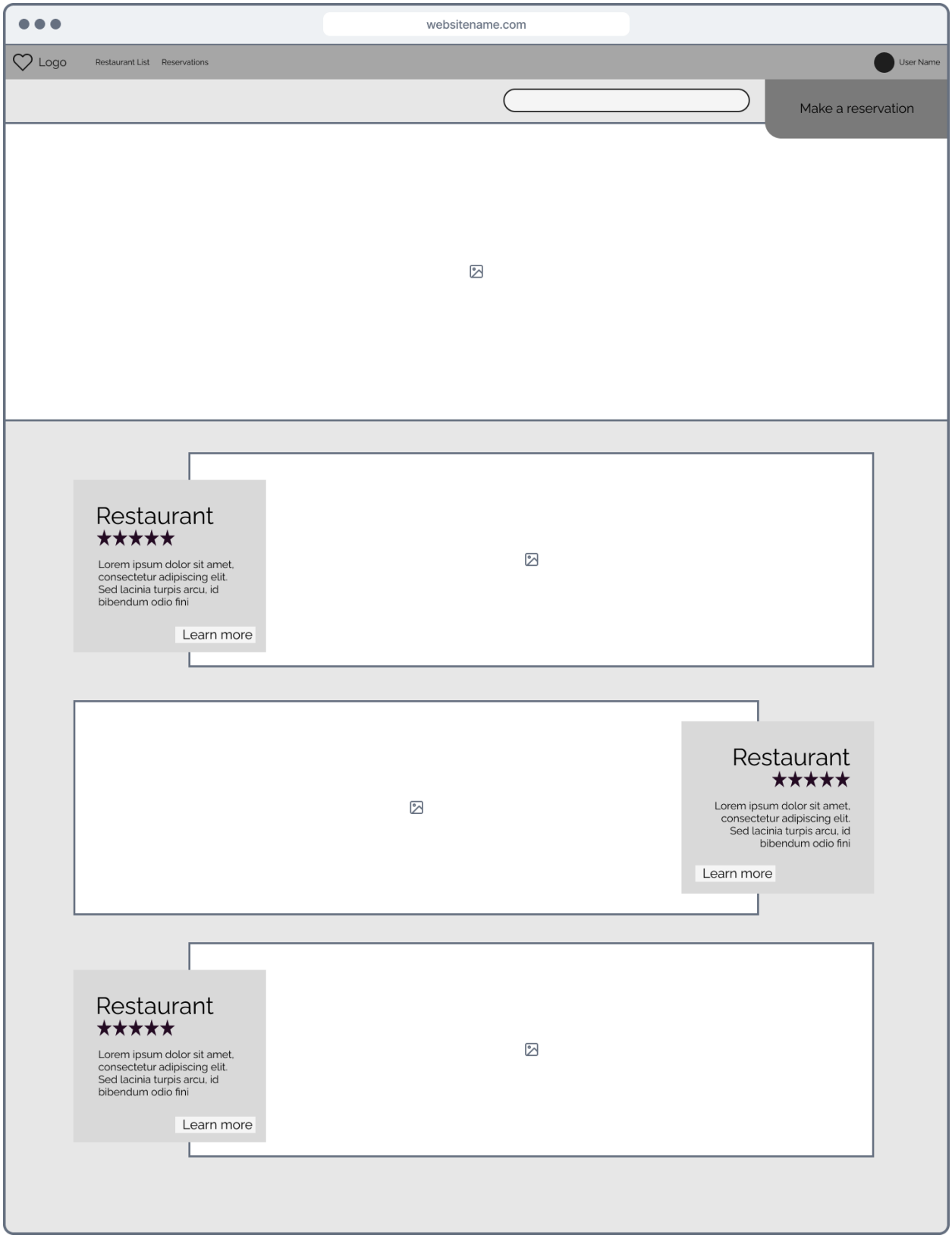
- US2. Întregul flow de creare a rezervarii la un restaurant, integral pentru succesul platformei, trebuie sa fie cat mai ușor de folosit posibil. Pentru a îndeplini acest lucru, atenție deosebită trebuie acordată utilizabilitatii. Asadar:
    - Interfata trebuie sa ghideze utilizatorul în timpul întregului proces. Vom apela la o solutie tip “step wizard”, un pattern folosit des în aplicațiile prezentului.
    - Interfata trebuie sa semnaleze eventualele inadvertente în completarea formularului de rezervari si sa notifice utilizatorul cu mesaje utile, care sa ajute la remedierea problemei.
    - Harta pentru selectarea locurilor trebuie sa fie clar de urmărit, să distingă între mesele disponibile, mesele ocupate și mesele aflate în selecția curentă a utilizatorului.
- US1. Plecând tot de la ideea familiaritatii, în implementare propunem o soluție care se bazează pe componenta de design “Card”, popularizata de catre Material Design. Asadar, ne dorim ca lista restaurantelor sa se înfățișează sub forma unei liste de elemente tip Card, ce contin o imagine, numele restaurantului, rating si o scurta descriere a sa. Apasarea pe Card va deschide pagina detaliată a restaurantului.

