

UNIVERSITATEA „LUCIAN BLAGA” DIN SIBIU  
FACULTATEA DE INGINERIE  
DEPARTAMENTUL DE CALCULATOARE ȘI INGINERIE  
ELECTRICĂ

# PROIECT DE DIPLOMĂ

*Coordonator Științific:* **Conf. dr. ing. Sima Dorin**

*Absolvent:* **Adrian CIOBOTEA**

*Specializarea:* Calculatoare

UNIVERSITATEA „LUCIAN BLAGA” DIN SIBIU  
FACULTATEA DE INGINERIE  
DEPARTAMENTUL DE CALCULATOARE ȘI INGINERIE  
ELECTRICĂ

# **SISTEM DE APLICAȚII PENTRU DIGITALIZAREA PROCESULUI DE COMANDĂ DIN RESTAURANTE**

*Coordonator Științific:* **Conf. dr. ing. Sima Dorin**

*Absolvent:* **Adrian CIOBOTEA**

*Specializarea:* Calculatoare

# Cuprins

<b>SISTEM DE APLICAȚII PENTRU DIGITALIZAREA PROCESULUI DE COMANDĂ DIN RESTAURANTE.....</b>	<b>1</b>
<b>CUPRINS.....</b>	<b>1</b>
<b>CAPITOLUL I. INTRODUCERE .....</b>	<b>4</b>
1.1 Abstract.....	4
1.2 Contextul proiectului .....	4
1.3 Ideea si scopul proiectului .....	5
1.4 Studiu state of the art.....	5
<b>CAPITOLUL II. TEHNOLOGII UTILIZATE.....</b>	<b>6</b>
2.1 Java.....	6
2.1.1 Java SE.....	6
2.1.2 Java EE .....	7
2.1.3 Java ME .....	7
2.1.4 Java FX .....	7
2.2 Swagger .....	8
2.3 Spring .....	10
2.3.1 Injectarea dependențelor.....	11
2.3.2 Inversiunea de control (IoC – Inversion of Control) .....	12
2.3.3 Programarea orientată pe aspecte(AOP – Aspect Oriented Programming)..	12
2.4 Spring Boot .....	13
2.4.1 Avantaje ale microserviciilor.....	13
2.4.2 Spring Data .....	14
2.4.3 Spring Security .....	14
2.5 Angular .....	14
2.5.1 Diferențe între Angular și AngularJS: .....	15
2.6 JavaScript(JS) .....	15
2.6.1 Securitatea unui browser .....	17
2.6.2 Ordinea de execuție a instrucțiunilor JavaScript .....	17
2.6.3 Cod interpretat versus cod compilat.....	17
2.6.4 Cod pe parte de client și cod pe parte de server .....	17
2.6.5 Cod dinamic versus cod static .....	18
2.7 TypeScript.....	18
2.7.1 Originile, credibilitatea și fiabilitatea TypeScript .....	18
2.7.2 Tipurile de date .....	19
2.7.3 Interfețe.....	19
2.8 HTML (Hypertext Markup Language) .....	19
2.8.1 Anatomia unui element HTML .....	20

.....	20
2.8.2 Anatomia unui document HTML .....	21
2.8.3 Verificarea corectitudinii unui document HTML.....	22
<b>2.9 CSS (Cascading Style Sheets).....</b>	<b>22</b>
2.9.1 Sintaxa CSS .....	22
2.9.2 Module CSS.....	23
<b>2.10 Angular Material.....</b>	<b>24</b>
2.10.1 De ce Angular Material Design .....	24
<b>2.11 Redux.....</b>	<b>24</b>
2.11.1 De ce am folosi Redux în aplicația noastră Angular .....	24
<b>2.12 Bootstrap.....</b>	<b>25</b>
<b>2.13 Postman.....</b>	<b>26</b>
2.13.1 Coduri de răspuns .....	26
2.13.2 Colecții.....	26
2.13.3 Medii de lucru.....	27
<b>2.14 WebClient .....</b>	<b>27</b>
<b>2.15 Feign Client.....</b>	<b>28</b>
2.15.1 WebClient vs Feign Client.....	28
<b>2.16 Baze de date .....</b>	<b>29</b>
2.16.1 MySql .....	29
2.16.2 MySql Workbench.....	30
<b>2.17 Hibernate .....</b>	<b>31</b>
2.17.1 Beneficiile Hibernate .....	31
2.23.1 Cum funcționează Hibernate?.....	32
2.17.2 De ce să folosiți Hibernate? .....	32
2.17.3 Hibernate și JPA versus JDBC .....	32
<b>2.18 DDL (Data Definition Language) .....</b>	<b>33</b>
2.23.1 Comenzi DDL.....	33
<b>2.19 DML (Data Manipulation Language) .....</b>	<b>34</b>
<b>2.20 Project Lombok.....</b>	<b>35</b>
<b>2.21 Controlul versiunilor .....</b>	<b>37</b>
2.23.2 Git .....	37
2.23.3 Diferența dintre Git și GitHub [27] .....	38
2.23.4 Proprietatea și costul Git vs GitHub .....	38
<b>2.22 IntelliJ IDEA .....</b>	<b>39</b>
2.23.1 Multi-platăformă.....	39
2.23.2 Limbaje JVM .....	39
2.23.3 Edițiile IntelliJ IDEA .....	39
2.23.4 Navigare și căutare .....	40
2.23.5 Structura fișierului .....	40
2.23.6 Completarea codului .....	41

2.23.7	Integrarea cu instrumentele de dezvoltare .....	41
2.23.8	Depanator.....	41
2.23.9	Instrumente pentru construcția proiectului .....	42
2.23.10	Controlul versiunii .....	42
2.23.11	Istorie locală.....	43
<b>2.23</b>	<b>Trello .....</b>	<b>43</b>
2.23.1	Panoul .....	43
2.23.2	Liste .....	44
2.23.3	Carduri .....	44
2.23.4	Meniul panoului.....	44
<b>2.24</b>	<b>Confluence .....</b>	<b>45</b>
 <b>CAPITOLUL III. ARHITECTURA APLICAȚIILOR .....</b>		<b>47</b>
<b>3.1</b>	<b>Aplicația pentru meniu.....</b>	<b>48</b>
<b>3.2</b>	<b>Aplicația pentru comenzi (orders).....</b>	<b>48</b>
<b>3.3</b>	<b>Aplicația pentru autentificare.....</b>	<b>49</b>
<b>3.4</b>	<b>Aplicația pentru plată (payment) .....</b>	<b>49</b>
<b>3.5</b>	<b>Aplicația de legătură.....</b>	<b>50</b>
<b>3.6</b>	<b>Baza de date.....</b>	<b>50</b>
<b>3.7</b>	<b>Detalii de implementare.....</b>	<b>51</b>
3.7.1	Interfața cu utilizatorul (aplicația frontend).....	52
3.7.1.1	Flow-ul aplicației .....	52
 <b>CAPITOLUL IV. CONCLUZII .....</b>		<b>55</b>
4.1	Dezvoltări ulterioare.....	55
 <b>CAPITOLUL V. BIBLIOGRAFIE .....</b>		<b>56</b>

# Capitolul I. Introducere

## 1.1 Abstract

Lucrarea are ca scop prezentarea detaliilor de design și implementare a unui sistem de digitalizare a procesului de comandă din restaurante.

Acest proiect este destinat comercianților din domeniul restaurantelor, dar și clienților acestora, el având scopul de a introduce o formă de optimizare a procesului de comandă respectiv preluare a comenzii în vederea scăderii timpului de așteptare al clienților dar și eficientizarea activității angajaților restaurantului respectiv , utilizând o metodă accesibilă oricui în ziua de azi : smartphone-ul.

Lucrarea este structurată pe capitole, încercând prin acestea să acopere informațiile de bază privind scopul, arhitectura , tehnologiile folosite , domeniul de aplicabilitate, etc.

Lucrarea este realizată în colaborare cu GSD Software & Technology, cu ajutorul căreia ar urma să devină o aplicație web disponibilă clienților iar un lanț de restaurante ar urma să o folosească în producție pentru a prelua și procesa comenzi.

## 1.2 Contextul proiectului

În ziua de azi, timpul este un lux pe care nu mulți și-l permit, și după părerea mea, nimeni nu vrea să-l irosească inutil așteptând zeci de minute după o persoană care trebuie să ia comenzile tuturor oamenilor din cadrul unui restaurant aglomerat.

Potrivit raportului Information Resources, Inc. (IRI) 2017, în Europa, una din cinci mese (18% din toate mesele) sunt consumate în afara locuinței.

Un alt studiu efectuat de biroul Executivului Yuan al guvernului taiwanez, în 2016, aproape 93% din Taiwanezi au mâncat în mod obișnuit afară, dintre care 33,74% au luat cina în afara locuințelor lor mai mult de patru ori pe săptămână, frecvența oamenilor care iau masa în restaurante crescând semnificativ față de anii precedenți.

Un alt studiu realizat de CBS Minnesota arată că 56% din americani servesc masa la restaurant , comandă sau își iau mâncare la pachet de cel puțin 2-3 ori pe săptămână.[1],

[2]

### 1.3 Ideea si scopul proiectului

Ideea proiectului a pornit de la nevoia constantă de a aștepta în restaurant după cineva care să preia comanda, mai apoi, în cazul în care cineva mai dorește să comande ceva, din nou trebuie să aștepte și în cele din urmă, când toată lumea a terminat de mâncat/băut, din nou trebuie să aștepte după nota de plată. Toate acestea se adună creând un timp de așteptare inutil din punctul meu de vedere și care ar putea fi semnificativ redus prin folosirea unei aplicații prin care să-ți fie preluată comanda fără ajutorul unei persoane din cadrul restaurantului.

Scopul acestui proiect este de a reduce timpul de așteptare din restaurante, acest lucru fiind benefic atât pentru clienți cât și pentru restaurant.

Un studiu realizat de Jelle De Vries , Debjit Roy și René B.M. De Koster a arătat că dacă clienții unui restaurant nu ar mai fi nevoiți să aștepte, profitul restaurantului ar crește cu până la 15%. [3]

În tabelul de mai jos este prezentată o simulare extrasă din studiul „Worth the wait? How restaurant waiting time influences customer behavior and revenue” [3] care ne arată că reducând la jumătate timpul de așteptare din restaurant, câștigurile restaurantului ar crește cu până la 4.7% iar în cazul în care timpul de așteptare ar fi redus la 0, câștigurile cu până la 14.5% mai mari.

Feb 2016–Jan 2017	Base case	Half the impact of waiting time	No impact of waiting time
Total # of visiting groups	[93,156–93,719]	[93,083–93,819]	[93,118–94,092]
Total # of returning groups	[22,563–23,119]	[22,487–23,226]	[22,527–23,506]
Total # of renegeing groups	[12,364–12,629]	[8357–8659]	0*
Total revenue generated	[\$5,033,312 - \$5,062,320]	[\$5,271,271 - \$5,307,804]	[\$5,749,495 - \$5,811,818]
CI avg. waiting time (min)	8.523 ± 0.098	4.688 ± 0.048	0
Revenue relative to base	–	+ 4.7%	+ 14.5%

Fig. 1 Efectul simulat al așteptării clienților asupra câștigurilor unui restaurant

### 1.4 Studiu state of the art

Proiectul se aseamănă ca și interfață pentru utilizator cu aplicațiile existente de comandat mâncare doar că scopul proiectului meu este acela de a folosi aplicația în cadrul unui restaurant și nu acasă.

Printre implementările asemănătoare găsite de mine se numără:

GloriaFood , o aplicație dezvoltată de Oracle care permite oricărui restaurant să primească comenzi online. [4]

Un articol publicat de Aman Arora povestește despre o idee asemănătoare în care clienții pot folosi un cod QR pentru a accesa meniul restaurantului și a comanda . [5]

## Capitolul II. TEHNOLOGII UTILIZATE

### 2.1 Java

Java a fost și încă este unul dintre cele mai populare limbaje de programare din lume.

În luna iunie a anului 1991, James Gosling, Mike Sheridan și Patrick Naughton pun bazele actualului limbaj de programare Java. Inițial Java fusese conceput pentru televiziunea interactivă dar era mult prea avansat pentru industria televiziunii digitale de la vremea aceea.

Limbajul s-a numit inițial Oak(Stejar) după un stejar care se afla în afara biroului lui Gosling. Mai apoi proiectul a luat numele de Green(Verde) ca mai apoi să fie numit Java după un tip de cafea din Indonezia.

Java este un limbaj de programare dar în același timp este și o platforma.

Este un limbaj de programare orientată pe obiecte dar totuși, nu este considerat un limbaj pur orientat pe obiecte deoarece ne oferă suport și pentru tipuri de date primitive (cum ar fi int, char, etc.).

O platformă Java este un mediu particular în care pot rula aplicațiile scrise în limbaj Java.

Java ne oferă 4 platforme:

Java Platform, Standard Edition(Java SE)

Java Platform, Enterprise Edition(Java EE)

Java Platform, Micro Edition(Java ME)

JavaFX

Toate platformele Java constau dintr-o mașină virtuală Java (JVM) și o interfață de programare a aplicațiilor (API). [6]

#### 2.1.1 Java SE

Când majoritatea oamenilor se gândesc la limbajul de programare Java, ei se gândesc de fapt la API-ul Java SE. Această interfață oferă funcționalitatea de bază a limbajului de programare Java. Definește totul de la tipurile de bază până la obiecte și clase de nivel înalt folosite pentru rețele, securitate, acces la baze de date, dezvoltarea interfețelor grafice sau analiza fișierelor XML.



În plus față de acest API, platforma Java SE mai constă și dintr-o mașină virtuală, instrumente pentru dezvoltare, tehnologii pentru implementare și alte librării des folosite în aplicațiile Java.

### **2.1.2 Java EE**

Platforma Java EE este construită peste platforma Java SE. Java EE ne oferă un mediu pentru dezvoltarea aplicațiilor web securizate la scară largă, scalabile și pe mai multe niveluri.

### **2.1.3 Java ME**

Platforma Java ME ne oferă un API și o mașină virtuală care ocupă mai puțin spațiu, fiind folosită pentru a executa aplicații Java pe dispozitive mici, cum ar fi telefoanele mobile.

API-ul Java ME este un subset al API-ului Java SE la care se adaugă librării specifice pentru dezvoltarea aplicațiilor pentru dispozitive de mici dimensiuni. Aplicațiile Java ME sunt de obicei clienți ai serviciilor Java EE.

### **2.1.4 Java FX**

Platforma JavaFX ne oferă posibilitatea creării unor aplicații web bogate(RIA - Rich Internet Application) folosind un API cu un consum redus de resurse.

O aplicație web bogată oferă caracteristicile unei aplicații desktop

În faza de compilare, codul Java este transformat într-un cod intermediar denumit și bytecode sau p-code (cod portabil). Bytecode-ul este un cod de nivel scăzut care poate fi interpretat de către un translator de software, în cazul Java fiind vorba despre mașina virtuală Java (JVM - Java Virtual Machine). Această mașină virtuală este specifică pentru fiecare platformă în parte și este integrată în kit-ul de instalare Java (JDK - Java Development Kit). Acest JVM transforma bytecode-ul în cod mașină care este înțeles și executat de procesor.

Pentru ca o aplicație Java să funcționeze în mod optim, JVM-ul împarte memoria în stivă (stack) și heap.

Stiva de memorie este creată la pornirea unui proces, dimensiunea acesteia este fixă și se stabilește, în faza de compilare, în funcție de variabilele declarate.

Dimensiunea stivei nu poate fi modificata de către un proces început.În stivă se regăsesc zone alocate fiecărei metode din program.Zonele dedicate metodelor există în stivă doar pe durata ciclului de viață al metodei respective( din momentul apelării metodei

până în momentul terminării metodei).Accesul la stivă se face în ordinea ultimul intrat, primul ieșit (LIFO).

