

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

Gestionarea recenziilor pentru restaurantele din Iași

propusă de

Ioana-Loredana Teodorescu

Sesiunea: *iulie, 2017*

Coordonator științific
Conf. dr. Iftene Adrian

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI
FACULTATEA DE INFORMATICĂ

Gestionarea recenziilor pentru restaurantele din Iași

Ioana-Loredana Teodorescu

Sesiunea: *iulie, 2017*

Coordonator științific
Conf. dr. Iftene Adrian

DECLARAȚIE PRIVIND ORIGINALITATE ȘI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licență cu titlul „Gestionarea recenziilor pentru restaurantele din Iași” este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau din străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referința precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
- codul sursă, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
- rezumarea ideilor altor autori precizează referința precisă la textul original.

Iași,

Absolvent Ioana-Loredana Teodorescu

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „Gestionarea recenziilor pentru restaurantele din Iași”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași,

Absolvent Ioana-Loredana Teodorescu

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări, Ioana-Loredana Teodorescu.

Încheierea acestui acord este necesară din următoarele motive:

Pe viitor vreau să dezvolt aplicația și să o fac publică și utilizabilă de către oricine.

Iași,

Decan Adrian Iftene

Absolvent Ioana-Loredana Teodorescu

Cuprins

Introducere	6
I. Soluții similare	7
A. Google Maps	7
B. TripAdvisor	10
C. BoogIt	13
II. Tehnologii utilizate	17
A. MySQL	17
B. JavaScript	21
C. Node.js	22
D. HTML	24
E. CSS	27
F. SASS	29
G. ReactJS	32
III. Soluția propusă	37
A. Arhitectura aplicației	38
1. Baza de date	39
2. Server-ul	42
3. Clientul	43
a. Componenta Restaurante	43
b. Componenta Hartă	46
c. Componenta Recenzii	47
IV. Concluzii	50
Bibliografie	51

Introducere

De multe ori am fost pusă în fața unor situații în care eram într-o anumită locație și nu știam unde să merg pentru a lua masa sau pentru a servi o băutură rece. În calitate de student venit dintr-un alt oraș, îmi era foarte greu să găsesc un restaurant acceptabil ca preț, dar și cu mâncare gustoasă și bună. În același timp, pentru a petrece pauzele dintre cursurile de la facultate într-un loc în care puteam să învăț, să îmi fac o temă sau pur și simplu să mă relaxez, aveam nevoie de un restaurant destinat în special studenților. Desigur că am întrebat colegi de la facultate care erau în ani mai mari decât mine, precum și alți prieteni care erau de mai multă vreme în Iași, însă răspunsurile și recomandările lor au fost prea puține și irelevante. Cu timpul, mergând în mai multe restaurante pentru a le găsi pe cele mai bune, am descoperit mai multe locații care mi-au îndeplinit cerințele. A fost nevoie de timp liber, precum și de bani cheltuiți pentru a achita nota de plată. Cum pentru majoritatea studenților banii nu sunt o problemă, ci numărul mic al lor, am decis să creez aplicația *Recenzii restaurante Iași* care să îi ajute mai ales pe studenții noi veniți. Aplicația este destinată tuturor persoanelor, dar, întrucât aceasta oferă informații despre restaurantele din jurul Facultății de Informatică, este destinată mai ales studenților de la această facultate.

Cap. I Soluții similare

Aplicații precum *Google Maps*, *Booking.com* și *Trivago.ro* sunt soluții similare aplicației *Recenzii restaurante Iași*, oferă deja posibilitatea scrierii unei recenzii, însă procesul acesta este puțin mai anevoios deoarece ele oferă și alte servicii în afară de scriere de recenzii.

A. Google Maps

*Google Maps*¹ este o aplicație gratuită disponibilă atât pe desktop, cât și pe mobil. Ea a apărut la sfârșitul anului 2004 când *Google*² a cumpărat compania *Where 2 Technologies*³. Printre opțiunile acestei aplicații se numără următoarele:

- se pot vedea șoselele de pe tot glodul: în modul satelit care arată toate elementele existente la o anumită locație, precum și în modul hartă care oferă imaginea schematică a străzilor;
- se pot vedea informații despre trafic în acel moment al folosirii aplicației;
- se poate vedea distanța de la un punct la altul ales de utilizator;
- vizualizarea de imagini dintr-o anumită locație.