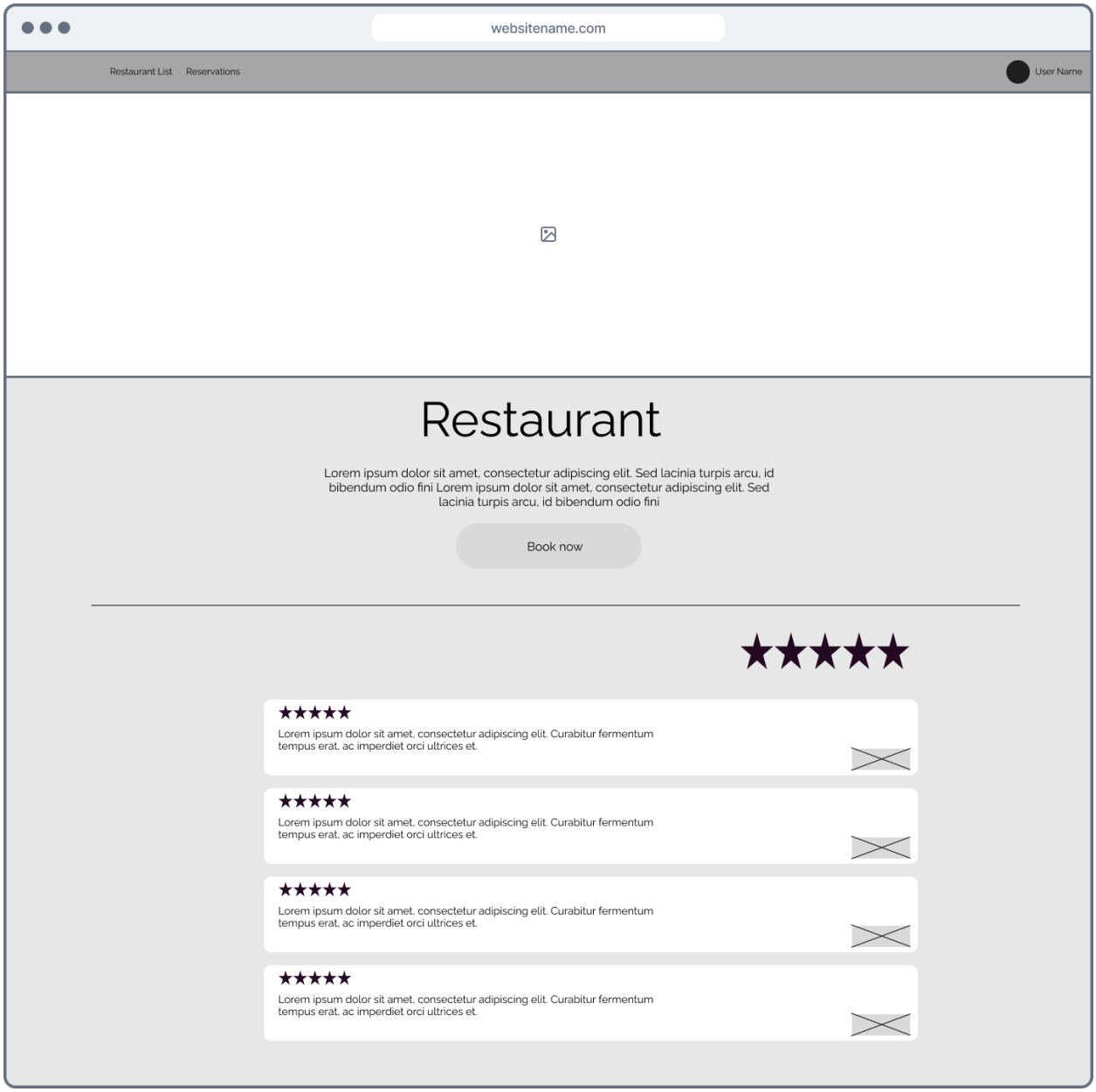
5. **Perspectiva de utilizare**

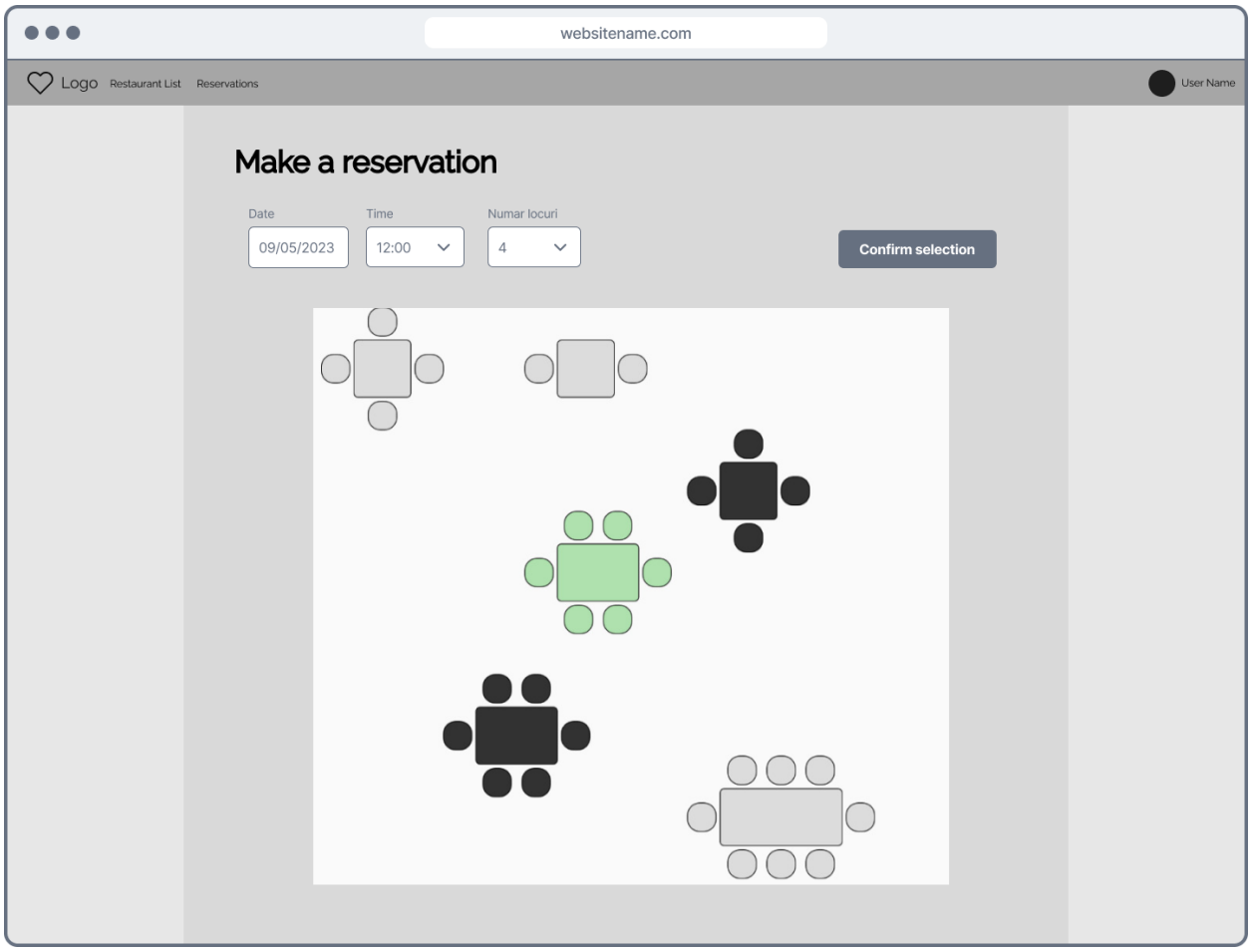
Se va descrie perspectiva arhitecturală asupra aplicației care se referă la interacțiunea utilizatorului cu aplicația, evidențiindu-se aspectele relevante pentru acoperirea specificațiilor de analiză sau proiectare selectate anterior. Se pot folosi diagrame de stări UML sau wireframes de interfață.