În cazul în care stiva este plină, primim eroarea `Java.lang.StackOverflowError`.

Heap-ul este un spațiu folosit pentru alocarea dinamică de memorie pentru obiectele Java și pentru clasele specifice JRE (Java Runtime Environment).

## 2.2 Swagger

Swagger este un instrument folosit pentru a documenta aplicațiile REST(Representational state transfer) în formatul de reprezentare JSON.[4]

În tărâmul software al zilelor noastre, nu se găsesc sisteme care rulează online fără un API(Application Programming Interface) expus. Ne-am mutat de la sisteme monolitice la microservicii și întregul design de microservicii se bazează pe API-uri REST(Representational State Transfer).

Specificațiile unei aplicații de obicei includ informații cum ar fi: operații suportate,parametrii, date de ieșire, cerințe pentru autorizare, nodurile terminale ale aplicației(endpoints) și licențele necesare. Swagger poate genera aceste informații în mod automat pe baza codului sursă și interogând aplicația pentru a ne întoarce o documentație bazată pe adnotări.

Swagger poate fi folosit în două abordări diferite:

- Top-down sau design-first, situație în care putem folosi swagger editor pentru a proiecta aplicația noastră fără a fi nevoie de cod scris.

- Bottom-up sau code-first , situație în care swagger ne generează automat documentația aplicației pe baza codului scris.

Swagger ne oferă 3 instrumente pentru construcția, design-ul, documentarea și testarea aplicațiilor:

- Swagger Editor [7]

- Swagger UI [8]

- Swagger Codegen

Swagger Editor este folosit pentru a modifica design-ul aplicației folosind structuri de tip YAML sau JSON.Accesul la editor se face din pagina API-ului, prin apăsarea pictogramei reprezentând logo-ul oficial swagger(fig 1):

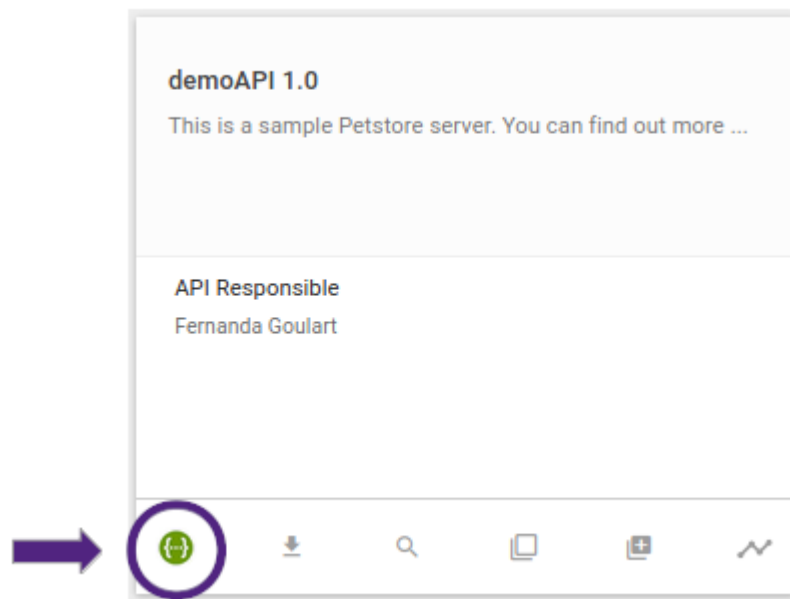


Fig. 2 Pictograma pentru accesul la Swagger Editor

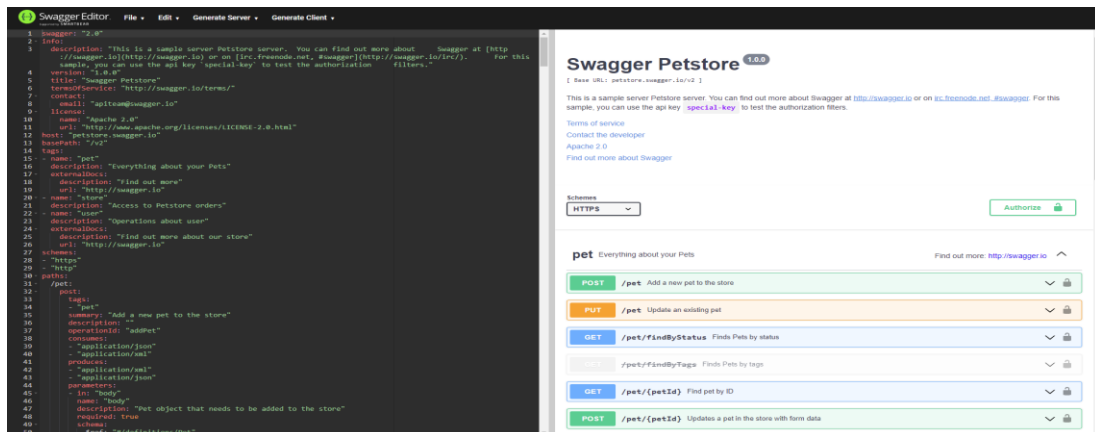


Fig. 3 Interfața Swagger Editor [7]

Swagger UI permite oricui – fie unei echipe de dezvoltare sau unor clienți obișnuiți – să vizualizeze și să interacționeze cu resursele aplicației fără a fi nevoie de o logică implementată. Documentația este generată automat pe baza standardului OpenAPI, partea vizuală ușurând implementarea pe partea de backend și accesarea pe partea de client.

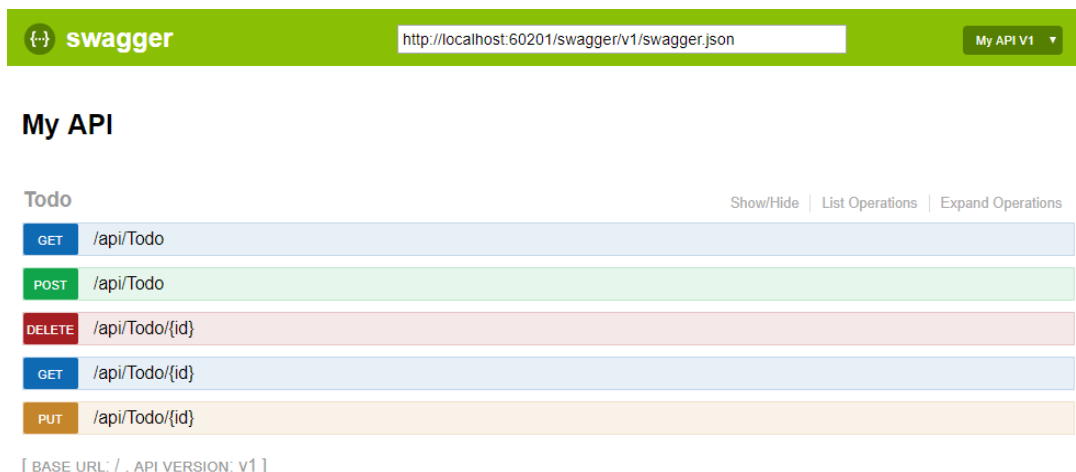


Fig. 4 Interfața Swagger UI

## 2.3 Spring

Spring este cel mai popular framework pentru dezvoltarea de aplicații Java.

Un framework reprezintă o structură de la care poți porni pentru a-ți dezvolta aplicația. Framework-urile sunt de obicei asociate cu limbaje de programare specifice și sunt

create pentru a îndeplini diferite tipuri de sarcini. Ele sunt create și testate de ingineri și dezvoltatori software, deci știm că sunt o fundație solidă. [9]

Framework-ul Spring este o platformă Java al cărei cod sursă este disponibil gratuit pe web. Codul sursă a fost scris inițial de către Rod Johnson și a fost lansat pentru prima dată în iunie 2003 sub licența Apache 2.0.

Spring este foarte compact în privința spațiului de stocare și al transparenței. Versiunea de bază a framework-ului Spring ocupă aproximativ 2MB. [4]

Beneficii ale folosirii framework-ului Spring:

- Spring este modularizat astfel încât, chiar dacă numărul de clase și pachete este substanțial, nu trebuie să îți faci griji decât de cele de care ai nevoie, iar pe restul le poți ignora.

- Testarea unei aplicații care a fost scrisă folosind Spring este simplă deoarece codul dependent de mediul de programare este mutat în acest framework. În plus, folosind POJO-uri, devine foarte ușor să folosim injectarea de dependențe pentru a injecta date de test.
- Spring oferă o interfață consistentă pentru administrarea tranzacțiilor care poate fi redusă până la o tranzacție locală (de exemplu folosirea unei baze de date) sau mărită până la tranzacții globale (de exemplu folosirea JTA - Java Transaction Api).

### 2.3.1 Injectarea dependențelor

Tehnologia prin care se identifică Spring cel mai des este injectarea dependențelor, care este doar unul dintre exemplele concrete ale inversiunii de control.

Injectarea dependențelor este un șablon (pattern) de proiectare în care inversiunea controlului se referă la dependențele unui obiect.

Conectarea obiectelor cu alte obiecte, sau “injectarea” obiectelor în alte obiecte este făcută de un asamblor decât de obiectele în sine.

În continuare avem un exemplu de creare a unei dependențe de obiect în mod obișnuit:

```
public class Store {  
    private Item item;  
  
    public Store() {  
        item = new ItemImpl();  
    }  
}
```

Fig. 5 Crearea unei dependențe simple

În exemplul de mai sus, trebuie să instanțiem o implementare a interfeței Item în clasa Store. Folosind injectarea dependențelor, putem rescrie exemplul de mai sus fără a mai specifica implementarea obiectului Item pe care o dorim:

```
public class Store {  
    private Item item;  
    public Store(Item item) {  
        this.item = item;  
    }  
}
```

Fig. 6 Exemplu pentru injectarea unei dependențe

### 2.3.2 Inversiunea de control (IoC – Inversion of Control)

Este un principiu în programare prin care controlul unor obiecte sau a unor porțiuni din program este transferat către un container sau framework.

În contrast cu programarea tradițională, în care codul nostru face apeluri către o librărie, IoC permite unui framework să preia controlul cursului unui program și să facă apeluri la codul nostru.

Avantajele acestei arhitecturi sunt:

- Decuplarea execuției unei sarcini de implementarea acesteia.
- Schimbul între diferite implementări se face mult mai simplu.
- Programul este mult mai modularizat.

### 2.3.3 Programarea orientată pe aspecte(AOP – Aspect Oriented Programming)

Una dintre componentele cheie ale Spring este framework-ul pentru programarea orientată pe aspecte. Metodele care cuprind mai multe puncte ale unei aplicații poartă numele de cross-cutting concerns(preocupări transversale/încrucișate) și acestea sunt separate conceptual de logica aplicației.

Elementul cheie al modularității în OOP este clasa, în timp ce în AOP elementul cheie este aspectul.

Aspectul este o clasă care implementează preocupări care se regăsesc în mai multe clase.

Spring se împarte în mai multe framework-uri, printre care:

- Spring Framework
- Spring Boot
- Spring Data
- Spring Security

## **2.4 Spring Boot**

Spring Boot este un framework open-source (codul sursă este disponibil gratuit) bazat pe limbajul de programare Java, folosit pentru a crea microservicii. Este dezvoltat de “Pivotal Team” și este folosit pentru a dezvolta aplicații Spring de sine stătătoare. Un microserviciu este o arhitectură care permite dezvoltatorilor să construiască și să lanseze servicii independente. [10]

### **2.4.1 Avantaje ale microserviciilor**

Microserviciile oferă următoarele avantaje dezvoltatorilor:

- Lansarea mai ușoară a aplicațiilor
- Scalabilitate simplă
- Compatibilitate cu containere
- Configurare minimă
- Timp mai scurt de producție

Scopul Spring Boot:

- Evitarea fișierelor de configurare xml foarte complexe
- Producerea unei aplicații Spring funcțională într-un mod simplu
- Reducerea timpului de dezvoltare și rularea aplicației în mod independent

Cum funcționează?

Spring Boot configurează în mod automat aplicația bazându-se pe dependențele adăugate în proiect, dacă este folosită adnotarea `@EnableAutoConfiguration`.

Punctul de plecare al unei aplicații Spring Boot este clasa care conține adnotarea `@SpringBootApplication` și metoda `main`.

Spring Boot scanează automat toate componentele incluse în proiect folosind adnotarea `@ComponentScan`.

### 2.4.2 Spring Data

Framework-ul Spring Data este proiectul părinte care conține mai multe sub-framework-uri. Toate aceste sub-framework-uri se ocupă de accesul la date din baze de date specifice. Obiectivul acestui framework este de a oferi un model familiar și consistent de acces la baze de date bazat pe Spring.

### 2.4.3 Spring Security

Spring Security este un framework care oferă diverse caracteristici de securitate cum ar fi: autentificare și autorizare care ajută la crearea aplicațiilor Java securizate.

Este un sub-proiect al Spring framework care a fost început în 2003 de către Ben Alex. Mai târziu, în 2004, a fost lansat sub licența Apache ca “Spring Security 2.0.0”.

Acest framework țintește două arii majore ale unei aplicații: autentificarea și autorizarea.

Autentificarea este procesul de identificare a utilizatorilor care solicită acces.

Autorizarea este procesul prin care i se permite unui anumit tip de utilizator accesul la anumite acțiuni din cadrul aplicației.

Putem aplica autorizarea pentru request-uri web, metode, sau pentru accesul la anumite domenii.

## 2.5 Angular

Angular este un framework bazat pe JavaScript folosit pentru crearea aplicațiilor interactive și dinamice în HTML. Framework-ul Angular a fost creat de echipa Angular din cadrul Google și de o comunitate de indivizi și corporații și este gratuit și open-source.

Angular este o rescriere completă de la aceeași echipă care a creat și AngularJS.

AngularJS a fost un framework de front-end open-source pentru dezvoltarea aplicațiilor web formate dintr-o singură pagină (SPA- Single-Page Applications) care, din data de 1 Ianuarie 2022 nu mai primește update-uri din partea Google pentru securitate, compatibilitatea browserelor și probleme legate de jQuery.

O aplicație formată dintr-o singură pagină web interacționează cu utilizatorul în mod dinamic, rescriind conținutul paginii web curente cu date noi de la server, în defavoarea metodei standard a browserelor web de a încărca pagini web noi. Scopul este de a face mult mai rapidă tranziția între “pagini”. [11]



### 2.5.1 Diferențe între Angular și AngularJS:

- Angular nu mai folosește concepte precum durată de viață (scope) sau controllere. În locul acestora, Angular folosește o ierarhie de componente ca și arhitectură caracteristică.
- Angular folosește o altfel de sintaxă, concentrându-se pe “[ ]” pentru legarea proprietăților(property binding) și pe “( )” pentru legarea evenimentelor(event binding).
- Modularitate - multă funcționalitate de bază a fost mutată în module.
- Angular recomandă folosirea limbajului de programare TypeScript, dezvoltat și menținut de Microsoft, care introduce următoarele caracteristici:
  - Static typing (tipul de date al variabilelor este cunoscut în faza de compilare, spre deosebire de dynamic typing unde tipul de date al variabilelor este cunoscut doar în faza de runtime).
  - adnotări
- TypeScript este un superset al ECMAScript 6 și este compatibil cu ECMAScript 5 (JavaScript).
- Încărcare dinamică
- Compilarea în mod asincron a fișierelor șablon.
- Suport pentru “Angular Universal”, care rulează aplicații Angular pe servere.

Noua versiune rescrisă a AngularJS a fost denumită inițial “Angular 2”, dar acest lucru a creat confuzie printre dezvoltatori. Pentru a clarifica acest aspect, echipa a standardizat folosirea termenilor de “AngularJS” pentru versiuni 1.X și “Angular” pentru versiuni 2 și mai noi.