Pe lângă aceste opțiuni, *Google Maps* are și opțiunea de a scrie recenzii pentru un anumit local. După cum se poate vedea și în Fig. I.1, în momentul în care un utilizator caută informații despre un restaurant i se oferă detalii despre adresa acestuia, numărul de telefon la care poate fi contactat sau orarul de funcționare.

¹ <https://www.google.ro/maps>

² <https://www.google.ro/>

³ [https://en.wikipedia.org/wiki/Lars_Rasmussen_\(software_developer\)](https://en.wikipedia.org/wiki/Lars_Rasmussen_(software_developer))

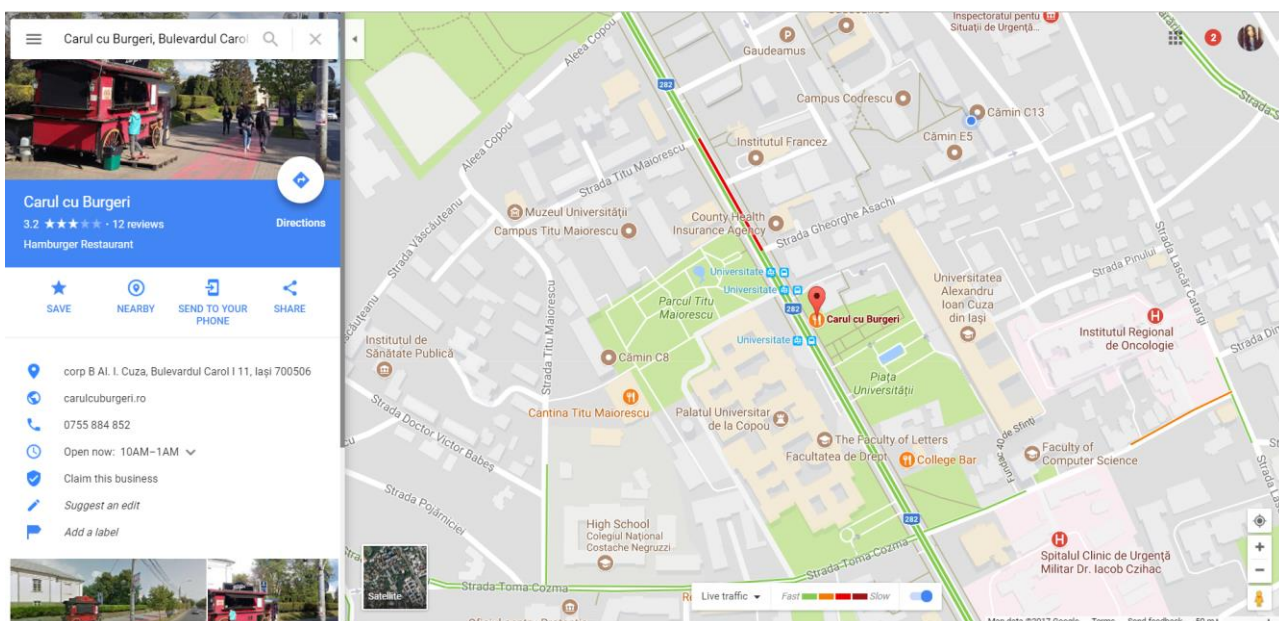


Fig. I.1 Google Maps - Informații despre „Carul cu Burgeri”

După cum se poate vedea în Fig. I.2 și Fig. I.3, pentru a scrie o recenzie, utilizatorul are la dispoziție opțiunea *Write a review* (ro. Scrie o recenzie), prin accesarea căruia se deschide un modal ce conține un câmp pentru textul recenziei, stelutele pentru a alege cât de mulțumitoare a fost, pe o scară de la 1 la 5, experiența în acel local, precum și posibilitatea adăugării unei fotografii.