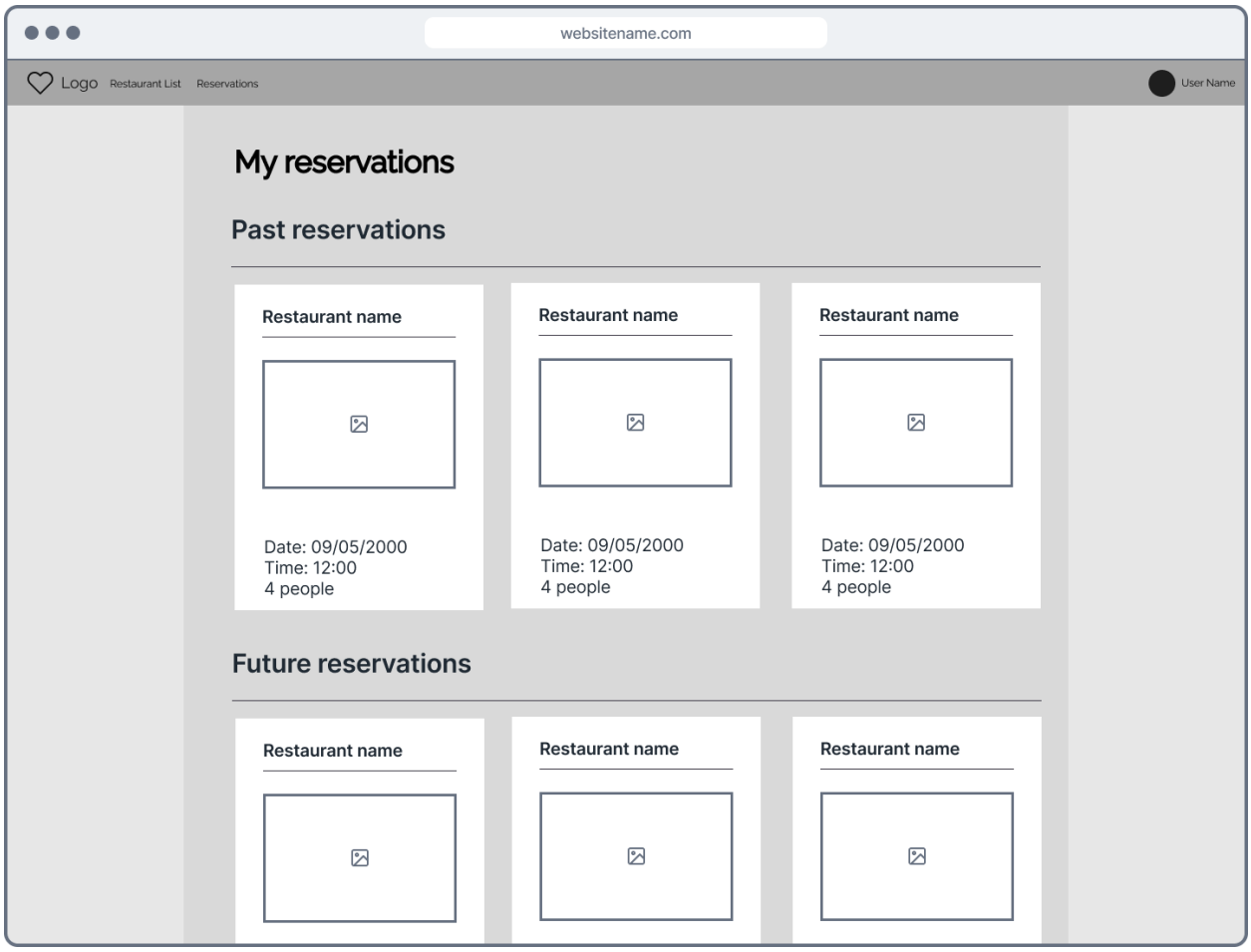
Din perspectiva interacțiunii utilizatorului cu aplicația, aspecte relevante pentru specificatiile de analiza sunt:

- Legat de: US3. Confirmări atunci unde sunt importante. Dorim ca utilizatorul să aibă siguranța faptului că rezervarea sa a fost înregistrată de către restaurant, fiind un punct critic al aplicației. La polul opus, managerii de restaurante își doresc să fie siguri ca rezervările din aplicație vor fi onorate. Asadar, vom implementa un sistem de dubla confirmare, prin care utilizatorul va trebui sa confirme folosind email/sms rezervarea sa.
- Legat de: US1/2. Interfața familiară. În faza de proiectare, am întocmit wireframes ale aplicației pentru a putea imagina flow-urile principale ale end-userilor și pentru a putea dezvolta elemente ale aplicației în paralel.
- Legat de: US4. Incredere. Utilizatorilor le este mai ușor sa facă o decizie atunci cand sunt informați și cand stiu ca informatia este reala. Asadar, am decis că sistemul de recenzii al restaurantelor sa permita oferirea unei note doar atunci cand rezervarea a fost confirmată și onorata prin intermediul sistemului nostru.
- Legat de: US5: Personalizare si istoric. Prin apelarea la pattern-ul sistemului de conturi, putem oferi utilizatorilor o experiența personalizată, în funcție de preferințele lor.


websitename.com
Logo
Restaurant List
Reservations
User Name
Make a reservation
Restaurant
★★★★★
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed lacinia turpis arcu, id bibendum odio fini
Learn more
Restaurant
★★★★★
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed lacinia turpis arcu, id bibendum odio fini
Learn more
Restaurant
★★★★★
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed lacinia turpis arcu, id bibendum odio fini
Learn more


websitename.com
Restaurant List
Reservations
User Name
Restaurant
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed lacinia turpis arcu, id bibendum odio fini Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed lacinia turpis arcu, id bibendum odio fini
Book now
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur fermentum tempus erat, ac imperdiet orci ultrices et.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur fermentum tempus erat, ac imperdiet orci ultrices et.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur fermentum tempus erat, ac imperdiet orci ultrices et.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur fermentum tempus erat, ac imperdiet orci ultrices et.