## 2.6 JavaScript(JS)

JavaScript este un limbaj de programare ușor în sensul în care oferă funcționalitate de programare, dar nu poate efectua operații mai puternice, spre deosebire de Java sau C++. [12]

Spre deosebire de Java, codul scris în JavaScript este compilat în faza de runtime (just-in-time compilation) ceea ce înseamnă că este permisă leagarea târzie a tipurilor de date(late-bound data types) și se impune garanția securității.

JavaScript este un limbaj de programare scriptic (care automatizează sarcinile care altfel ar fi trebuit executate manual) care permite dezvoltatorilor să implementeze caracteristici complexe pe pagini web (de ex. hărți interactive, animații 2D/3D etc.).

JavaScript face parte din cele 3 tehnologii web standard: HTML, CSS, JavaScript. Prin funcționalitatea de bază din JavaScript, poți realiza următoarele și nu numai:

- stocarea valorilor folosite în variabile.
- operații pe părți dintr-un text (cunoscut ca “String” în programare)
- executarea unei părți din cod ca și răspuns la evenimentele care se petrec pe o anumită pagină web.

Ceea ce este și mai incitant este funcționalitatea construită peste limbajul JavaScript. Așa numitele Interfețe pentru programarea aplicațiilor (API- Application Programming Interface) care ne oferă “super-puteri” pe care le putem folosi în codul JavaScript.

Aceste API-uri sunt bucăți de cod deja scrise care permit dezvoltatorilor să implementeze programe care altfel ar fi foarte greu sau chiar imposibil de implementat. Un API s-ar putea asemana cu un kit pentru construcția unei piese de mobilier, în care ai toate piesele și uneltele necesare respectivei construcții.

Aceste API-uri se împart în general în 2 categorii:

Interfețe pentru Browser, care sunt deja integrate în browser-ul dumneavoastră web și ne permit să facem niște operații complexe cum ar fi:

- Interfața DOM (Document Object Model) ne permite să manipulăm codul HTML și CSS dintr-o pagină web, aplicând în mod dinamic modificările.
- Interfața pentru geolocație preia informații geografice. În acest fel Google Maps poate să ne găsească locația pentru a o afișa pe hartă.

Interfețe externe, care nu sunt integrate în browser și pentru a fi folosite, trebuie adus codul și informațiile necesare de pe Web. Exemple de interfețe externe ar fi:

- Interfața Twitter, care ne permite să afișăm tweeturile noastre direct pe site-ul pe care îl construim.
- Interfața Google Maps și OpenStreetMap, care ne permit să afișăm hărți particularizate pe site-ul nostru.

### **2.6.1 Securitatea unui browser**

Fiecare filă a unui browser are propriul mediu pentru execuția codului. Ceea ce înseamnă că, în cele mai multe cazuri, codul dintr-o filă este executat independent și nu poate fi afectat de codul din alte file, sau de pe alt site.

Aceasta este o bună măsură de siguranță pentru că dacă nu s-ar aplica, persoanele rău intenționate ar putea scrie cod pentru a fura informații de pe alte site-uri.

### **2.6.2 Ordinea de execuție a instrucțiunilor JavaScript**

Când browserul întâlnește un bloc de cod JavaScript, în general îl execută în ordine, de sus în jos. Asta înseamnă că trebuie să fim atenți în ce ordine scriem instrucțiunile.

### **2.6.3 Cod interpretat versus cod compilat**

În cadrul limbajelor de programare cu cod interpretat, codul este executat de sus până jos iar rezultatul execuției este imediat returnat. Codul nu trebuie transformat în alt format înainte de a fi executat de către un browser. Codul rezultat este într-un format text, ușor de înțeles de către programatori.

Pe de altă parte, în cadrul limbajelor de programare cu cod compilat, codul este transformat în alt format înainte de a fi executat de către calculator. De exemplu, codul C/C++ este compilat în cod mașină care mai apoi este executat de către calculator. Programul de executat este în format binar, care a fost generat din programul sursă original.

JavaScript este un limbaj de programare cu cod interpretat. Browserul web primește codul JavaScript în forma sa originală. Din punct de vedere tehnic, majoritatea interpretoarelor de cod JavaScript actuale folosesc o tehnică numită “just-in-time compiling” pentru a îmbunătăți performanța prin care codul JavaScript este compilat într-un format binar, mult mai rapid, în timp ce este rulat scriptul, pentru a face execuția cât mai rapidă.

Cu toate acestea, JavaScript încă este considerat un limbaj de programare cu cod interpretat, din moment ce compilarea este tratată în faza de runtime și nu înainte.

### **2.6.4 Cod pe parte de client și cod pe parte de server**

Codul de pe partea de client este executat pe calculatorul utilizatorului. Când o pagină web este vizualizată, codul de pe partea de client este descărcat, executat și afișat în browser.

Codul de pe partea de server, pe de altă parte, este executat pe un server, apoi rezultatele sunt descărcate și afișate în browser.

Exemple de limbaje de programare web pe parte de server ar fi: PHP, Python, Ruby, ASP.NET și chiar și JavaScript.

### **2.6.5 Cod dinamic versus cod static**

Cuvântul dinamic este folosit atât pe partea de client, cât și pe partea de server și se referă la abilitatea de a actualiza conținutul unei pagini web pentru a afișa lucruri diferite în circumstanțe diferite, generând conținut nou în funcție de cerere. Codul de pe partea de server generează conținut nou pe server (de exemplu: extragerea de informații din baza de date), în timp ce, pe partea de client, JavaScript generează în mod dinamic conținut nou în browserul clientului (de exemplu: crearea unei noi tabel HTML, popularea tabelului cu datele cerute de la server, apoi afișarea tabelului într-o pagină web afișată utilizatorului).

## **2.7 TypeScript**

TypeScript este un limbaj de programare dezvoltat și menținut de către Microsoft. Este un superset al JavaScript care adaugă funcționalitatea de verificare a tipurilor de date în faza de compilare (static typing). TypeScript a fost proiectat pentru construcția aplicațiilor mari și este transcompilat (tradus dintr-un limbaj de programare în altul) în limbaj JavaScript, ceea ce înseamnă că orice program scris în JavaScript este un program TypeScript valid. [13]

### **2.7.1 Originile, credibilitatea și fiabilitatea TypeScript**

Ca și dezvoltator, ar trebui să fii mereu atent când îți alegi tehnologiile pe care le vei folosi pentru că se pot schimba foarte rapid - proiectele sunt abandonate, tehnologiile devin depreciate și limbajele de programare devin învechite. Când vine vorba de TypeScript, nu ar trebui să-ți pui problemele acestea deoarece TypeScript este susținut de către Microsoft (creatorul TypeScript) și Google (Angular folosește TypeScript). În plus, Microsoft a implementat suport pentru TypeScript în Visual Studio și în Visual Studio Code.

### 2.7.2 Tipurile de date

Dacă scrieți o aplicație mai mare decât un simplu “Hello, World!”, la un moment dat, inevitabil, veți uita tipul de date al celui de-al 3-lea parametru al funcției X. De fapt, va fi greu să vă amintiți și dacă funcția are sau nu 3 parametri. Multe medii de dezvoltare moderne încearcă să rezolve această problemă analizând codul JavaScript dar rezultatele sunt dezamăgitoare. TypeScript rezolvă această problemă, fiind încă de la început prietenos cu mediul de dezvoltare (IDE- integrated development environment). Asta înseamnă că în orice moment, IDE-ul poate identifica sursa unei anumite funcții pe care doriți să o folosiți și poate afișa documentația, parametrii și tipurile acestora.

### 2.7.3 Interfețe

Pentru început, făcând o comparație, în JavaScript nu există interfețe. În JavaScript ne folosim de tehnica “duck typing”, care practic presupune că noi sperăm ca obiectul A să conțină proprietățile X, Y, Z. Convențiile sunt bune, dar când lucrăm la o aplicație de dimensiuni mari, nu este o idee prea bună să ne bazăm pe memorie pentru tot. Situația se înrăutățește când în echipă sunt mai mulți dezvoltatori. Ar trebui puse în aplicare niște contracte și aici vin în ajutor interfețele din TypeScript.

## 2.8 HTML (Hypertext Markup Language)

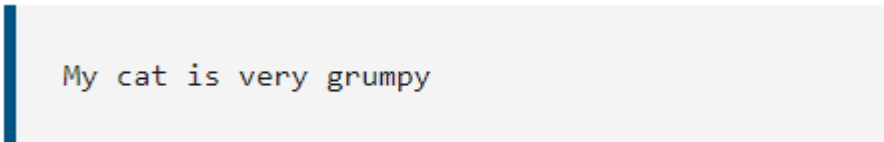
HTML este limbajul de marcare standard pentru documentele destinate pentru a fi afișate în browsere web. Acesta poate fi folosit în combinație cu tehnologii precum CSS (Cascading Style Sheets) și limbaje de programare scriptice precum JavaScript.

Limbajul HTML a fost pentru prima oară menționat de către fizicianul Tim Berners-Lee în anul 1990. [14]

HTML este codul folosit pentru a structura o pagină web și conținutul acesteia. De exemplu, conținutul poate fi structurat într-un set de paragrafe, o listă de puncte cheie, imagini sau tabele.

HTML este format dintr-o serie de elemente pe care le folosim pentru a delimita sau a îngrădi diferite părți din conținut pentru a fi afișat într-un anumit fel sau pentru a acționa într-un anumit fel.

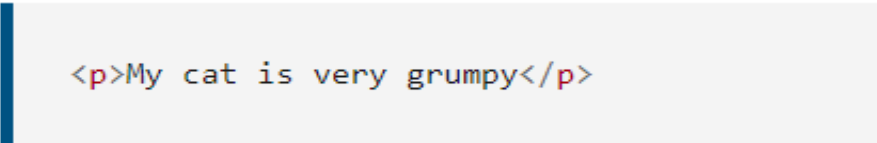
Etichetele din HTML pot face un cuvânt sau o imagine să ne trimită în altă parte, pot face fontul mai mare sau mai mic, ș.a.m.d. De exemplu, avem următoarea linie de conținut:



```
My cat is very grumpy
```

Fig. 7 Exemplu de paragraf fără etichete html

Dacă vrem ca această linie să fie de sine stătătoare, putem să o încadrăm în etichete de paragraf:



```
<p>My cat is very grumpy</p>
```

Fig. 8 Exemplu de paragraf folosind etichetele html pentru paragraf

### 2.8.1 Anatomia unui element HTML

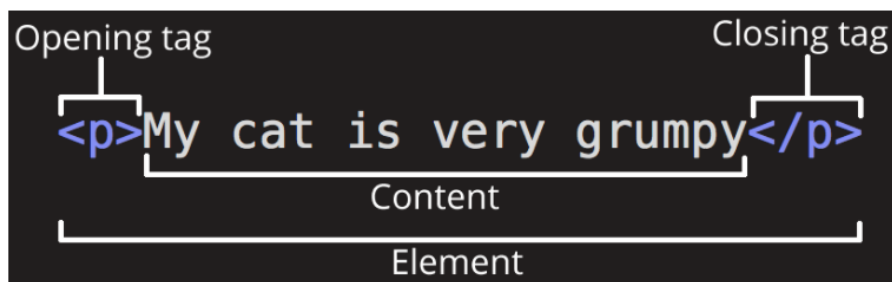


Fig. 9 Părțile componente ale unui element HTML

Principalele părți ale elementului nostru sunt:

1. Eticheta de deschidere: este formată din numele elementului (în cazul nostru, p) între paranteze unghiulare. Această etichetă ne arată unde începe elementul, în cazul nostru unde începe paragraful.
2. Eticheta de închidere: este la fel ca eticheta de deschidere, excepție face bara oblică care se pune înaintea numelui elementului. Această etichetă ne indică unde se încheie elementul. Omiterea etichetei de închidere este una dintre frecventele greșeli ale începătorilor și poate duce la rezultate neprevăzute.
3. Conținutul: reprezintă conținutul elementului, care în cazul nostru este doar text.

4. Elementul:Eticheta de deschidere împreună cu eticheta de închidere și cu conținutul formează elementul.

## 2.8.2 Anatomia unui document HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    
  </body>
</html>
```

Fig. 10 Exemplu de document HTML

- `<!DOCTYPE html>` : Toate documentele HTML trebuie să înceapă cu declarația `<!DOCTYPE>`. Această declarație nu este o etichetă HTML, este o informație furnizată browserului pentru a ști la ce tip de document să se aștepte.
- `<html></html>` : Elementul `<html>` cuprinde tot conținutul unei pagini web și este cunoscut drept elementul rădăcină.
- `<head></head>` : Acest element se comportă ca un recipient în care se stochează toate elementele pe care vrem să le includem în pagină dar nu fac parte din conținutul afișat. Printre aceste elemente putem menționa cuvinte cheie sau descrierea unei pagini care vrem să apară în rezultatele căutării.
- `<meta charset="utf-8">` : Acest element stabilește setul caracterelor pe care documentul ar trebui să le folosească, în cazul nostru "UTF-8", care include majoritatea caracterelor dintr-o vastă majoritate de limbi scrise.
- `<title></title>` : Acest element setează titlul paginii web, care va apărea în fila browserului în care este încărcată pagina. Elementul `<title>` mai este folosit și la descrierea paginii când aceasta este adăugată la favorite.
- `<body></body>` : Acest element conține tot conținutul pe care dorim să-l afișăm utilizatorilor care accesează pagina.

### 2.8.3 Verificarea corectitudinii unui document HTML

În mod normal codul HTML nu este verificat de erori în timp real. Browserul deschide orice document HTML chiar dacă acesta conține erori iar acest lucru poate produce rezultate neașteptate. Pentru a preveni acest lucru, putem verifica corectitudinea unui document HTML accesând site-ul web [www.validator.w3.org](http://www.validator.w3.org) unde putem da adresa unui site, putem încărca un fișier HTML sau putem verifica direct codul HTML.

## 2.9 CSS (Cascading Style Sheets)

CSS este un limbaj folosit pentru stilizarea unui document scris într-un limbaj de marcare cum este HTML. De exemplu, putem schimba fontul, culoarea, dimensiunea și spațierea conținutului, putem să-l împărțim în coloane, să adăugăm animații și alte caracteristici decorative. [15]

Prezentarea unui document unui utilizator presupune convertirea documentului într-o formă accesibilă. Browsersle sunt concepute să prezinte documentele în mod vizual, pe un monitor, proiector sau unei imprimante.

### 2.9.1 Sintaxa CSS

CSS este un limbaj bazat pe reguli. Se definesc reguli în care specificăm stiluri și elementele paginii web cărora trebuie atribuite. De exemplu : “Aș vrea ca rubrica principală a paginii mele să apară ca un text mare și roșu.”

Următoarea secvență de cod definește o simplă regulă CSS care îndeplinește cerința de mai sus:

```
h1 {  
    color: red;  
    font-size: 5em;  
}
```

Fig. 11 Exemplu de regulă CSS



Regula începe cu un selector. Acesta alege elementul HTML căruia i se va atribui stilul definit. În cazul nostru, stilizăm rubricile de nivel 1 (<h1>).

Mai apoi, avem un set de acolade. Înăuntrul acoladelor vor fi una sau mai multe declarații, care sunt de forma perechi proprietate-valoare. Fiecare pereche specifică o proprietate a unui element selectat și o valoare pe care vrem să o atribuim acelei proprietăți.

Proprietatea este separată de valoare prin semnul “:”. Proprietățile CSS au diferite valori admisibile, care depind de proprietatea specificată.

În exemplul nostru avem proprietatea “color” care poate lua diferite valori de culoare (de exemplu : antiquewhite, blueviolet, greenyellow). Pe lângă “color” mai avem și proprietatea “font-size” care poate lua diferite valori de unități de măsură (de exemplu: cm, mm, Q-quarter millimeters, in-inches, pc-picas, pt-points, px-pixels etc.).

O pagină CSS cel mai probabil va conține mai multe astfel de reguli, scrise una după alta.