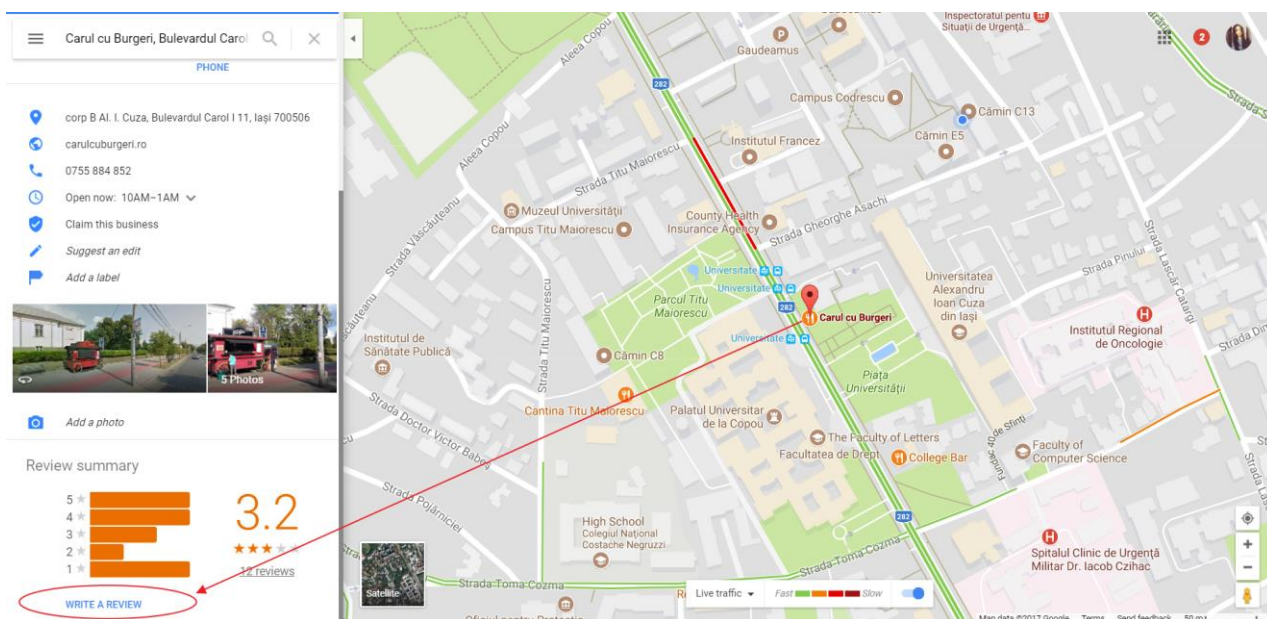


Fig. I.2 Google Maps - Opțiunea pentru scrierea unei recenzii

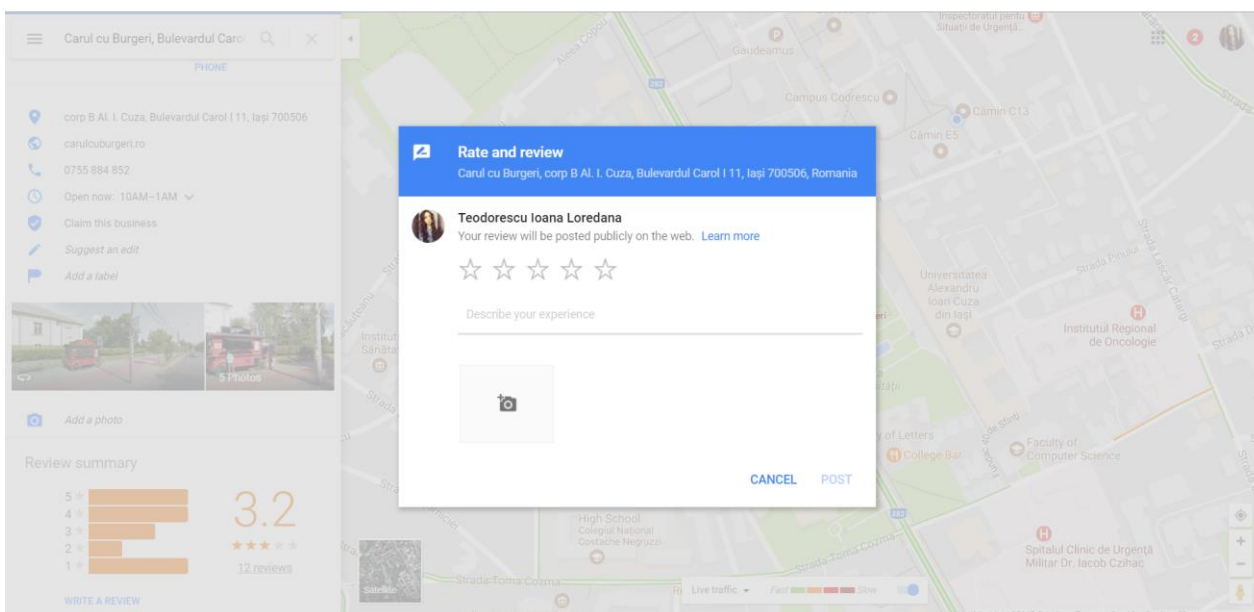


Fig. I.3 Google Maps - Scrierea unei recenzii

În cazul în care o persoană nu vrea să scrie o recenzie, ea are posibilitatea doar de a citi recenziile celorlalți utilizatori și de a vedea media tuturor recenziilor, după cum se poate vedea în Fig. I.4.

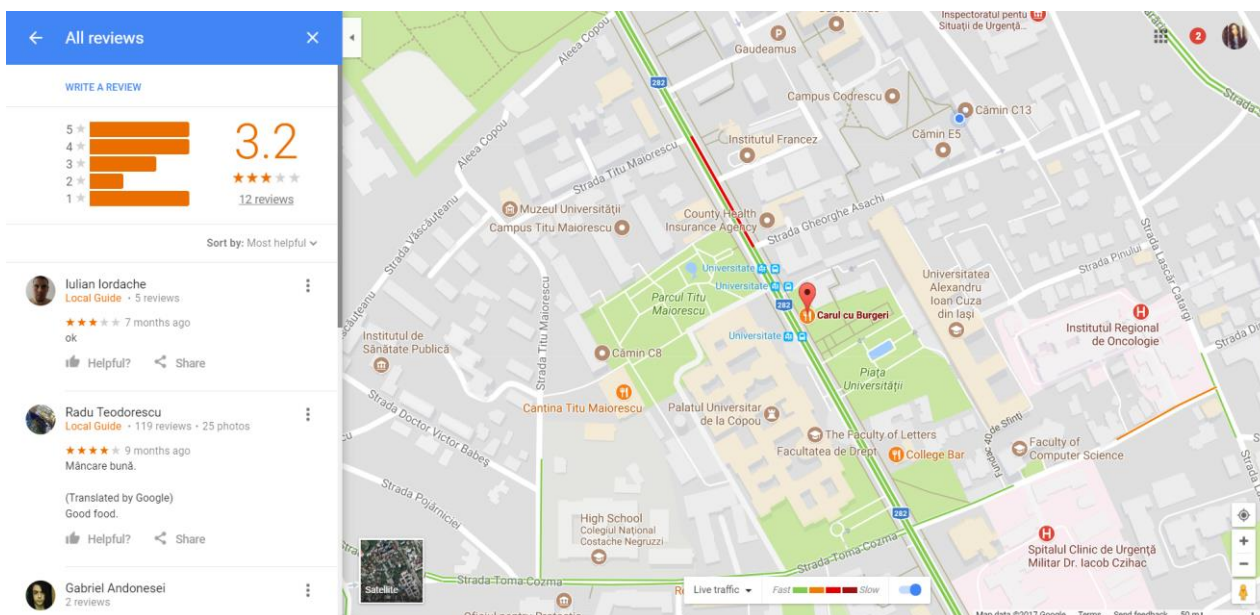


Fig. I.4 Google Maps - Vizualizarea recenziilor

După părerea mea, *Google Maps* este o aplicație complexă, care oferă o multitudine de opțiuni, cu ajutorul căreia un utilizator obține informații exacte și actualizate. Este foarte practică și utilă mai ales pentru că are versiunea pentru desktop, dar și pentru mobil. În ceea ce privește recenziile, experiența folosirii acestei aplicații este una plăcută întrucât pentru a scrie sau vizualiza o recenzie nu este nevoie de foarte mult efort, fiind simplă de utilizat pentru orice utilizator care știe să folosească la nivel minimal un calculator și navigarea pe internet.