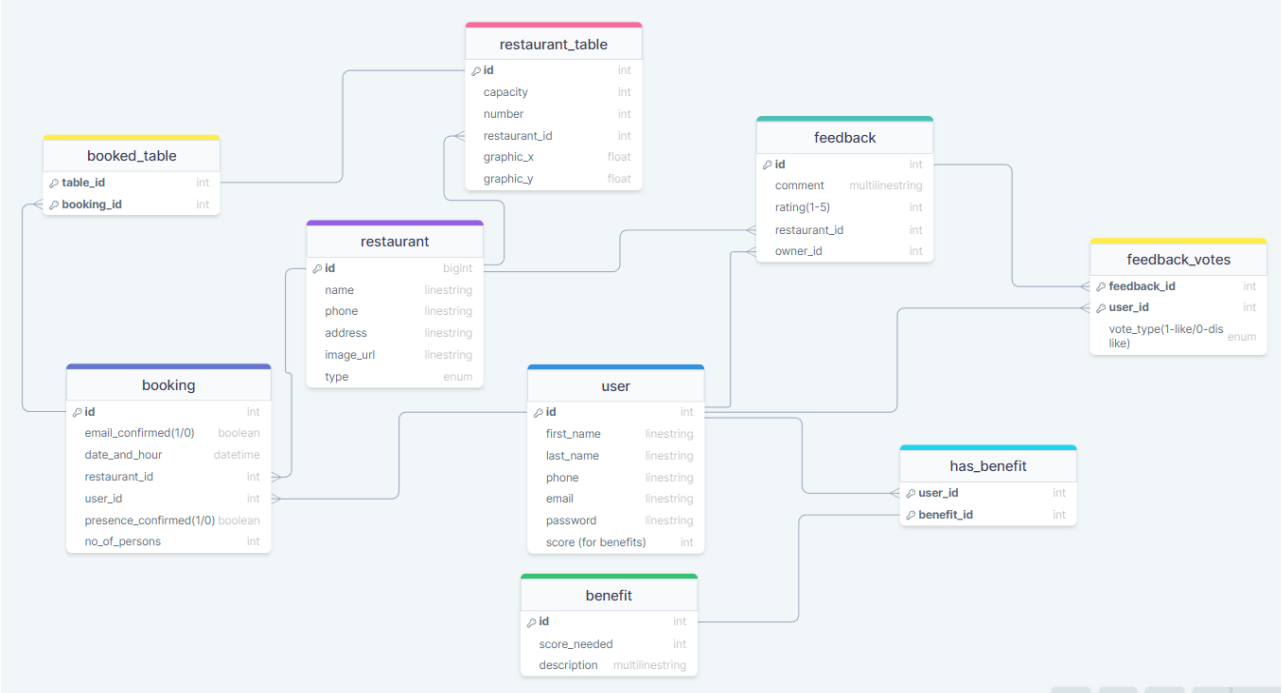

websitename.com
Logo
Restaurant List
Reservations
User Name
Make a reservation
Date
09/05/2023
Time
12:00
Numar locuri
4
Confirm selection



6. **Perspectiva asupra datelor**

Se va descrie perspectiva arhitecturală asupra aplicației care se referă la modelul conceptual al datelor utilizate în aplicație, evidențiindu-se aspectele relevante pentru acoperirea specificațiilor de analiză sau proiectare selectate anterior. Se pot folosi diagrame de date UML sau alte diagrame de prezentare adecvate modelului de date folosit.

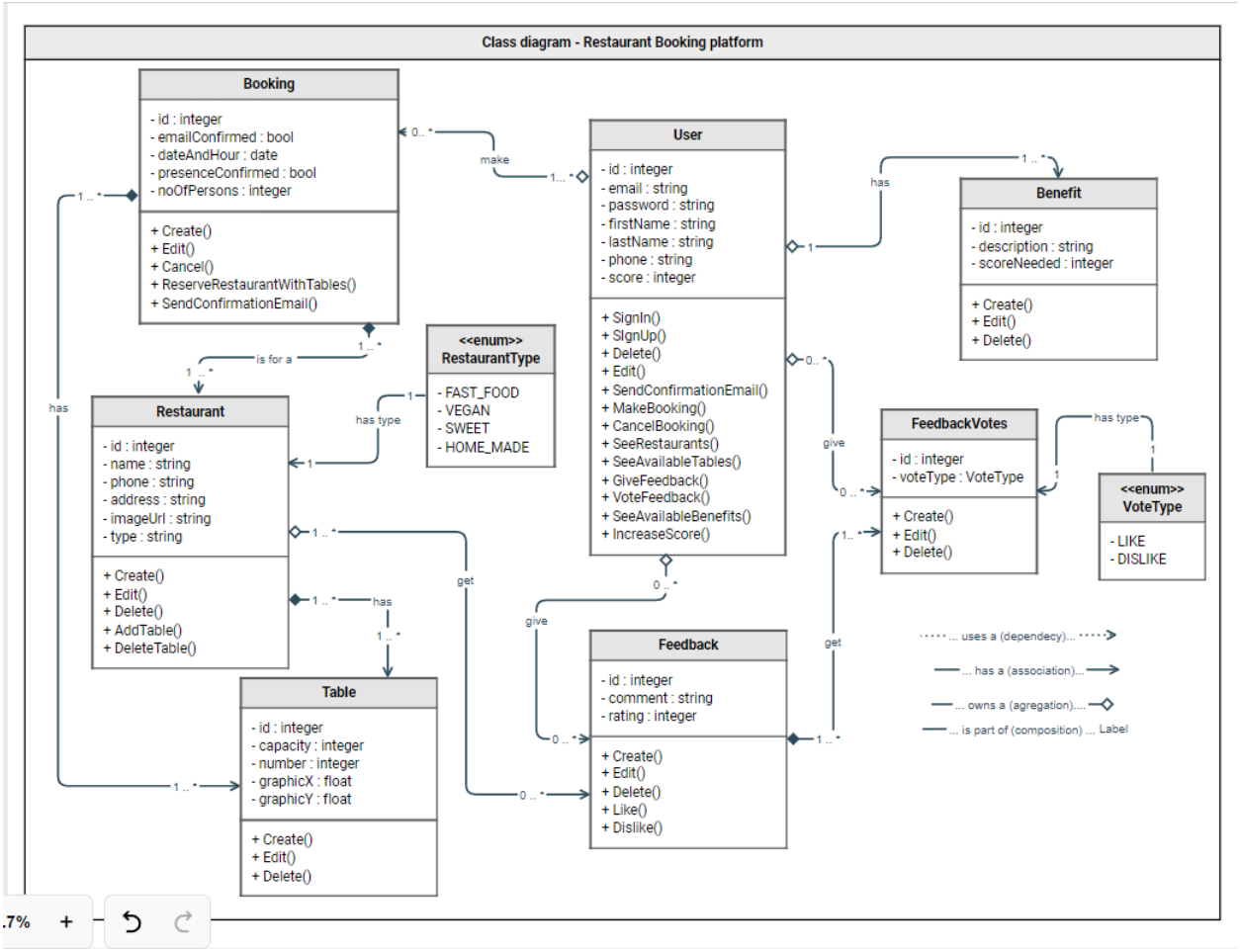
Fiind o aplicație web destinată rezervărilor la restaurante nu va presupune stocarea unor cantități de date prea mare. Astfel, modele de date pot fi reprezentate utilizând MySQL ca sistem de gestionare a bazelor de date, prin intermediul căruia vor fi stocate și gestionate informațiile despre restaurante, utilizatori, rezervări, mese, beneficii și recenzii. Așa cum se poate observa și în diagrama UML de mai jos există 6 entități principale și 3 entități secundare, folosite sub forma unor tabele de legătură în relațiile de tip many-to-many. În plus, Spring Boot se îmbină perfect cu MySQL, având deja plugins dedicate integrate, fiind ușor de folosit împreună cu data transfer objects (DTOs) și clasele de tip POJO.