```
h1 {  
    color: red;  
    font-size: 5em;  
}  
  
p {  
    color: black;  
}
```

Fig. 12 Exemplu de folosire a mai multor reguli CSS

## 2.9.2 Module CSS

Deoarece sunt foarte multe elemente care pot fi stilizate folosind CSS, limbajul a fost împărțit în module.

Un astfel de modul îl reprezintă “Backgrounds and Borders” care conține proprietățile “background-color” și “border-color”

Cu modulele CSS este garantat că toate stilurile aparținând unei componente:

1. Se regăsesc într-un singur loc.
2. Sunt aplicate doar acelei componente.

Această abordare este menită să rezolve problema domeniului global de aplicare al CSS.

## **2.10 Angular Material**

Angular Material este o librărie dezvoltată de Google în anul 2014. Este dezvoltată în special pentru dezvoltatorii AngularJS și ajută la proiectarea aplicațiilor într-un mod structurat.[16]

Componentele librăriei ajută la construirea paginilor și aplicațiilor web atractive, consistente și funcționale.

Google a dezvoltat Angular Material în anul 2014. În momentul acela era destinat dezvoltatorilor de AngularJS pentru a face aplicațiile mai atractive și mai performante ca și timp. În 2016, Google a rescris complet codul și l-a denumit Angular Material.

### **2.10.1 De ce Angular Material Design**

Material Design își propune să unifice experiențele utilizatorilor de pe web, mobil și de pe tablete. Angular Material Design este construit cu ajutorul AngularJS, Polymer (librărie JavaScript pentru crearea componentelor web) și Ionic (un framework UI construit pentru crearea aplicațiilor mobile hibride bazat pe HTML5).

## **2.11 Redux**

Redux este o librărie JavaScript open-source pentru gestionarea și centralizarea stării unei aplicații. Librăria este dezvoltată de Facebook începând cu anul 2015.

Această librărie este bazată pe modelul Flux. Diferența majoră între Flux și Redux constă în modul în care tratează acțiunile; În cazul Flux avem de obicei mai multe depozite și un dispecer, pe când în cazul Redux există un singur depozit ceea ce înseamnă că nu mai avem nevoie de dispecer.

Pentru a folosi Redux în framework-ul Angular, putem folosi librăria NgRx. Cu NgRx putem stoca toate evenimentele (datele) din aplicația Angular într-un singur loc (depozit). Când vrem să folosim datele stocate, trebuie să le luăm din depozit folosind librăria RxJs. RxJs (Reactive Extensions for JavaScript) este o librărie bazată pe modelul "Observable", folosit în Angular pentru procesarea operațiilor asincrone.[17]

### **2.11.1 De ce am folosi Redux în aplicația noastră Angular**

Am putea folosi un serviciu pentru a transfera datele între componente (trebuie totuși să fim atenți să ne dezabonăm de la observable de fiecare dată, altfel riscăm ca acel observable să ruleze în fundal, ceea ce consumă resurse) sau am putea folosi

decoratorii Input/Output(aici trebuie să ne asigurăm că există o relație părinte-copil între componente).

“Decorator marchează o clasă ca componentă Angular și ne permite să setăm metadata de configurație care determină modul în care componenta ar trebui procesată, instanțiată și utilizată în timpul rulării.”

Am mai putea folosi și decoratorul ViewChild pentru componente imbricate.Dar în cazul unui proiect de dimensiuni mai mari, aceste soluții vor crește complexitatea proiectului.

Dacă avem un număr mare de componente riscăm să pierdem controlul asupra fluxului de date dintr-o componentă.(de unde au venit aceste date și care este destinația lor?)

Acesta este motivul pentru care folosim Redux în Angular: depozitul și fluxul de date unidirecțional reduc complexitatea aplicației. Fluxul datelor este mult mai clar și mai ușor de înțeles de către membrii echipei.

## 2.12 Bootstrap

Bootstrap este un framework gratuit și open-source folosit pentru crearea site-urilor și a aplicațiilor web.Framework-ul este construit pe baza limbajelor HTML, CSS și JavaScript pentru a facilita dezvoltarea aplicațiilor web receptive.[18]

Design-ul receptiv permite paginilor sau aplicațiilor web să detecteze dimensiunile ecranului utilizatorului și orientarea (în cazul dispozitivelor mobile) pentru a adapta conținutul în concordanță.

Bootstrap include componente pentru interfața cu utilizatorul, aspecte, și unele JavaScript. Acest software este disponibil precompilat sau ca și cod sursă.

Mark Otto și Jacob Thornton au dezvoltat Bootstrap în timpul în care lucrau la Twitter cu scopul de a îmbunătăți consistența uneltelor folosite pe site dar și reducerea mentenanței.

Software-ul a fost cunoscut ca “Twitter Blueprint” și câteodată mai este referit ca “Twitter Bootstrap”.

În lumea calculatoarelor , cuvântul bootstrap face referire la boot: a încărca un program (de obicei este vorba de un sistem de operare) într-un calculator folosind un program inițial mai mic .

În viața de zi cu zi, un “bootstrap” reprezintă o curea atașată la spatele care ne permite să ne încălțăm mult mai ușor.

Bootstrap vine cu foarte multe componente cum ar fi: butoane, formulare, liste, tabele, teme, bare de navigare etc. care pot fi folosite așa cum sunt sau pot fi modificate în funcție de cerințele aplicației.

## **2.13 Postman**

Postman este un client API care face mult mai ușor pentru dezvoltatori procesele de creare, distribuire, testare și documentare a API-urilor. Este un client HTTP care testează cererile (requests) HTTP, folosind o interfață grafică prin care obținem diferite tipuri de răspunsuri care trebuie validate ulterior. [19]

Postman oferă mai multe metode pentru interacțiunea cu endpointurile. Următoarele sunt unele dintre cele mai folosite:

- GET: obținerea de informații
- POST: adăugarea de informații
- PUT: înlocuirea informațiilor
- PATCH: modificarea anumitor informații
- DELETE: ștergerea informațiilor

### **2.13.1 Coduri de răspuns**

Când testăm API-uri cu Postman, de obicei obținem diferite coduri de răspuns. Câteva dintre aceste coduri includ:

- seria de coduri 100 -> răspunsuri temporare, de exemplu “102 Processing”
- seria de coduri 200 -> răspunsuri în care clientul acceptă cererea și server-ul o procesează cu succes, de exemplu “200 Ok”.
- seria de coduri 300 -> răspunsuri legate de redirectarea URL, de exemplu “301 Moved Permanently”.
- seria de coduri 400 -> răspunsuri de eroare pe partea de client, de exemplu “400 Bad Request”.
- seria de coduri 500 -> răspunsuri de eroare pe partea de server, de exemplu “500 Internal Server Error”.

### **2.13.2 Colecții**

Postman ne oferă posibilitatea de a grupa diferite cereri. Această facilitate este cunoscută drept “colecție” și ne ajută la organizarea testelor.

Aceste colecții sunt directoare unde sunt stocate cererile și pot fi structurate în orice fel preferă echipa. Este posibil ca acestea să fie importate respectiv exportate.

### 2.13.3 Medii de lucru

Postman ne permite să creăm diferite medii de lucru prin generarea/folosirea de variabile; de exemplu, o variabilă URL care are ca scop folosirea în diferite medii de testare, permițându-ne să executăm teste în diferite medii folosind cereri existente.

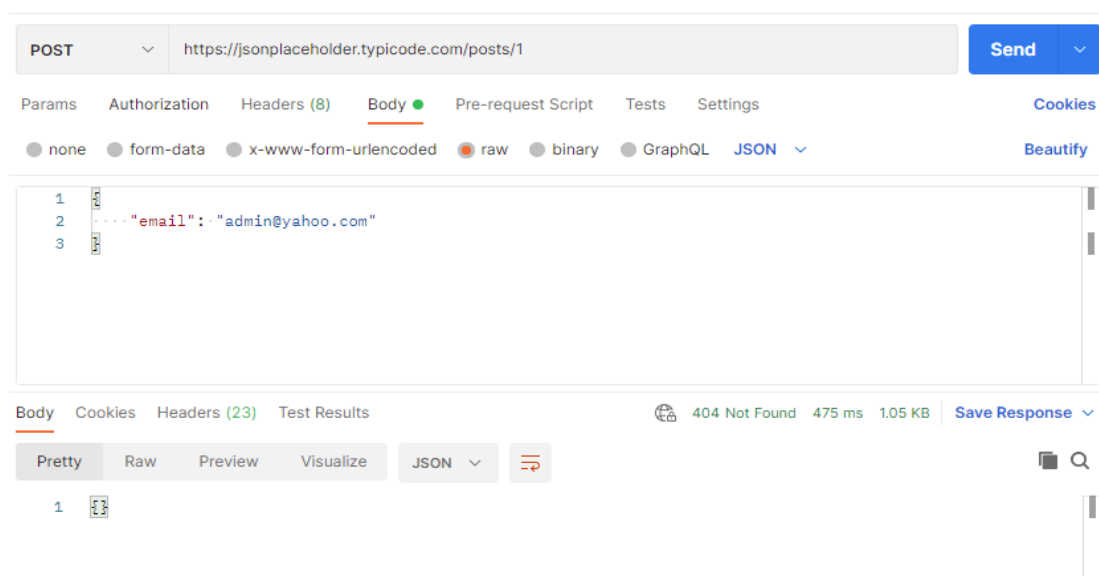


Fig. 13 Interfața Postman

## 2.14 WebClient

WebClient este un client web reactiv și neblocant pentru efectuarea cererilor HTTP. WebClient a fost adăugat în Spring 5 și oferă un API de stil funcțional fluent.

Înainte de Spring 5, principala tehnică de accesare a cererilor de tip HTTP era RestTemplate, care face parte din Spring MVC project. Începând cu Spring 5, abordarea recomandată este WebClient. [20]

De notat este însă faptul că deși este un client neblocant și aparține librăriei spring-webflux, această soluție oferă suport atât pentru operații sincrone cât și pentru operații asincrone, făcându-l potrivit și pentru aplicații care rulează pe stive de servleți.

Acest lucru se poate realiza blocând operația pentru a obține rezultatul. Desigur, această practică nu este recomandată dacă lucrăm pe o stivă reactivă.

Prin reactiv ne referim la programarea modelelor care sunt construite cu scopul de a reacționa la schimbare. (De exemplu, componente de internet care reacționează la

evenimente de intrare/ieșire, controllere pentru interfața cu utilizatorul care reacționează la evenimente ale mouse-ului etc.). În acest sens, un client neblocant este reactiv, pentru că, în loc să fie blocat, acum este în modul de reacționare la notificări în timp ce operațiile se termină sau datele devin disponibile.

Pentru a folosi WebClient avem nevoie de modulul “spring-boot-starter-webflux” care se găsește în Maven Repository.

## 2.15 Feign Client

Feign este un client HTTP declarativ dezvoltat de Netflix.

Un client declarativ poate fi creat prin adnotarea oricărei interfețe sau clase abstracte cu adnotarea @Client. [21]

Feign urmărește să simplifice clienții HTTP. Pe scurt, dezvoltatorii trebuie doar să declare și să adnoteze o interfață în timp ce implementarea în sine este asigurată în timpul execuției. Acest client generează o clasă Proxy în timpul execuției folosind modelul Dynamic Proxy.

Pentru acest concept se folosesc două adnotări:

- @EnableFeignClients - se aplică pe clasa de pornire

@FeignClient(name=”ApplicationName”) - se folosește pentru a defini o interfață pentru un consumator.

### 2.15.1 WebClient vs Feign Client

Spre deosebire de WebClient, care este neblocant, Feign este blocant ceea ce înseamnă că firul de execuție va fi blocat până când clientul Feign primește răspunsul. Problema cu codul blocant este că trebuie să aștepte până când firul de execuție se termină ceea ce înseamnă memorie ocupată și ciclul procesor pierduți.

În concluzie, folosim WebClient când avem nevoie de cereri HTTP neblocante, altfel folosim Feign Client pentru simplitatea lui.

De notat este faptul că putem fără probleme să folosim WebClient pentru operații blocante dar Feign este mai matur și modelul lui bazat pe adnotări îl face mult mai simplu de folosit.

## **2.16 Baze de date**

### **2.16.1 MySql**

MySql este al doilea cel mai popular sistem de gestionare a bazelor de date (după o statistică din Ianuarie 2022), primul loc fiind ocupat de Oracle. MySql este dezvoltat, distribuit și întreținut de Oracle Corporation. [22]

#### **2.16.1.1 MySql este un sistem de gestionare a bazelor de date**

O bază de date este o colecție structurată de date. O bază de date poate conține orice, de la o simplă listă de cumpărături până la cantități mari de informații dintr-o rețea corporativă.

Pentru a adăuga , accesa și procesa datele stocate într-o bază de date, avem nevoie de un sistem de gestionare a bazelor de date cum este MySQL Server. Deoarece calculatoarele sunt foarte bune la manipularea unor cantități mari de date, sistemele pentru gestionarea bazelor de date joacă un rol important în calcul, ca utilități autonome sau ca părți ale altor aplicații.

#### **2.16.1.2 Bazele de date MySql sunt relaționale**

O bază de date relațională stochează datele în tabele separate mai degrabă decât să pună toate datele într-un singur depozit mare.

Structurile bazelor de date sunt organizate în fișiere fizice optimizate pentru viteză. Modelul logic cu obiecte precum baze de date, tabele, rânduri și coloane oferă un mediu de programare flexibil. Se stabilesc reguli care guvernează relațiile dintre diferite câmpuri de date, cum ar fi unu-la-unu , unu-la-mulți , unic , obligatoriu sau opțional și pointeri între diferite tabele. Baza de date aplică aceste reguli, astfel încât, cu o bază de date bine concepută, aplicația dvs. nu va vedea niciodată date inconsecvente, duplicate, orfane, învechite sau date lipsă.

Partea “SQL” din “MySQL” vine de la “Structured Query Language”. SQL este cel mai comun limbaj standardizat folosit pentru accesarea bazelor de date. Depinzând de mediul de programare folosit, putem folosi direct sintaxa SQL, declarații SQL încorporate într-un cod scris în alt limbaj de programare, sau am putea folosi un API specific limbajului care ascunde sintaxa SQL.

Serverul de baze de date MySQL este foarte rapid, de încredere, scalabil și ușor de folosit.

Serverul MySQL poate rula confortabil pe un desktop sau un laptop, alături de alte aplicații, servere web, și așa mai departe , necesitând puțină sau chiar nici un pic de atenție.

MySQL a fost inițial dezvoltat pentru a se ocupa de baze de date de dimensiuni mari mult mai repede decât soluțiile existente și este folosit cu succes în medii de producție cu cerere mare de câțiva ani.

### **2.16.2 MySql Workbench**

MySql poate fi folosit fără probleme din linie de comandă doar că nu este la fel de simplu precum o interfață grafică. De aceea, pentru a face totul mai ușor, Oracle ne pune la dispoziție MySQL Workbench, un instrument vizual pentru proiectarea bazelor de date. Este succesorul instrumentului DBDesigner 4 de la fabFORCE.net și înlocuiește pachetul de programe MySQL GUI Tools Bundle. MySQL Workbench este disponibil pe Windows, Linux și Mac OS X. [23]

MySQL Workbench permite utilizatorilor să proiecteze vizual, să modeleze , să genereze și să gestioneze baze de date. Acest instrument include tot ce are nevoie un modelator de date pentru a crea modele ER (Entity-Relationship) complexe , permite ingineria directă și inversă și, de asemenea oferă caracteristici cheie pentru efectuarea modificărilor sarcinilor dificile legate de management și documentare care în mod normal ar necesita un timp și efort îndelungat.

#### **2.16.2.1 Dezvoltare**

MySQL Workbench oferă instrumente vizuale pentru crearea, executarea și optimizarea interogărilor SQL. Editorul SQL oferă evidențierea sintaxei, completarea automată, reutilizarea fragmentelor SQL și istoricul execuției SQL. Panoul de conexiuni la baze de date le permite dezvoltatorilor să gestioneze cu ușurință conexiunile standard la baze de date, inclusiv MySQL Fabric.



### 2.16.2.2 Migrarea bazelor de date

MySQL Workbench oferă acum o soluție completă și ușor de utilizat pentru migrarea Microsoft SQL Server, Microsoft Access, Sybase ASE, PostgreSQL și alte tabele, obiecte și date RDBMS către MySQL. Dezvoltatorii și DBA(Database Administrator) pot converti rapid și ușor aplicațiile existente pentru a rula pe MySQL atât pe Windows, cât și pe alte platforme. Migrarea acceptă, de asemenea, migrarea de la versiunile anterioare de MySQL la cele mai recente versiuni.

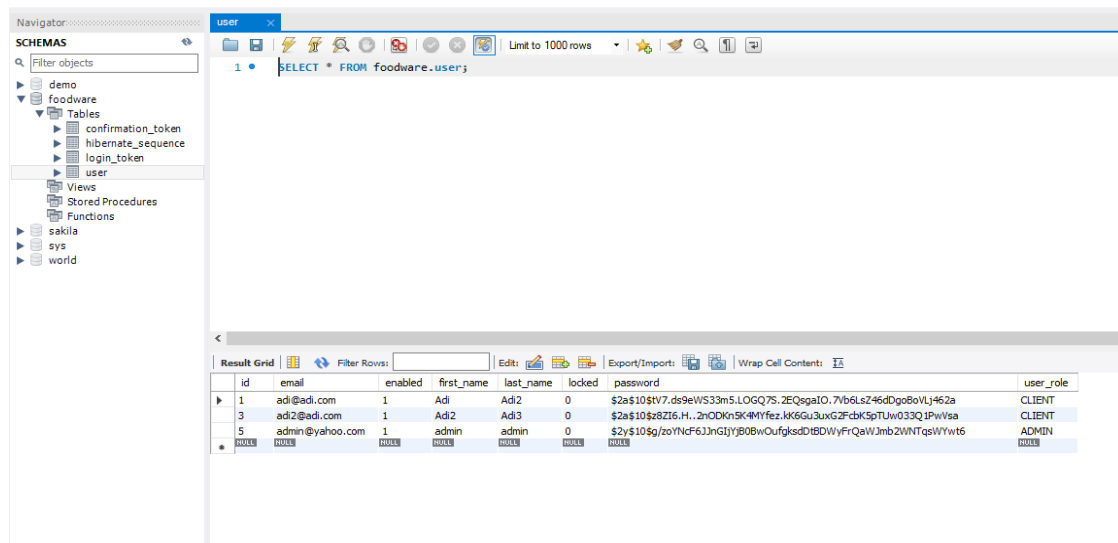


Fig. 14 Interfața MySQL Workbench

## 2.17 Hibernate

Hibernate este un instrument open source de mapare relațională a obiectelor (ORM) care oferă un cadru pentru maparea modelelor de domenii orientate pe obiecte la baze de date relaționale pentru aplicații web.

Maparea relațională a obiectelor se bazează pe containerizarea obiectelor și pe abstractizarea care asigură această capacitate. Abstracția face posibilă adresarea, accesarea și manipularea obiectelor fără a fi nevoie să luați în considerare modul în care acestea sunt legate de sursele lor de date.

Cadrul Hibernate ORM ghidează maparea claselor Java la tabele de baze de date și a tipurilor de date Java la tipuri de date SQL și oferă interogări și regăsire.

### 2.17.1 Beneficiile Hibernate

Orice modificări efectuate sunt încapsulate în sursa de date în sine, astfel încât atunci când acele surse sau interfețele lor de programare a aplicațiilor (API) se schimbă,

aplicațiile care folosesc ORM nu trebuie să facă modificări sau chiar să fie conștiente de acele informații. În mod similar, programatorii pot avea o vedere consecventă asupra obiectelor în timp, deși sursele care le livrează, clasele care le primesc (sinks) și aplicațiile care le accesează se pot schimba.

Hibernate este disponibil gratuit pentru descărcare și este licențiat în baza licenței publice GNU Lesser General Public License (LGPL) cu sursă deschisă.

### **2.23.1 Cum funcționează Hibernate?**

Hibernate este un serviciu open source de persistență și interogare obiect-relațională pentru orice aplicație Java. Hibernate mapează clasele Java la tabelele bazei de date și de la tipurile de date Java la tipurile de date SQL și scutește dezvoltatorul de cele mai comune sarcini de programare legate de persistența datelor.

Hibernate se află între obiectele tradiționale Java și serverul de baze de date pentru a gestiona toate lucrările de persistență a acestor obiecte pe baza mecanismelor și modelelor O/R adecvate.

### **2.17.2 De ce să folosiți Hibernate?**

Hibernate reduce liniile de cod prin menținerea mapării obiect-tabel în sine și returnează rezultatul aplicației sub formă de obiecte Java. Eliberează programatorul de manipularea manuală a datelor persistente, reducând astfel timpul de dezvoltare și costul de întreținere.

### **2.17.3 Hibernate și JPA versus JDBC**

Java Database Connectivity (JDBC) este un API împachetat cu ediția Java SE care face posibilă standardizarea și simplificarea procesului de conectare a aplicațiilor Java la sisteme externe de gestionare a bazelor de date relaționale (RDBMS).

În mod fundamental, aplicațiile scrise în Java realizează logica. Limbajul Java oferă facilități pentru efectuarea logicii iterative cu bucle(loops), logica condiționată cu instrucțiuni if și analiză orientată pe obiecte prin utilizarea claselor și interfețelor. Dar aplicațiile Java nu stochează date în mod persistent.

Persistența datelor este de obicei delegată bazelor de date NoSQL, cum ar fi MongoDB și Cassandra, sau bazelor de date relaționale, cum ar fi IBM DB2 sau Microsoft SQL Server sau populara bază de date open source MySQL.

Pentru a ajuta la abordarea nepotrivirii obiect-relaționale, există o serie de framework-uri care simplifică sarcina de a muta datele între o bază de date relațională și un program Java. Framework-urile populare de mapare obiect-relațională (ORM) includ Hibernate, TopLink și DataNucleus.

## 2.18 DDL (Data Definition Language)

Limbajul de definire a datelor (DDL) este un limbaj care permite utilizatorului să definească datele și relația lor cu alte tipuri de date.[24]

Instrucțiunile limbajului de definire a datelor funcționează cu structura tabelelor bazei de date.

### 2.23.1 Comenzi DDL

- Create - Este folosit pentru a crea un tabel nou sau o bază de date nouă.
- Alter - Este folosit pentru a modifica sau modifica structura tabelului bazei de date.
- Drop - Este folosit pentru a șterge un tabel, index sau vederi din baza de date.
- Truncate - Este folosit pentru a șterge înregistrările sau datele din tabel, dar structura acestuia rămâne așa cum este.
- Rename - Este folosit pentru a redenumi un obiect din baza de date.

Când creați un tabel, trebuie să specificați următoarele:

- Numele tabelului.
- Numele fiecărei coloane.
- Tipul de date al fiecărei coloane.
- Dimensiunea fiecărei coloane.

Hibernate oferă posibilitatea de a genera automat tabelele bazei de date pe baza entităților definite în Java, alternativa fiind folosirea fișierelor DDL sau generarea bazei de date într-un sistem pentru gestionarea bazelor de date.

Eu am ales folosirea unui fișier DDL pentru generarea bazei de date pentru a avea mai mult control asupra tabelelor și coloanelor acestora.

## 2.19 DML (Data Manipulation Language)

Instrucțiunile DML sunt folosite pentru a lucra cu datele din tabele. Când sunteți conectat la majoritatea bazelor de date cu mai mulți utilizatori (fie într-un program client sau printr-o conexiune dintr-un script de pagină Web), lucrați de fapt cu o copie privată a tabelelor dvs. care nu poate fi văzută de altcineva până când nu sunt terminate (sau spuneți sistemului că ați terminat). [25]

Instrucțiunea “insert” este folosită pentru a adăuga noi rânduri într-un tabel.