B. TripAdvisor

*TripAdvisor*⁴ reprezintă o aplicație ce pune la dispoziția utilizatorilor informații despre locațiile din toată lumea. După cum se poate vedea în Fig. I.5, prin intermediul *TripAdvisor* se pot afla informații despre hotelurile din toată lumea, informații precum disponibilitatea unui camere dintr-un hotel în funcție de ceea ce se caută, oferta de preț împreună cu reducerile ce apar, facilitățile oferite de acel hotel și părerile persoanelor ce au fost cazați cel puțin o zi.

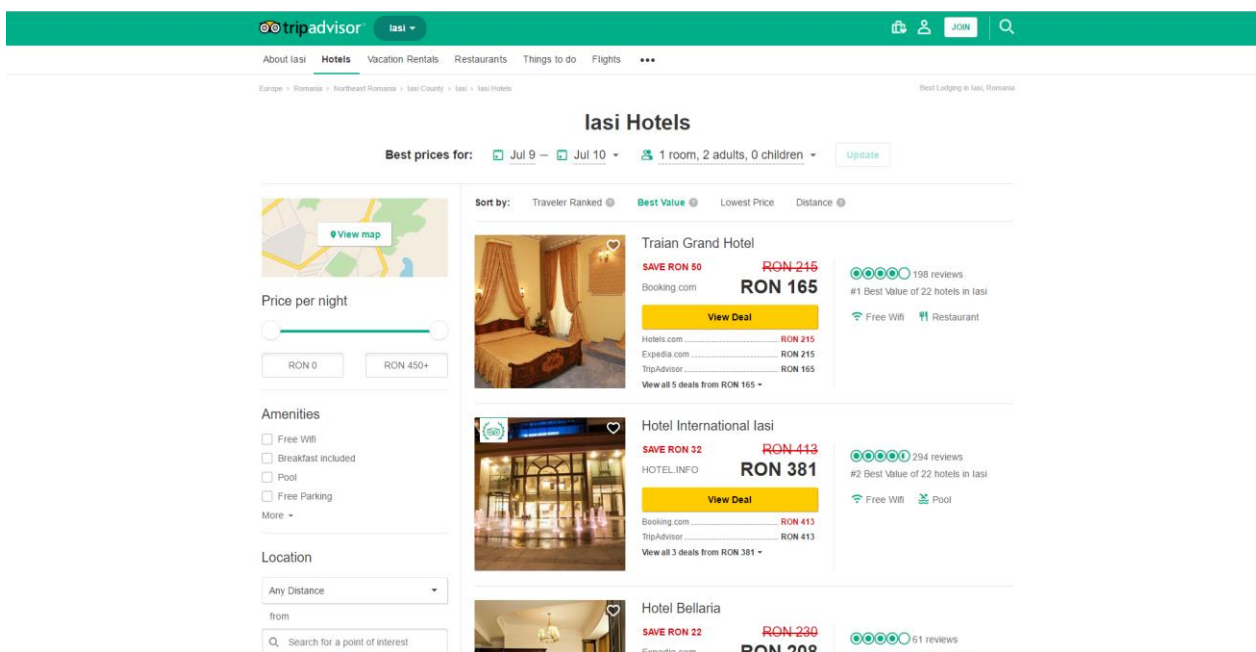


Fig. I.5 TripAdvisor - Informații despre hotelurile din Iași

În același timp această aplicație afișează informații despre locațiile care oferă posibilitatea cazării pe o perioadă mai mare de timp, cum ar fi pe perioada unei vacanțe. După cum se poate vedea în Fig. I.6, pentru aceasta se fac oferte de preț ce sunt afișate la cererea utilizatorului, după completarea câmpurilor pentru perioada de cazare dorită.