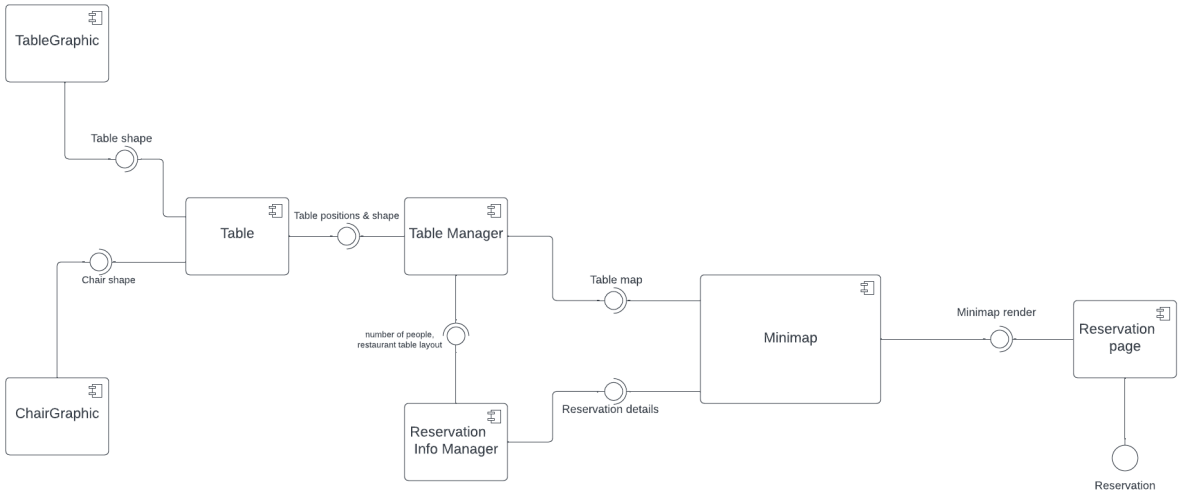


7. **Perspectiva structurală**

Se va descrie perspectiva arhitecturală asupra aplicației care se referă la componentele ei principale și modul de conectare a acestora, evidențiindu-se aspectele relevante pentru acoperirea specificațiilor de analiză sau proiectare selectate anterior. Se pot folosi diagrame structurale UML sau sau alte diagrame de prezentare adecvate.

Diagrama claselor arată structurarea aplicației în clase, enums, interfețe și relațiile dintre ele, oferind o înțelegere a modului în care componentele sunt organizate și cum colaborează între ele. Deși diagrama claselor este foarte asemănătoare cu diagrama bazei de date, aceasta oferă informații suplimentare despre modul de funcționare la nivel structural prin adăugarea metodelor operaționale pentru fiecare entitate în parte, dar și vizualizarea dependențelor între clase, cum ar fi asocierea, compunerea sau extinderea. Din schema UML reprezentată mai jos se pot observa anumite detalii importante ale structurii aplicației cum ar fi categoriile de restaurante pe care le implementează, operațiile specifice utilizatorilor sau faptul că anumite entități nu pot exista unele independent de celelalte (mesele nu pot exista fără a fi asociate cu un anumit restaurant sau un vot nu poate exista fără a fi asociat cu un anumit feedback).





8. **Perspectiva comportamentală**

Se va descrie perspectiva arhitecturală asupra aplicației care se referă la modul intern de

funcționare a acesteia, evidențiîdu-se aspectele relevante pentru acoperirea specificațiilor de analiză sau proiectare selectate anterior. Se pot folosi diagrame comportamentale UML, cum ar fi diagrame de stări, de activități sau de tranziții.