```
INSERT INTO <table name>  
VALUES (<value 1>, ... <value n>);
```

Fig. 15 Exemplu de comandă DDL pentru inserare

Lista de valori delimitată prin virgulă trebuie să se potrivească exact cu structura tabelului în numărul de atribute și tipul de date al fiecărui atribut. Valorile de tip caracter sunt întotdeauna cuprinse între ghilimele simple; valorile numerice nu sunt niciodată între ghilimele; valorile datei sunt adesea (dar nu întotdeauna) în formatul „aaaa-ll-zz” (de exemplu, „2006-11-30”). Instrucțiunea “update” este utilizată pentru a modifica valorile care sunt deja într-un tabel.

```
UPDATE <table name>  
SET <attribute> = <expression>  
WHERE <condition>;
```

Fig. 16 Exemplu de comandă DDL pentru update

Instrucțiunea update poate fi o constantă, orice valoare calculată sau chiar rezultatul unei instrucțiuni SELECT care returnează un singur rând și o singură coloană. Dacă clauza WHERE este omisă, atunci atributul specificat este setat la aceeași valoare în fiecare rând al tabelului (ceea ce de obicei nu este ceea ce doriți să faceți). De asemenea, puteți seta mai multe valori de atribut în același timp cu o listă de perechi atribut-expresie delimitată prin virgulă. Instrucțiunea delete șterge rândurile dintr-un tabel.

```
DELETE FROM <table name>  
WHERE <condition>;
```

Fig. 17 Exemplu de comandă DDL pentru ștergere

Dacă clauza WHERE este omisă, atunci fiecare rând al tabelului este șters (care din nou, de obicei, nu este ceea ce doriți să faceți) - și din nou, nu veți primi un mesaj de genul „chiar doriți să faceți asta?” .

## 2.20 Project Lombok

Java este un limbaj foarte popular, dar are și unele dezavantaje. Unul dintre cele mai populare dezavantaje este că încă mai trebuie să scriem codurile standard, cum ar fi getters, setters, metoda toString în Java, în timp ce Kotlin și Scala, care sunt, de asemenea, bazate pe JVM, nu au nevoie de acest lucru și, prin urmare, acesta este motivul pentru creșterea lor de popularitate în comunitate. Aici intervine Lombok și depășește acest dezavantaj al Java. [26]

Proiectul Lombok este o bibliotecă Java care este folosită pentru a minimiza/elimina codul standard și pentru a salva timpul prețios al dezvoltatorilor în timpul dezvoltării, folosind doar câteva adnotări. În plus față de acesta, crește și lizibilitatea codului sursă și economisește spațiu. Dar s-ar putea să vă gândiți că în zilele noastre toată lumea folosește IDE-uri care oferă o opțiune pentru generarea acestor coduri standard, atunci la ce folosește Lombok. Ori de câte ori folosim IDE-uri pentru a genera aceste coduri standard, pur și simplu ne ferim de a scrie toate aceste coduri, dar este prezent de fapt în codul nostru sursă și crește LOC (liniile de cod) și reduce mentenabilitatea și lizibilitatea. Pe de altă parte, Lombok adaugă toate aceste coduri standard la momentul compilării în fișierul „.class” și nu în codul nostru sursă.

Să comparăm codul sursă cu și fără utilizarea Lombok.

1. Fără Lombok: O clasă de model java cu patru câmpuri private și getters, setters, constructor no-args, constructor parametrizat și metoda toString.

```
public class Employee {  
    private Integer employeeId;  
    private String name;  
    private String company;  
    private String emailId;  
  
    public Employee() {}  
  
    public Employee(Integer employeeId, String name,  
                    String company, String emailId)  
    {  
        super();  
        this.employeeId = employeeId;  
        this.name = name;  
        this.company = company;  
        this.emailId = emailId;  
    }  
  
    public Integer getEmployeeId() { return employeeId; }  
    public void setEmployeeId(Integer employeeId)  
    {  
        this.employeeId = employeeId;  
    }  
  
    public String getName() { return name; }  
    public void setName(String name) { this.name = name; }  
    public String getCompany() { return company; }  
    public void setCompany(String company)  
    {  
        this.company = company;  
    }  
  
    public String getEmailId() { return emailId; }  
    public void setEmailId(String emailId)  
    {  
        this.emailId = emailId;  
    }  
  
    @Override public String toString()  
    {  
        return "Employee ["  
            + "employeeId=" + employeeId + ", name=" + name  
            + ", "  
            + "company=" + company + ", emailId=" + emailId  
            + "];"  
    }  
}
```

Fig. 18 Exemplu de clasă de model java fără să folosim Lombok

2. Cu Lombok: O clasă de model java cu patru câmpuri private și getters, setters, constructor no-args, construcție parametrizată și metoda toString folosind adnotări Lombok.