⁴ <https://www.tripadvisor.com/>

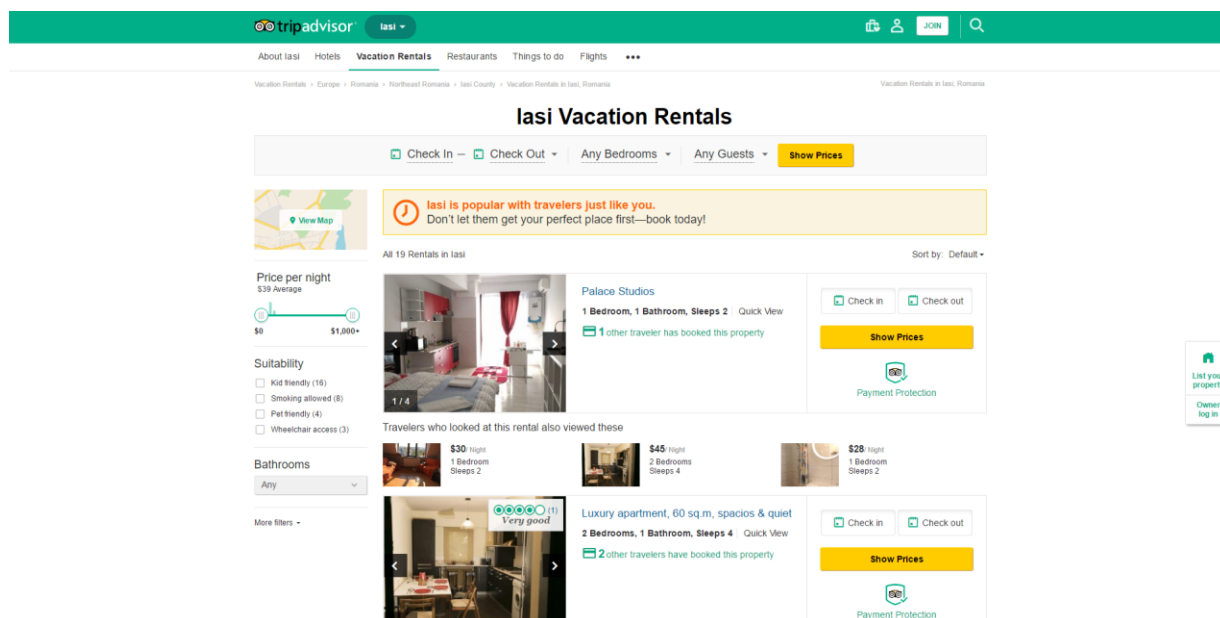


Fig. I.6 TripAdvisor - Posibilități de cazare în Iași

Totodată, după cum se poate vedea în Fig. I.7 și Fig. I.8, *TripAdvisor* oferă utilizatorului posibilitatea de a căuta zboruri către o anumită locație în perioada dorită, dar și de a vizualiza cele mai importate obiective de vizitat din orașul căutat.