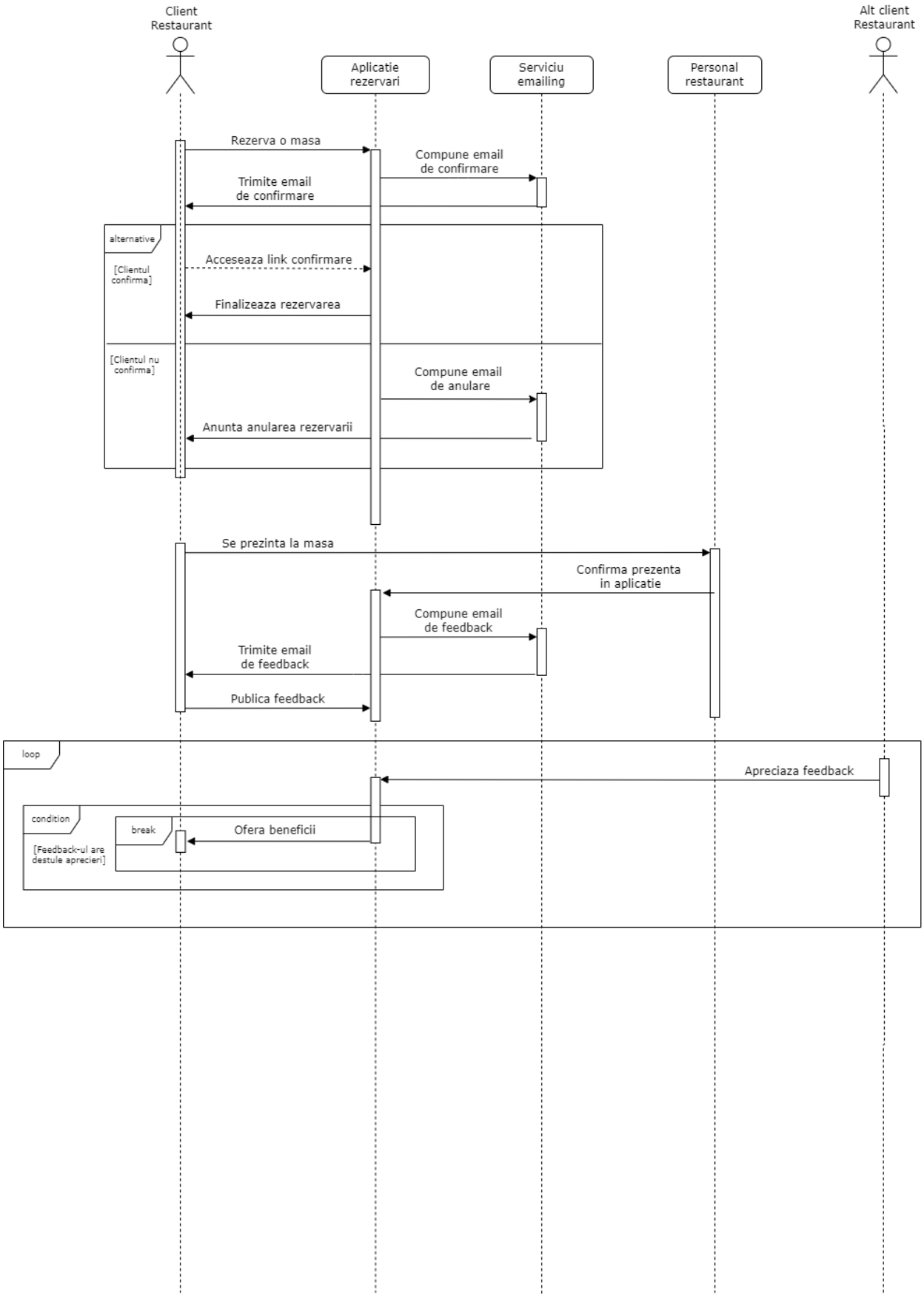
Având în vedere specificul aplicației, comportamentul este definit în jurul sistemului de rezervări. Întreaga dinamică a sistemului se învârte în jurul diferitelor stadii ale acestui proces.

Inițial utilizatorul poate să parcurgă o listă de restaurante, urmând apoi să își aleagă unul dintre ele în funcție de preferințe și poate și în funcție de recenzii și scorul general al restaurantului.

Următorul sistem integrat este pagina de realizare a rezervării: utilizatorul poate să selecteze o oră, numărul de persoane pentru care vrea să facă rezervarea, dar și care dintre mese dorește, în funcție de amplasarea meselor în restaurant.

Dacă rezervarea este validă, apoi se trece în stadiul de confirmare, în care utilizatorul primește un email prin care trebuie să confirme că își dorește rezervarea pe care a inițiat-o.

După ce rezervarea a fost făcută și utilizatorul a mers la restaurant, acesta poate lăsa o recenzie restaurantului la care tocmai a fost. Dacă utilizatorul este suficient de activ în ceea ce privește recenziile lăsate și suficienți utilizatori votează pentru recenzia lui, atunci acesta va primi anumite beneficii în aplicație, cum ar fi posibilitatea selectării unui număr mai mare de mese sau pentru mai multe persoane.


Client Restaurant
Aplicatie rezervari
Serviciu emailing
Personal restaurant
Alt client Restaurant
Rezerva o masa
Compune email de confirmare
Trimite email de confirmare
alternative
[Clientul confirma]
Acceseaza link confirmare
Finalizeaza rezervarea
[Clientul nu confirma]
Compune email de anulare
Anunta anularea rezervarii
Se prezinta la masa
Confirma prezenta in aplicatie
Compune email de feedback
Trimite email de feedback
Publica feedback
loop
Apreciaza feedback
condition
[Feedback-ul are destule aprecieri]
break
Ofera beneficii