```

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@NoArgsConstructor
@AllArgsConstructor
@ToString
public class Employee {
    private @Getter @Setter Integer employeeId;
    private @Getter @Setter String name;
    private @Getter @Setter String company;
    private @Getter @Setter String emailId;
}

```

Fig. 19 Exemplu de clasă de model java folosind Lombok

## 2.21 Controlul versiunilor

### 2.23.2 Git

Controlul versiunilor se referă la procesul de salvare a diferitelor fișiere sau „versiuni” în diferitele etape ale unui proiect. Acest lucru le permite dezvoltatorilor să urmărească ceea ce s-a făcut și să revină la o fază anterioară dacă decid că doresc să anuleze unele dintre modificările pe care le-au făcut.

Acest lucru este util din mai multe motive. De exemplu, facilitează rezolvarea erorilor și remedierea altor greșeli care ar putea apărea în timpul dezvoltării. De asemenea, puteți nota modificări în fiecare versiune, pentru a ajuta membrii echipei să rămână la curent cu ceea ce s-a finalizat și ce mai trebuie să fie realizat.

De asemenea, Git vă permite să „împingeți”(push) și să „trageți”(pull) modificări către și de la instalări pe alte computere. Acest lucru îl face cunoscut sub numele de „Sistem de control al versiunilor distribuite” și permite mai multor dezvoltatori să lucreze la același proiect.

Cu toate acestea, există câteva dezavantaje în gestionarea dezvoltării în acest fel. Deoarece software-ul local este instalat pe computerul dvs. individual, git nu poate citi editările pe care alți dezvoltatori le pot face în timp real. Aceasta înseamnă că, dacă tu și un coechipier lucrați simultan la un proiect, nu vă veți putea vedea munca celuilalt.

### 2.23.3 Diferența dintre Git și GitHub [27]

GitHub facilitează colaborarea folosind git. Este o platformă care poate deține depozite de cod în stocare bazată pe cloud, astfel încât mai mulți dezvoltatori să poată lucra la un singur proiect și să vadă editările celorlalți în timp real:

În plus, include și funcții de organizare și gestionare a proiectelor. Puteți să atribui sarcini unor persoane sau grupuri, să setați permisiuni și roluri pentru colaboratori și să utilizați moderarea comentariilor pentru a menține toată lumea în temă.

În plus, depozitele GitHub sunt disponibile public. Dezvoltatorii de pe tot globul pot interacționa cu codul celuilalt și pot contribui la el pentru a-l modifica sau îmbunătăți, ceea ce este cunoscut sub numele de „codare socială”. Într-un fel, acest lucru face din GitHub un site de rețea pentru profesioniștii web.

Există trei acțiuni principale pe care le puteți face atunci când vine vorba de interacțiunea cu codul altor dezvoltatori pe GitHub:

- Fork: Procesul de copiere a codului altuia din depozit pentru a-l modifica.
- Pull: după ce ați terminat de făcut modificări la codul altcuiva, le puteți partaja proprietarului inițial printr-o „cerere de extragere”(pull request).
- Merge: proprietarii pot adăuga noi modificări la proiectele lor printr-o fuziune și pot acorda credit colaboratorilor care le-au sugerat.

Pentru a rezuma diferența dintre Git și GitHub:

Git este un software VCS local care le permite dezvoltatorilor să salveze versiuni ale proiectelor lor în timp. În general, este cel mai bine pentru uz individual.

GitHub este o platformă web care încorporează caracteristicile de control al versiunilor Git , astfel încât acestea să poată fi utilizate în colaborare. Include, de asemenea, funcții de management de proiect și de echipă, precum și oportunități pentru crearea de rețele și codificare socială.

### 2.23.4 Proprietatea și costul Git vs GitHub

Deoarece sunt atât de strâns legate, ar avea sens dacă Git și GitHub ar fi deținute de aceeași companie. Dimpotrivă, Git este un software open source, în timp ce GitHub este deținut de Microsoft.

Platformele open source – inclusiv Git și WordPress – pot fi utilizate, modificate și distribuite gratuit.



Modelul de preț al GitHub este diferit, dar oferă un plan gratuit interesant. De fapt, toate funcțiile de bază GitHub sunt gratuite pentru toată lumea.

## **2.22 IntelliJ IDEA**

IntelliJ IDEA este un mediu de dezvoltare integrat (IDE) pentru limbaje JVM conceput pentru a maximiza productivitatea dezvoltatorului. Îndeplinește sarcinile de rutină și repetitive pentru dvs., oferind completare inteligentă a codului, analiză statică a codului și refactorizări și vă permite să vă concentrați pe partea bună a dezvoltării software, făcându-l nu numai productiv, ci și o experiență plăcută. [28]

### **2.23.1 Multi-platformă**

IntelliJ IDEA este un IDE multi-platformă care oferă o experiență constantă pe Windows, macOS și Linux.

#### **Limbi acceptate**

Dezvoltarea aplicațiilor moderne implică utilizarea mai multor limbi, instrumente, cadre și tehnologii. IntelliJ IDEA este conceput ca un IDE pentru limbaje JVM, dar numeroase plugin-uri îl pot extinde pentru a oferi o experiență poliglotă.

### **2.23.2 Limbaje JVM**

Utilizați IntelliJ IDEA pentru a dezvolta aplicații în următoarele limbi care pot fi compilate în bytecode JVM, și anume:

- Java
- Kotlin
- Scala
- Groovy

### **2.23.3 Edițiile IntelliJ IDEA**

IntelliJ IDEA vine în trei ediții:

#### **2.22.3.1 IntelliJ IDEA Ultimate**

Ediția comercială pentru JVM, web și dezvoltarea întreprinderilor. Include toate caracteristicile ediției comunitare, plus adaugă suport pentru limbile pe care se concentrează alte IDE-uri bazate pe platforma IntelliJ, precum și suport pentru o varietate

de framework-uri pe server și front-end, servere de aplicații, integrare cu bazele de date și profilare. instrumente și multe altele.

### 2.22.3.2 IntelliJ IDEA Community Edition

Ediția gratuită bazată pe open-source pentru dezvoltarea JVM și Android.

### 2.22.3.3 IntelliJ IDEA Edu

Ediția gratuită cu lecții încorporate pentru învățarea Java, Kotlin și Scala, sarcini și sarcini interactive de programare și funcții speciale pentru ca profesorii să își creeze propriile cursuri și să gestioneze procesul educațional (vezi IntelliJ IDEA Edu).

## 2.23.4 Navigare și căutare

IntelliJ IDEA oferă o navigare rapidă nu numai în interiorul fișierelor de cod sursă, ci și în întregul proiect.

Una dintre cele mai utile comenzi rapide care merită reținută este Shift dublu, care afișează dialogul “Căutați peste tot”: începeți să tastați și IntelliJ IDEA va căuta șirul dvs. de căutare printre toate fișierele, clasele și simbolurile care aparțin proiectului dvs. și chiar printre Acțiuni IDE.

## 2.23.5 Structura fișierului

Apăsați Ctrl+F12 pentru a deschide fereastra pop-up cu structura fișierului care vă oferă o privire de ansamblu asupra tuturor elementelor utilizate în fișierul curent și vă permite să săriți la oricare dintre ele:

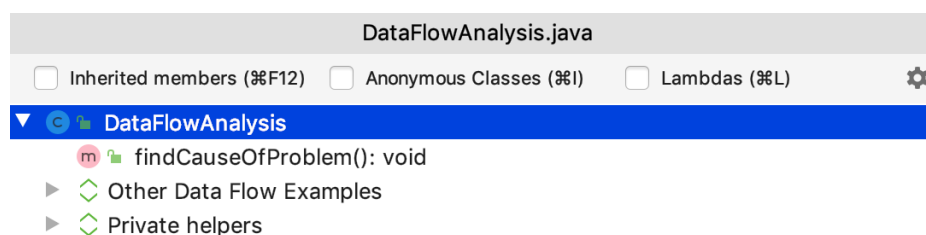


Fig. 20 Fereastra cu structura unui fișier în IntelliJ

### 2.23.6 Completarea codului

IntelliJ IDEA vă ajută să accelerați procesul de codare, oferind completarea codului în funcție de context.

- Completarea de bază vă ajută să completați numele claselor, metodelor, câmpurilor și cuvintelor cheie în domeniul de vizibilitate:

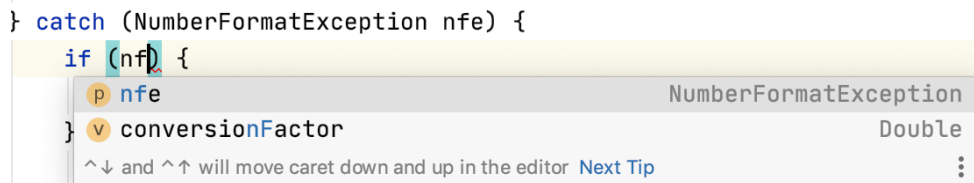


Fig. 21 Exemplu de autocompletare oferită de IntelliJ

- Completarea inteligentă sugerează cele mai relevante simboluri aplicabile în contextul actual când IntelliJ IDEA poate determina tipul potrivit:

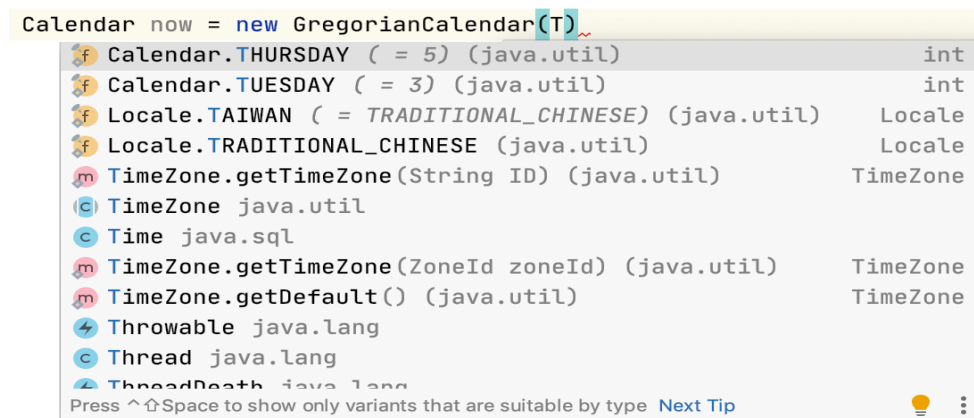


Fig. 22 Exemplu de sugestii inteligente pe baza tipului de date oferite de IntelliJ

### 2.23.7 Integrarea cu instrumentele de dezvoltare

Pe lângă furnizarea de asistență inteligentă pentru navigare și codare, IntelliJ IDEA integrează instrumentele esențiale pentru dezvoltatori și vă permite să depanați, să analizați și să organizați baza de cod a aplicațiilor dvs. din IDE.

### 2.23.8 Depanator

IntelliJ IDEA oferă un depanator JVM încorporat. Vă permite să obțineți și să analizați informațiile de rulare, care sunt utile pentru diagnosticarea problemelor și pentru a obține o înțelegere mai profundă a modului în care funcționează un program. Vă permite

să suspendați execuția programului pentru a-i examina comportamentul folosind punctele de întrerupere. Mai multe tipuri de puncte de întrerupere, împreună cu condiții și filtre, vă permit să specificați momentul exact în care o aplicație trebuie să fie întreruptă.

IntelliJ IDEA vă permite să modificați valorile variabilelor în timpul rulării, evaluând expresii și așa mai departe.

Examinați valorile variabilelor, stivele de apeluri, stările firelor și așa mai departe. Controlați execuția pas cu pas a programului.

### **2.23.9 Instrumente pentru construcția proiectului**

IntelliJ IDEA vine cu o integrare Gradle și Maven complet funcțională, care vă permite să vă automatizați procesul de construire, împachetare, rularea testelor, implementarea și alte activități.

Când deschideți un proiect Gradle sau Maven existent sau creați unul nou, IntelliJ IDEA detectează și descarcă automat toate depozitele și pluginurile necesare, astfel încât practic nu trebuie să configurați nimic și vă puteți concentra doar pe procesul de dezvoltare. Puteți edita fișierele build.gradle și pom.xml direct din editor și puteți configura IDE-ul pentru a sincroniza automat toate modificările la configurațiile de construire.

### **2.23.10 Controlul versiunii**

IntelliJ IDEA oferă integrare cu cele mai populare instrumente de control al versiunilor: Git, Mercurial, Perforce și Subversion.

Puteți să revizuiți istoricul întregului proiect sau al fișierelor separate, să comparați versiunile fișierelor, să gestionați ramurile și chiar să procesați cererile de extragere GitHub fără a părăsi IDE-ul.

Puteți accesa rapid toate acțiunile VCS din fereastra pop-up operațiuni VCS Alt+`:

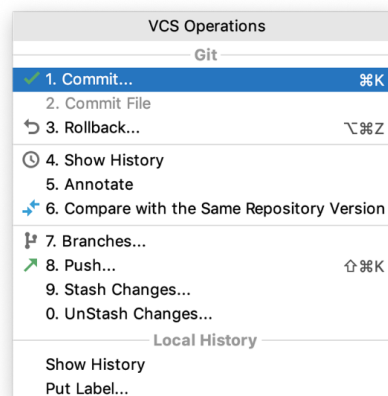


Fig. 23 Fereastra pentru controlul versiunilor din IntelliJ

### 2.23.11 Istorie locală

Chiar dacă nu este încă activat controlul versiunilor pentru proiectul dvs., puteți încă să urmăriți modificările aduse proiectului și să restaurați fișierele șterse sau modificările separate cu Istoricul local. Acționează ca un sistem personal de control al versiunilor, care înregistrează automat revizuirile proiectului declanșate de diverse evenimente pe măsură ce editați codul, rulați teste, implementați aplicații și așa mai departe.

## 2.23 Trello

Trello este un instrument de gestionare vizuală a muncii care dă echipelor puterea de a crea, planifica, gestiona și sărbători munca lor împreună într-un mod colaborativ, productiv și organizat. [29]

Indiferent dacă tu și echipa ta începeți ceva nou sau încercați să vă organizați mai bine cu munca dvs. existentă, Trello se adaptează oricărui proiect. Vă ajută să simplificați și să standardizați procesul de lucru al echipei dvs. într-un mod intuitiv. Trello este ușor de utilizat, dar totuși capabil să gestioneze cele mai solide proiecte ale echipei tale.

Trello conține 4 elemente cheie :

### 2.23.1 Panoul

Un panou (fig 25 - A) reprezintă un loc pentru a ține evidența informațiilor - adesea pentru proiecte mari, echipe sau fluxuri de lucru. Indiferent dacă lansați un nou site web, urmăriți vânzările sau planificați următoarea petrecere la birou, o tablă Trello

este locul pentru a organiza sarcinile, toate micile detalii și, cel mai important, pentru a colabora cu colegii tăi.

### 2.23.2 Liste

Listele (fig 25 - B) păstrează carduri, sau sarcini specifice sau informații, organizate în diferitele lor etape de progres. Listele pot fi folosite pentru a crea un flux de lucru în care cardurile sunt mutate de-a lungul fiecărei etape a procesului de la început până la sfârșit sau pur și simplu acționează ca un loc pentru a ține evidența ideilor și informațiilor. Nu există limită pentru numărul de liste pe care le puteți adăuga la un panou și pot fi aranjate și intitulate după cum doriți.

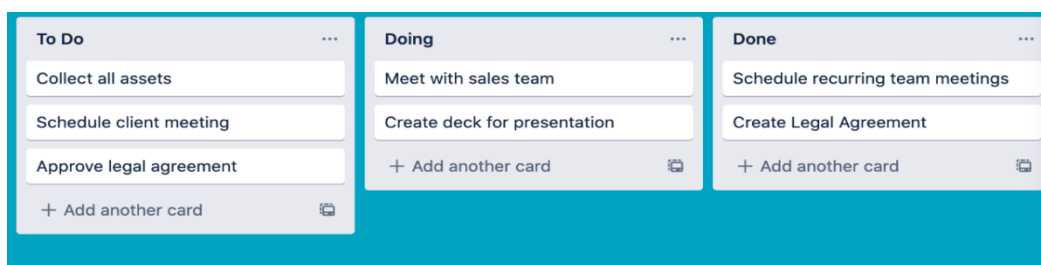


Fig. 24 Un proiect Trello care conține 3 liste fiecare conținând carduri

### 2.23.3 Carduri

Cea mai mică, dar cea mai detaliată unitate a unei table este un card (fig 25 - C). Cardurile sunt folosite pentru a reprezenta sarcini și idei. Un card poate fi ceva care trebuie făcut, cum ar fi o postare pe blog care trebuie scrisă, sau ceva care trebuie reținut, cum ar fi politicile de vacanță ale companiei.

Cardurile pot fi personalizate pentru a conține o mare varietate de informații utile făcând clic pe ele. Trageți și plasați carduri pe liste pentru a afișa progresul. Nu există limită pentru numărul de cărți pe care le puteți adăuga la o tablă.

### 2.23.4 Meniul panoului

În partea dreaptă a plăcii Trello se află meniul (fig 25 - D) - centrul de control al misiunii pentru panoul dvs. Meniul este locul în care gestionați permisiunile membrilor consiliului de administrație, controlați setările, căutați carduri, activați pornirile și creați automatizări. De asemenea, puteți vedea toate activitățile care au avut loc pe un panou în fluxul de activități din meniu.

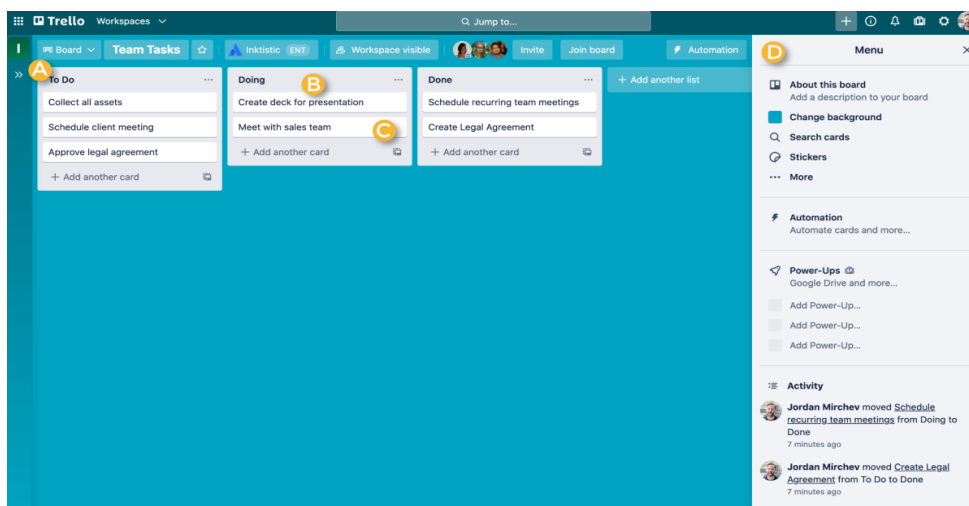


Fig.25 Interfața Trello

## 2.24 Confluence

Confluence este un instrument wiki de colaborare folosit pentru a ajuta echipele să colaboreze și să împărtășească cunoștințele în mod eficient. Cu confluență, putem capta cerințele proiectului, atribui sarcini anumitor utilizatori și gestiona mai multe calendare simultan cu ajutorul suplimentului Team Calendars. [30]

Caracteristici ale Confluence:

- Confluence este un editor colaborativ puternic, deoarece vă oferă puterea de a crea notițe, planuri de proiect, cerințe de produs, în același timp în care alți utilizatori editează și văd toate modificările simultan.
- Feedback reprezentat de comentarii pe pagini și fișiere atașate.
- Poate oferi feedback direct asupra fișierelor dvs. (imagini, PDF-uri, foi de calcul și prezentări) și ține automat evidența versiunilor, astfel încât să lucrați mereu la cea potrivită.
- Spații de lucru organizate: în Confluence, puteți crea un spațiu pentru fiecare echipă, departament sau proiect major pentru a împărtăși informații și a menține munca organizată.
- Versiune pentru pagini și fișiere: puteți urmări fiecare versiune și modificarea pe care o faceți unei pagini. Fișierele sunt versionate automat.
- O varietate de metode de a partaja conținutul dvs. de confluence: vă puteți „abona” la modificările de conținut prin notificări prin e-mail sau RSS, puteți trimite prin e-mail o pagină direct utilizatorilor.

- Controale granulare ale permisiunilor: restricționează accesul la conținut la trei niveluri – global, spațiu și per pagină individuală.
- Include, de asemenea, un editor de text îmbogățit cu “drag and drop” de atașamente, integrare profundă cu Microsoft Office, comenzi rapide de la tastatură și încorporare de conținut bogat.



### Capitolul III. Arhitectura aplicațiilor

Proiectul este alcătuit din 4 microservicii (Menu, Orders, Authentication, Payment) unite printr-un al 5-lea microserviciu(BackEnd for FrontEnd),toate scrise în limbajul Java , cel din urmă fiind apelat de către aplicația FrontEnd scrisă în limbajul TypeScript.

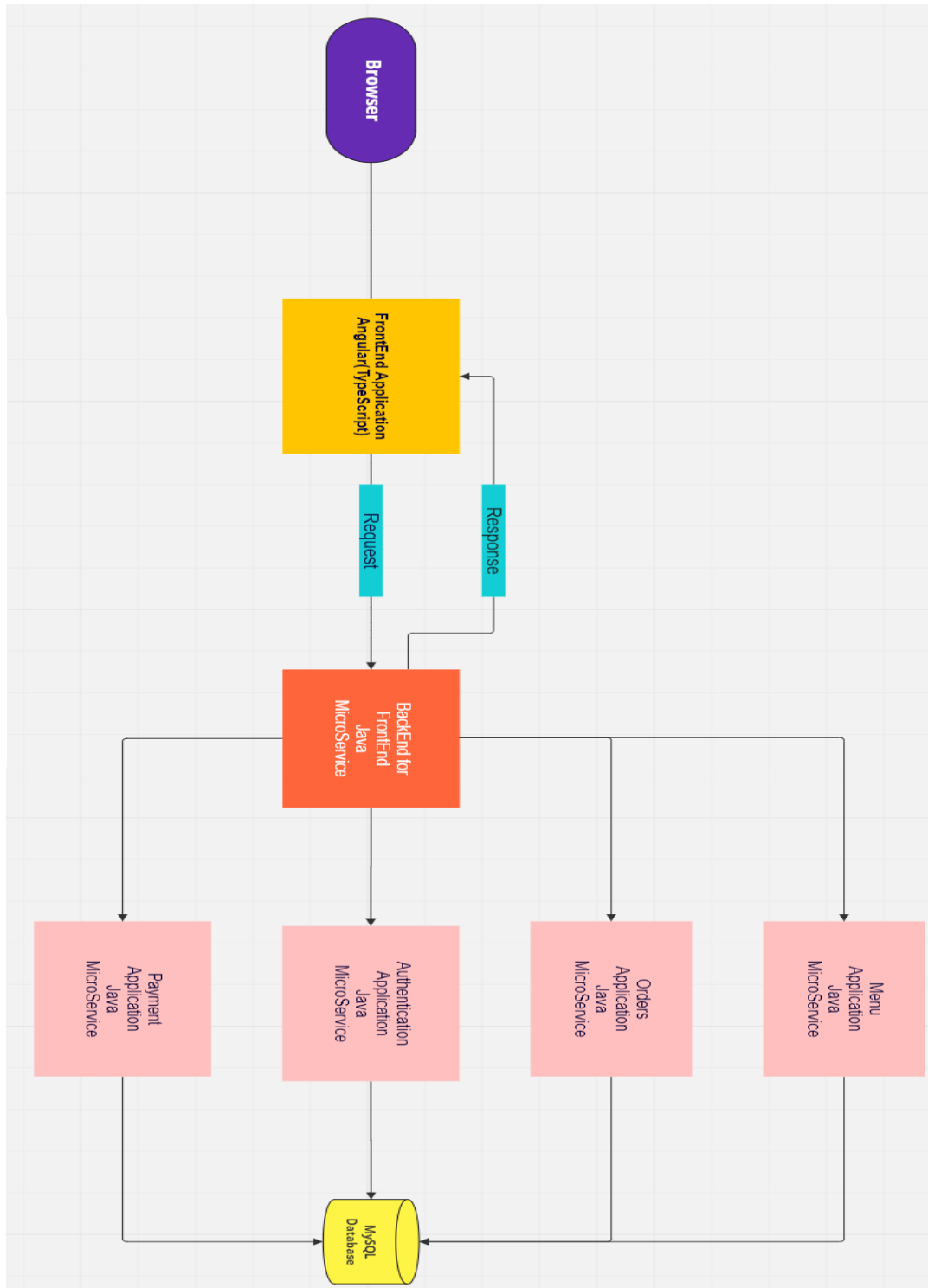


Fig. 26 Arhitectura proiectului

Microserviciile se folosesc de arhitectura REST pentru a comunica cu exteriorul prin endpoint-uri.

### 3.1 Aplicația pentru meniu

Expune endpoint-uri pentru inserarea, modificarea , ștergerea și citirea unui/unei:

- produs(ex:Paste carbonara),
- extra produs(ex: Sos de usturoi),
- categorii de produse (ex: Pizza,Paste,Ceai),
- grup de categorii (ex: Mâncare, Băutură)
- imagini asociate produselor

De asemenea , aplicația pentru meniu mai expune endpoint-uri și pentru aducerea produselor din baza de date în funcție de grupul de categorii sau categoria de produse precizată.

Un exemplu de endpoint din cadrul aplicației de meniu este prezentat în fig. 26 și este folosit pentru a salva un nou produs în baza de date.