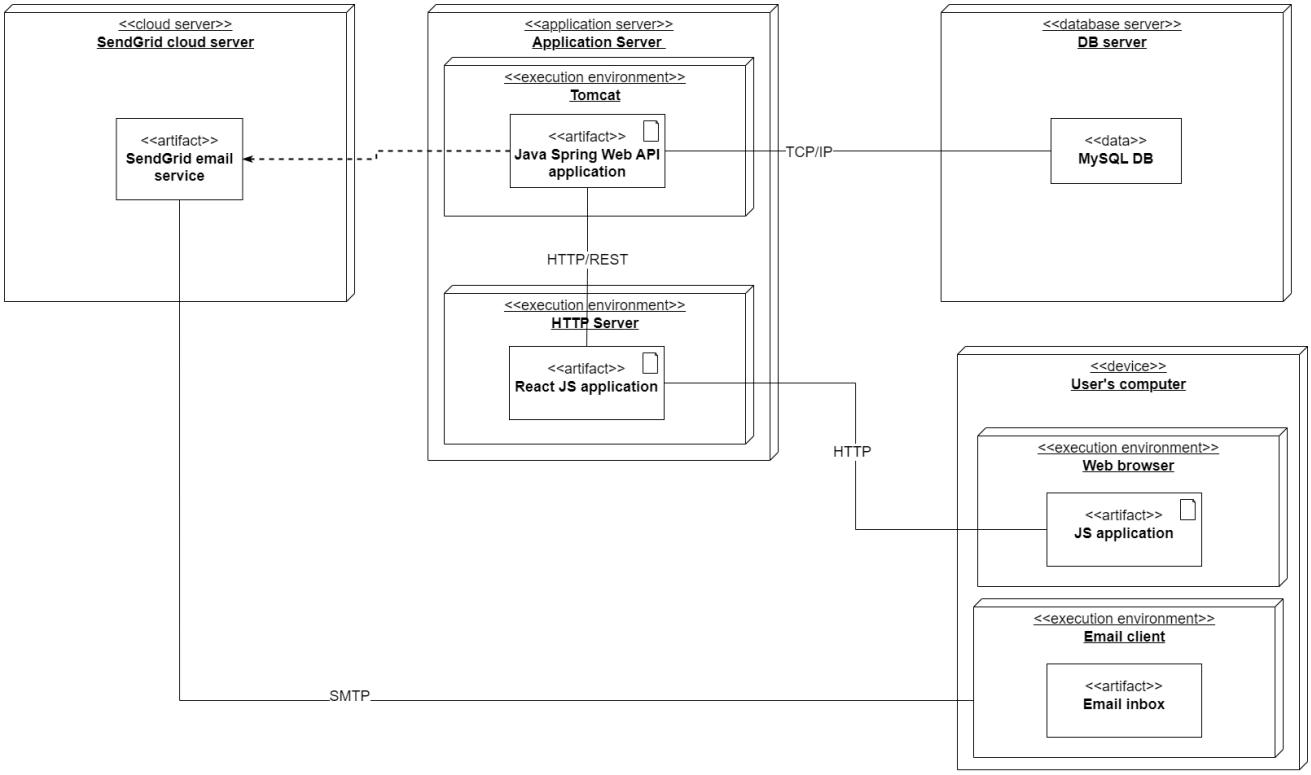
9. **Perspectiva fizică**

Se va descrie perspectiva arhitecturală asupra aplicației care se referă la mediul (software, hardware și de rețea) în care va funcționa aplicația, evidențiindu-se aspectele relevante pentru acoperirea specificațiilor de analiză sau proiectare selectate anterior. Se pot folosi diagrame de deployment UML.

Resursele necesare pentru rularea aplicației sunt un server de baze de date și două servere web, unul pentru backend și unul pentru frontend. Toate acestea pot fi ținute pe un server propriu sau în cloud. Aplicația client comunică cu aplicația server prin protocolul HTTPS.

Conform stack-ului de tehnologii ales, baze de date va fi hostată pe un server MySQL. Serverul backend implementat în Spring Boot va rula pe un environment Java va expune un server web Tomcat pe un port HTTPS. Aplicația frontend va rula pe un environment Node JS.

Printre cerințele funcționale se numără trimiterea de email-uri sau SMS-uri pentru confirmarea rezervării, drept pentru care apelăm la o subscripție la un serviciu extern oferit de SendGrid. Serverul va trimite request-uri către SendGrid prin HTTP, care mai departe va expedia email-uri către inbox-ul utilizatorului.



10. Perspectiva asupra testării

Se va descrie perspectiva arhitecturală asupra testării aplicației, evidențiindu-se aspectele relevante pentru acoperirea specificațiilor de analiză sau proiectare selectate anterior.

Strategia de testare va consta majoritar din testare funcțională, pentru a asigura că soluția acoperă nevoile de business cerute iar fluxurile de interacțiune ale utilizatorului sunt corecte. Un accent important trebuie pus pe sistemul de rezervări, unde trebuie să se asigure integritatea și consistența datelor atunci când mai mulți utilizatori creează concurent rezervări pentru aceleași locuri în același interval de timp.

În plus, va fi nevoie de testare non-funcțională de performanță. Se vor stabili timpi rezonabili de răspuns la acțiuni uzuale (listare entități, creare rezervări, recenzii etc.) care vor trebui îndepliniți atunci când aplicația rulează pe un mediu de capacitate similară cu cel de producție.

Din punct de vedere al software lifecycle-ului și prin paralelă la perspectiva asupra dezvoltării, testarea unui feature va avea loc în primă fază la finalizarea unei unități de muncă (user story). Validarea testelor funcționale va fi una dintre condițiile care fac parte din "definition of done". Ulterior se vor scrie teste automate pentru standardizarea și automatizarea acestor cazuri de test. Acestea se vor putea rula periodic pentru asigurarea funcționării pe termen lung și prinderea oricăror bug-uri apărute în urma unor dezvoltări ulterioare (smoke & regression testing). Teste adiționale de regresie vor avea loc înaintea release-ului unei versiuni de aplicație.

11. Perspectiva asupra dezvoltării

Se va descrie perspectiva arhitecturală asupra dezvoltării aplicației, evidențiîdu-se aspectele relevante pentru acoperirea specificațiilor de analiză sau proiectare selectate anterior. În acest scop, aplicația va fi descompusă în componente de dezvoltare, care pot fi atât structurale, cât și comportamentale, plecând de la specificațiile de analiză și proiectare definite sub formă de user stories. Aceste componente vor fi planificate pe un grafic de dezvoltare format din iterații succesive de câte 2 săptămâni (sprinturi).