```
@PostMapping(path = "insert", consumes = {MediaType.MULTIPART_FORM_DATA_VALUE})
public String addProduct(@RequestPart Product product, @RequestPart(required = false) MultipartFile image) {
    if (image!=null) {
        int imageId = imageService.insertImage(image);
        product.setImageId(imageId);
    }
    return productService.insertProduct(product);
}
```

Fig. 27 Exemplu de endpoint din aplicația de meniu

După cum se poate observa, în figură este prezentat un request de tip POST care primește un parametru obligatoriu, produsul în sine (prezentat în figură ca “product”) și un parametru opțional de tip MultipartFile care reprezintă imaginea aferentă produsului.

### 3.2 Aplicația pentru comenzi (orders)

Expune endpoint-uri pentru inserarea, modificarea , ștergerea și citirea unui/unei:

- comenzi (“order”). Comanda poate conține una sau mai multe subcomenzi, fiecare subcomandă aparținând unei persoane diferite aflate la masa pe care s-a deschis comanda.

- subcomenzi("subOrder"). O subcomandă conține id-ul utilizatorului care a inițiat subcomanda și id-ul plății.
- produs care aparține unei subcomenzi ("orderItem"). Această entitate are ca și proprietăți : cantitatea produsului selectat, id-ul produsului și id-ul subcomenzii din care face parte.
- plăți ("payment"), această entitate având ca și proprietăți suma totală de plată și metoda de plată.
- status (ex: "comandă preluată" sau "comandă finalizată")

### 3.3 Aplicația pentru autentificare

Aplicația pentru autentificare permite utilizatorilor să-și creeze un cont nou și să se logheze. Autentificarea se face cu ajutorul Spring Security. Spring Security ne pune la dispoziție un interceptor care preia request-urile înainte de a ajunge la controller și verifică dacă ele conțin un token atașat. Acest token trebuie să conțină numele utilizatorului, data la care a fost emis și data la care expiră.

La fiecare request, se verifică ca acest token să nu fie expirat. Dacă token-ul a expirat, utilizatorul trebuie să se logheze din nou.

În cadrul aplicației există 2 roluri pentru utilizatori: CLIENT și ADMIN.

Deocamdată, aplicația fiind doar un proof of concept pentru partea de client, se pot crea doar conturi de tip CLIENT.

### 3.4 Aplicația pentru plată (payment)

Aplicația pentru plată deocamdată este doar în stadiul de mock, în dezvoltări ulterioare putând folosi o platformă pentru plăți online sau chiar propria implementare pentru o platformă de plăți.

Aplicația este gândită să proceseze plăți separate pentru fiecare membru participant la o masă.

### 3.5 Aplicația de legătură

Aplicația de legătură folosește modelul Backend for Frontend care presupune preluarea unei părți din logica de pe frontend, pe backend. Este un nivel intermediar între frontend și microservicii.

Logica este în felul următor:

Aplicația de legătură va apela microserviciile necesare pentru preluarea datelor din baza de date. Datele vor fi formate pentru a se potrivi aplicației frontend iar apoi vor fi trimise aplicației frontend pentru a putea fi afișate.

### 3.6 Baza de date

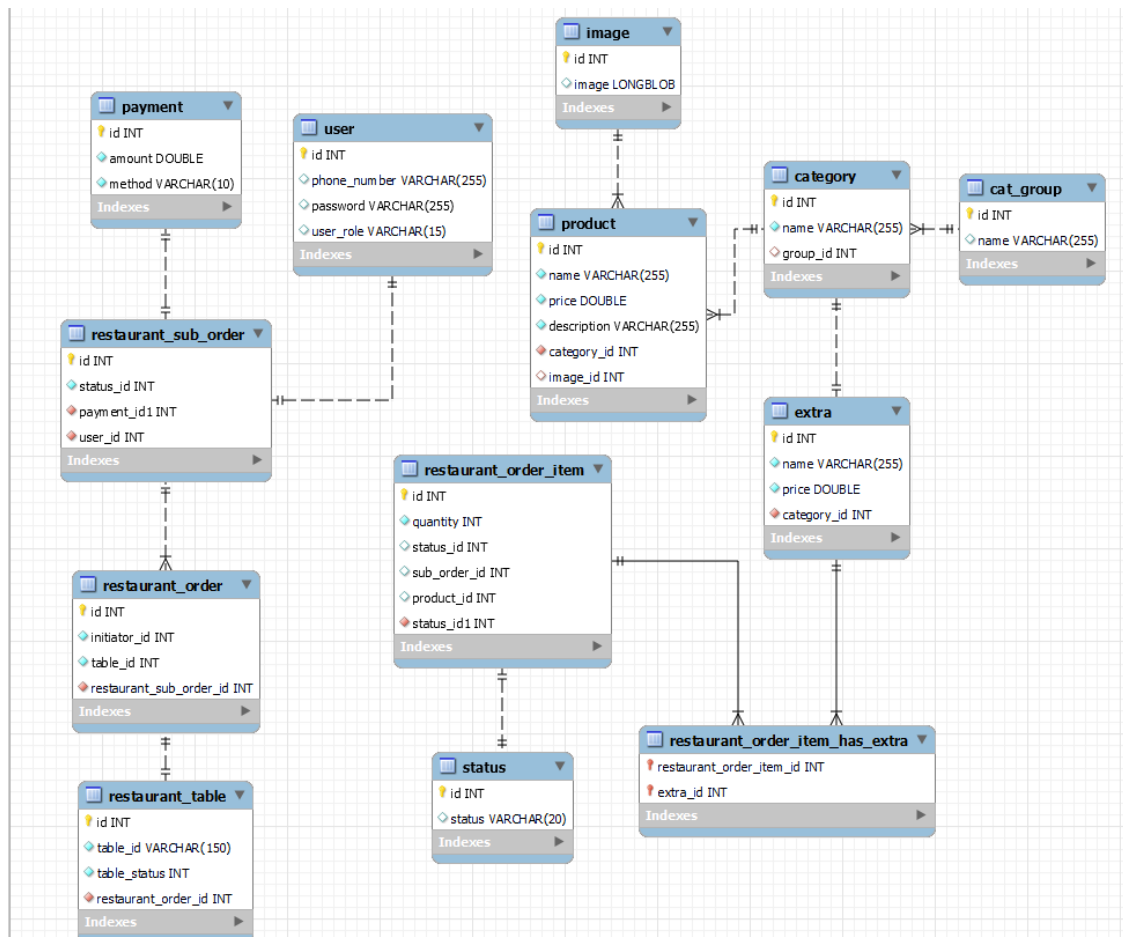


Fig. 27 Structura bazei de date

Pentru baza de date am folosit limbajul MySQL și instrumentul pentru proiectarea bazelor de date: MySQL WorkBench.

Toate aplicațiile sunt conectate la aceeași bază de date și lucrează pe tabele separate.

Baza de date este gândită pentru scalabilitate, proiectul putând fi utilizat pentru mai multe restaurante, nu este specific doar pentru unul.

### **3.7 Detalii de implementare**

Pentru toate aplicațiile am folosit modelul MVC (Model View Controller) în care partea de model și controller se află în cadrul aplicațiilor backend iar partea de view este asigurată de aplicația frontend.

Logica proiectului este păstrată în mare măsură în partea de backend, aplicația de frontend ocupându-se majoritar doar de afișarea datelor primite.

### 3.7.1 Interfața cu utilizatorul (aplicația frontend)

#### 3.7.1.1 Flow-ul aplicației

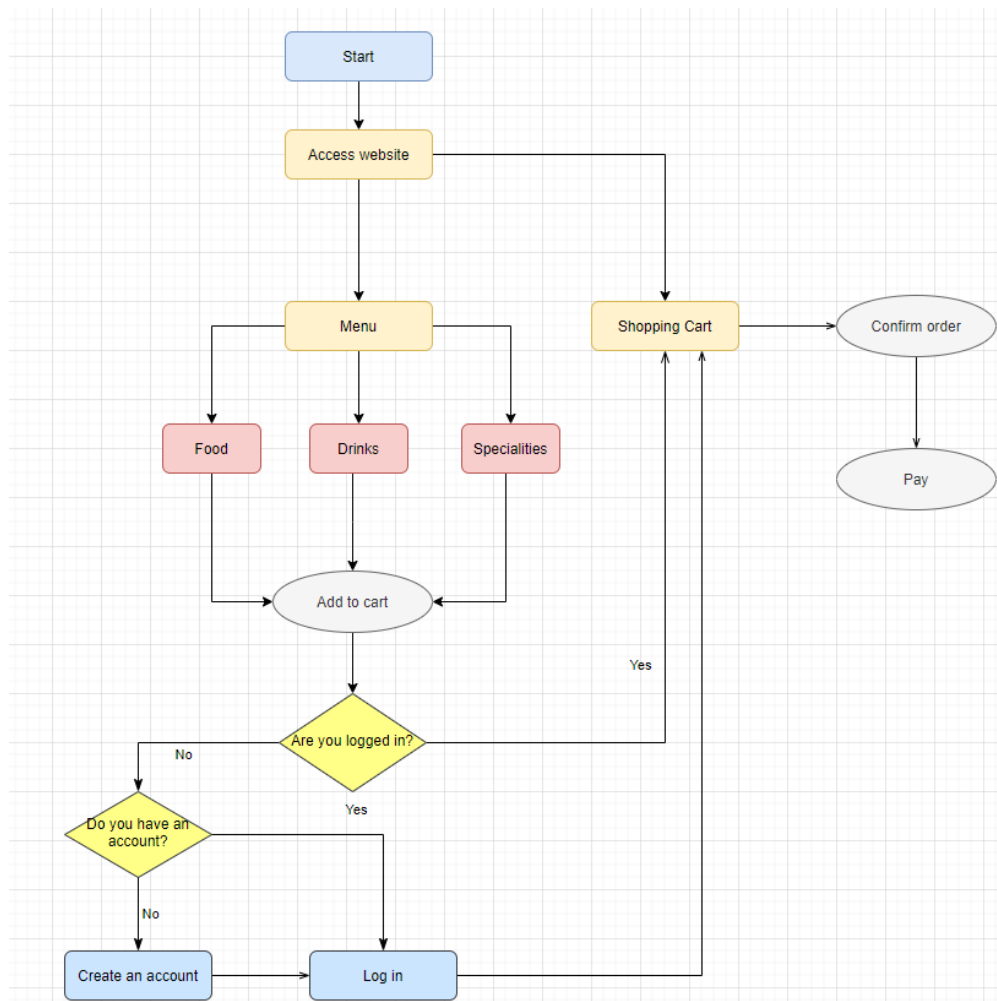


Fig. 28 Flow-ul aplicației

Am încercat să fac flow-ul cât mai simplu pentru a nu pune în dificultate clienții restaurantului pentru că ei pot fi de toate vârstele și poate nu este la fel de ușor pentru toată lumea.

Flow-ul este simplu:

Clientul intră în restaurant, accesează un link aflat pe masă (în dezvoltări ulterioare, va fi folosit un cod QR aflat pe masă) după care este trimis la aplicația frontend unde va avea acces la meniu. Dacă totul este ok iar clientul decide că vrea să rămână, el va trebui să se înregistreze în cazul în care nu deține deja un cont, sau să se autentifice în cazul în care are un cont. În cazul în care clientul nu este autentificat, el nu va putea adăuga produse în coș.

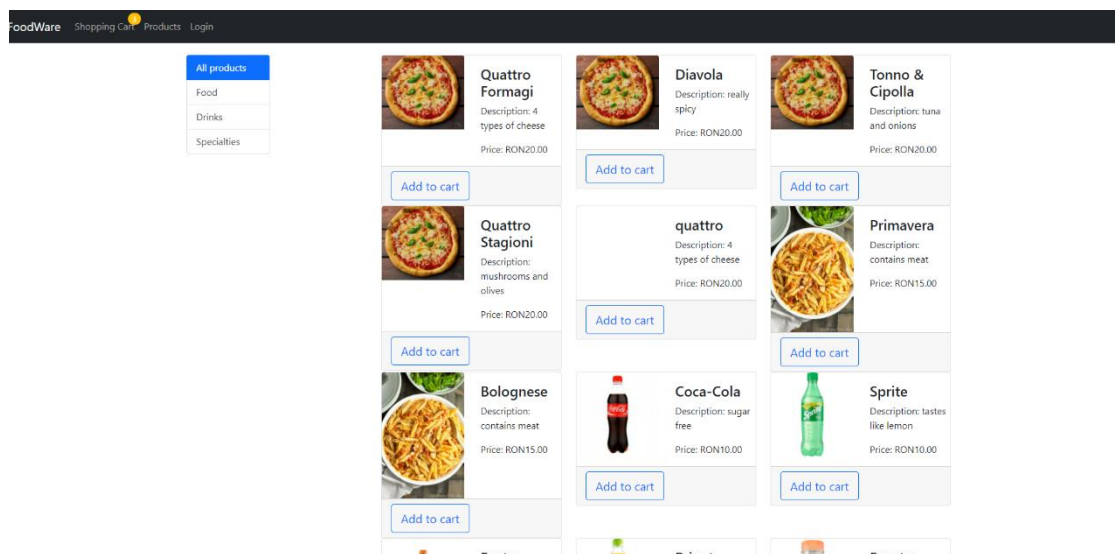


Fig. 29 Pagina principală a aplicației pe partea de client

Pagina principală conține meniul și este accesibilă oricui, chiar dacă nu este logat. Aici putem vedea toate produsele disponibile în cadrul restaurantului în care ne aflăm, produse care pot fi sortate în funcție de grupul („Food”, „Drinks”) și categoria („Pizza”, „Pasta”) din care fac parte.

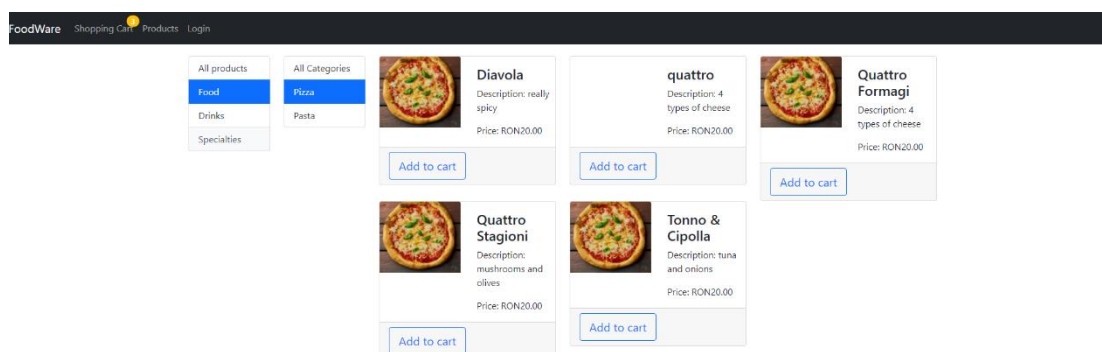


Fig. 30 Exemplu de meniu sortat după grupul „Food” și categoria „Pizza”

La apăsarea butonului de adăugare în coș, o fereastră pop-up va apărea pe ecran, solicitându-vă cantitatea dorită și, dacă este cazul, produse extra.

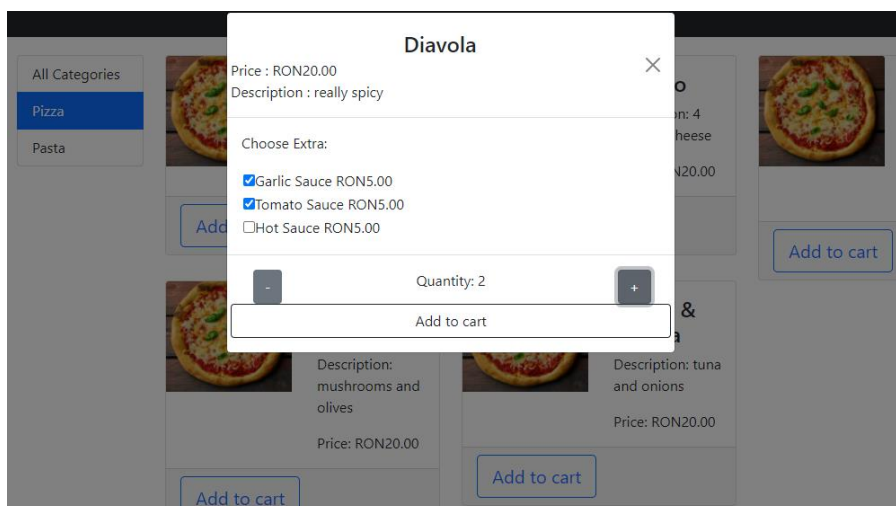


Fig. 31 Fereastra pop-up pentru alegerea cantității și produselor extra

După adăugarea produsului în coș, putem vizualiza pagina cu coșul de cumpărături. Această pagină conține toate produsele adăugate în coș și opțiunea de a le edita (schimbarea produselor extra), șterge sau modifica cantitatea.

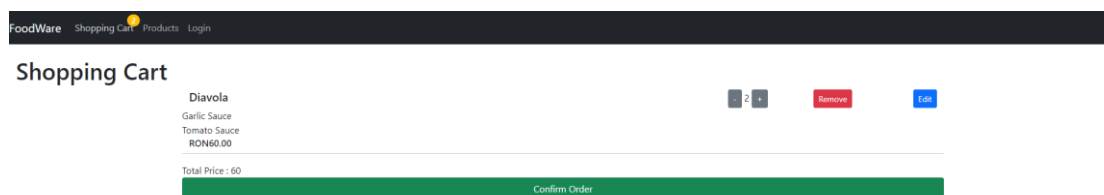


Fig. 32 Pagina pentru coșul de cumpărături

La finalizarea comenzii, după ce au fost adăugate toate produsele preferate în coș, se apasă butonul pentru confirmare comandă („Confirm Order”), moment în care produsele se trimit sub formă de subcomandă la backend, mai apoi putând fi afișate în pagina cu produsele grupului.



## **Capitolul IV. Concluzii**

Deși nu am reușit să testez aplicația într-un restaurant real, eu consider că folosirea acesteia ar reduce timpul de așteptare pentru clienți și ar crește câștigurile restaurantelor, clienții revenind la restaurant dacă consideră că totul a fost pe placul lor iar așteptarea nu a fost o problemă.

### **4.1 Dezvoltări ulterioare**

Unele din dezvoltările ulterioare ale aplicațiilor ar presupune adăugarea suportului pentru accesul aplicației cu ajutorul unui cod QR, îmbunătățirea interfeței cu utilizatorul ( o mai bună stilizare) și crearea unei metode pentru plata online.

Acest set de aplicații face parte dintr-un proiect mult mai amplu din care mai face parte și partea de backoffice (aplicațiile folosite de angajații restaurantului), acesta fiind următorul pas în implementarea proiectului.

Și în cele din urmă, pe viitor, aplicația web s-ar putea transforma în aplicații dedicate pentru mobil.

## Capitolul V. Bibliografie

- [1] E. Gesteiro, A. García-Carro, R. Aparicio-Ugarriza, and M. González-Gross, “Eating out of Home: Influence on Nutrition, Health, and Policies: A Scoping Review,” *Nutrients*, vol. 14, no. 6, Art. no. 6, Jan. 2022, doi: 10.3390/nu14061265.
- [2] C.-T. Ting, Y.-S. Huang, C.-T. Lin, and S.-C. Pan, “Evaluation of Consumers’ WTP for Service Recovery in Restaurants: Waiting Time Perspective,” *Adm. Sci.*, vol. 9, no. 3, Art. no. 3, Sep. 2019, doi: 10.3390/admsci9030063.
- [3] J. De Vries, D. Roy, and R. De Koster, “Worth the wait? How restaurant waiting time influences customer behavior and revenue,” *J. Oper. Manag.*, May 2018, doi: 10.1016/j.jom.2018.05.001.
- [4] “Sistem de comenzi online gratuit pentru restaurante.” <https://www.gloriafood.com/ro> (accessed Jun. 23, 2022).
- [5] A. Arora, “Table Reservation and Meal Ordering System Using QR Code,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 9, pp. 1387–1390, Jul. 2021, doi: 10.22214/ijraset.2021.36492.
- [6] “Differences between Java EE and Java SE - Your First Cup: An Introduction to the Java EE Platform.” <https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html> (accessed Jun. 18, 2022).
- [7] “Swagger Editor.” <https://editor.swagger.io/> (accessed Jun. 18, 2022).
- [8] “REST API Documentation Tool | Swagger UI.” <https://swagger.io/tools/swagger-ui/> (accessed Jun. 18, 2022).
- [9] “Spring Framework - Overview.” [https://www.tutorialspoint.com/spring/spring\\_overview.htm](https://www.tutorialspoint.com/spring/spring_overview.htm) (accessed Jun. 18, 2022).
- [10] “Spring Boot - Introduction.” [https://www.tutorialspoint.com/spring\\_boot/spring\\_boot\\_introduction.htm](https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm) (accessed Jun. 18, 2022).
- [11] “Cum Se Efectuează Operații CRUD în Angular | Routech,” Jan. 21, 2021. <https://www.routech.ro/cum-se-efectueaza-operatii-crud-in-angular/> (accessed Jun. 18, 2022).
- [12] “JavaScript — Dynamic client-side scripting - Learn web development | MDN.” <https://developer.mozilla.org/en-US/docs/Learn/JavaScript> (accessed Jun. 18, 2022).

- [13] “Why you should use TypeScript for your next project,” *P&C Blog*, Oct. 20, 2016. <https://blog.priceandcost.com/development/why-you-should-use-typescript-for-your-next-project/> (accessed Jun. 18, 2022).
- [14] “HTML basics - Learn web development | MDN.” [https://developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/HTML\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics) (accessed Jun. 18, 2022).
- [15] “CSS first steps overview - Learn web development | MDN.” [https://developer.mozilla.org/en-US/docs/Learn/CSS/First\\_steps](https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps) (accessed Jun. 18, 2022).
- [16] A. C. Team, “Angular Material,” *Angular Material*. <https://material.angular.io/> (accessed Jun. 19, 2022).
- [17] “How to use Redux in an Angular application,” Jun. 19, 2019. <https://assist-software.net/blog/how-use-redux-angular-application> (accessed Jun. 18, 2022).
- [18] M. O. contributors Jacob Thornton, and Bootstrap, “Get started with Bootstrap.” <https://getbootstrap.com/docs/5.2/getting-started/introduction/> (accessed Jun. 19, 2022).
- [19] “Postman Tutorial for Beginners to Perform API Testing,” *Encora*. <https://www.encora.com/insights/what-is-postman-api-test> (accessed Jun. 18, 2022).
- [20] “Consuming Async REST APIs with Spring WebClient - GET, PUT, POST, DELETE Examples,” *HowToDoInJava*, Mar. 24, 2020. <https://howtodoinjava.com/spring-webflux/webclient-get-post-example/> (accessed Jun. 18, 2022).
- [21] “How To Implement Feign Client In Spring Boot Microservices? | Making Java Easy To Learn,” Jul. 29, 2021. <https://javatechonline.com/how-to-implement-feign-client-in-spring-boot-microservices/> (accessed Jun. 18, 2022).
- [22] “MySQL :: MySQL 5.7 Reference Manual :: 1.2.1 What is MySQL?” <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html> (accessed Jun. 18, 2022).
- [23] “MySQL :: MySQL Workbench.” <https://www.mysql.com/products/workbench/> (accessed Jun. 19, 2022).
- [24] “What are the DDL commands in DBMS?” <https://www.tutorialspoint.com/what-are-the-ddl-commands-in-dbms> (accessed Jun. 18, 2022).
- [25] “Database Design - DDL & DML.” <https://web.csulb.edu/colleges/coe/cecs/dbdesign/dbdesign.php?page=sql/ddldml.php> (accessed Jun. 18, 2022).

- [26] “Introduction to Project Lombok in Java and How to get started?,” *GeeksforGeeks*, Jan. 17, 2020. <https://www.geeksforgeeks.org/introduction-to-project-lombok-in-java-and-how-to-get-started/> (accessed Jun. 18, 2022).
- [27] “Git vs Github: What’s the Difference and How to Get Started with Both.” <https://kinsta.com/knowledgebase/git-vs-github/> (accessed Jun. 18, 2022).
- [28] “IntelliJ IDEA overview | IntelliJ IDEA.” <https://www.jetbrains.com/help/idea/discover-intellij-idea.html> (accessed Jun. 18, 2022).
- [29] “Trello 101: How to Use Trello Boards & Cards | Trello.” <https://trello.com/guide/trello-101> (accessed Jun. 18, 2022).
- [30] Atlassian, “Confluence | Your Remote-Friendly Team Workspace,” *Atlassian*. <https://www.atlassian.com/software/confluence> (accessed Jun. 19, 2022).