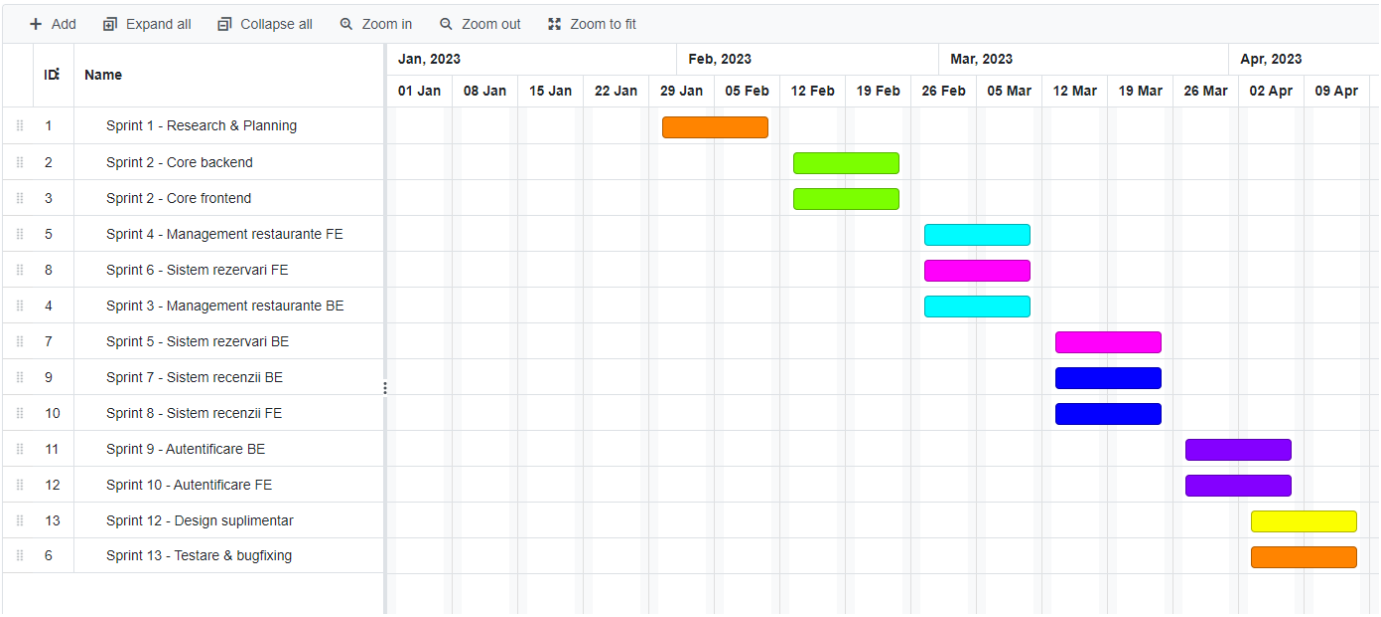
În ceea ce privește perspectiva arhitecturală asupra dezvoltării, putem împărți aplicația în două părți principale: server și interfață client, sau Backend și Frontend. Fiecare dintre cele două echipe a avut câte doi membri.

Pentru realizarea proiectului, am optat, drept tehnologii de implementare, pentru Spring (Java) pentru Backend și React (Typescript) pentru Frontend. Tehnologiile au fost alese atât din punct de vedere arhitectural, pentru beneficiile de care dispun în ceea ce privește viteza de implementare și performanța componentelor generate prin intermediul lor, cât și din punct de vedere al resurselor, astfel că am ales tehnologii cunoscute, cu suport continuu și care au informații relevante disponibile, atât oficial, prin intermediul documentațiilor, cât și din punct de vedere social, prin intermediul discuțiilor din forumuri.

În ceea ce privește aspectul comportamental al aplicației, am optat pentru un flow natural, având ca punct de referință aplicații asemănătoare ce se folosesc pe scară largă, având în vedere să respectăm standardele de UI/UX, atât din punct de vedere vizual, prin indicatori ce inspiră exact la ceea ce se referă acțiunea realizată, cum ar fi folosirea culorilor potrivite pentru erori, avertizări și operații realizate cu succes, cât și din punct de vedere structural, prin folosirea unui sistem de notificări prin care putem transmite utilizatorului informații utile.

Privind această perspectivă într-un mod mai specificat, putem să ne concentrăm pe deciziile luate pentru integrarea user story-urilor:

- US1: Pentru listarea restaurantelor s-a optat pentru implementarea unei pagini ce prezintă o listă de elemente sub formă de componentă de tip Card ce prezintă informații relevante pentru utilizator, surprinsă într-un mod restrâns, astfel încât să afișăm doar informații necesare.
- US2: Pentru realizarea paginii de rezervare s-au realizat următoarele elemente:
    - în pagina de afișare a restaurantului putem vizualiza date extinse, atât din partea managerului de restaurant care a scris descrierea acestuia, cât și din partea comunității care a oferit un scor și o părere, cuprinse în lista de review-uri
    - dacă utilizatorul decide că acel restaurant este ceea ce își dorește va apăsa pe butonul de rezervare unde va putea să selecteze informațiile relevante completând un element de tip formular
    - după ce datele au fost completate, se va afișa o fereastră legată la un sistem grafic care afișează amplasamentul meselor în restaurant. Utilizatorul va putea selecta orice masă disponibilă (fapt evidențiat prin folosirea de culori ce induc starea de liber/ocupat), sau chiar și mai multe, până se ajunge la numărul de persoane pentru care se realizează
- US3: După ce o rezervare a fost făcută, pentru a confirma autenticitatea persoanei care a inițiat-o, aceasta trebuie să confirme prin intermediul unui SMS sau a unui email pentru ca rezervarea să devină validă, astfel încât să ne asigurăm că majoritatea utilizatorilor care fac rezervări sunt utilizatori legitimi. Acest aspect structural de confirmare poate fi dublat și ca element de securitate, pentru că adaugă un strat suplimentar de siguranță pentru atacuri de tip DDoS. Această funcționalitate se va integra pe partea de Backend, folosind tehnologii de emailing, cum ar fi Mailgun, sau SMS.
- US4: Pentru a confirma validitatea recenziilor lăsate de utilizatori, prezența acestora trebuie să fie confirmată în cadrul restaurantului. Acest sistem poate avea mai multe direcții de implementare. Un sistem rudimentar este să se valideze automat o rezervare după ce timpul pentru care era plasată a trecut.
- US5: Pentru a stoca informații relevante pentru utilizatori, cum ar fi istoricul rezervărilor, sau ce recenzii a lăsat, sau pentru aplicarea de beneficii în urma primirii de aprecieri pe recenzii, aplicația dispune de un sistem de autentificare. În acest fel informațiile se pot gestiona mult mai ușor, atât din partea utilizatorilor, cât și din partea sistemului.


Add
Expand all
Collapse all
Zoom in
Zoom out
Zoom to fit
ID
Name
Jan, 2023
Feb, 2023
Mar, 2023
Apr, 2023
01 Jan
08 Jan
15 Jan
22 Jan
29 Jan
05 Feb
12 Feb
19 Feb
26 Feb
05 Mar
12 Mar
19 Mar
26 Mar
02 Apr
09 Apr
1 Sprint 1 - Research & Planning
2 Sprint 2 - Core backend
3 Sprint 2 - Core frontend
5 Sprint 4 - Management restaurante FE
8 Sprint 6 - Sistem rezervari FE
4 Sprint 3 - Management restaurante BE
7 Sprint 5 - Sistem rezervari BE
9 Sprint 7 - Sistem recenzii BE
10 Sprint 8 - Sistem recenzii FE
11 Sprint 9 - Autentificare BE
12 Sprint 10 - Autentificare FE
13 Sprint 12 - Design suplimentar
6 Sprint 13 - Testare & bugfixing