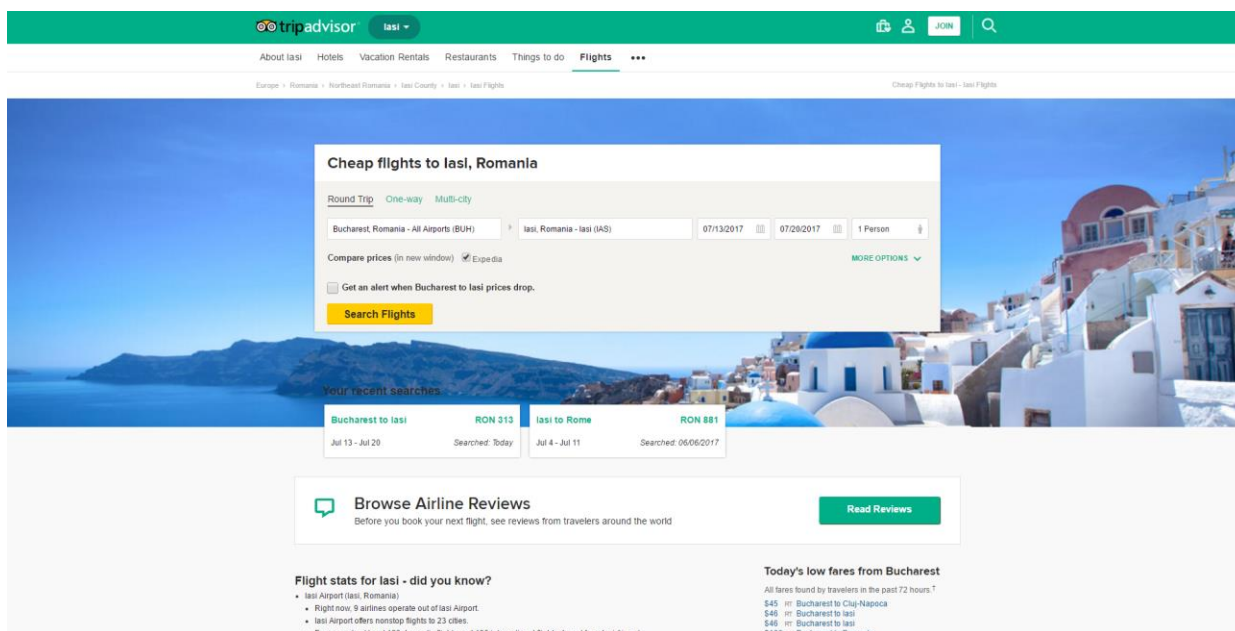


Fig. I.7 TripAdvisor - Vizualizarea zborurilor disponibile

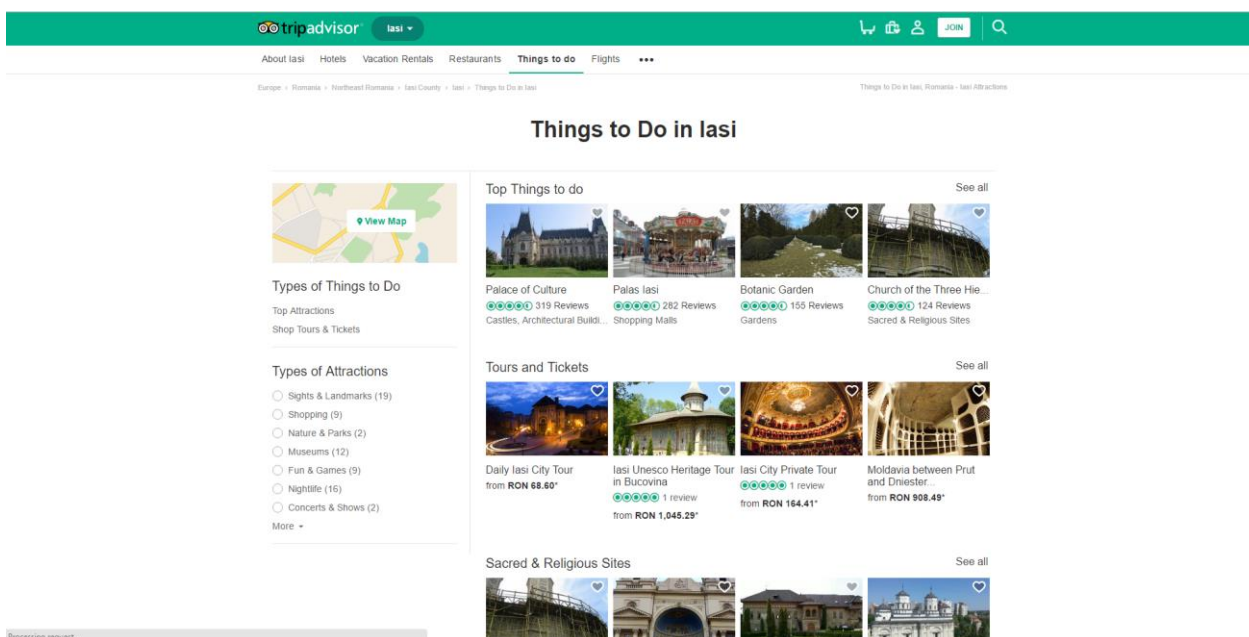


Fig. I.8 TripAdvisor - Vizualizarea obiectivelor turistice

După cum se poate vedea în Fig. I.9, aplicația permite vizualizarea de informații și despre restaurante, la care afișează link-ul către pagina restaurantului, premiile pe care le-a obținut la concursuri, programul de funcționare, adresa unde este amplasat, fotografii din acel local, locația pe hartă, precum și recenziiile date de clienți.

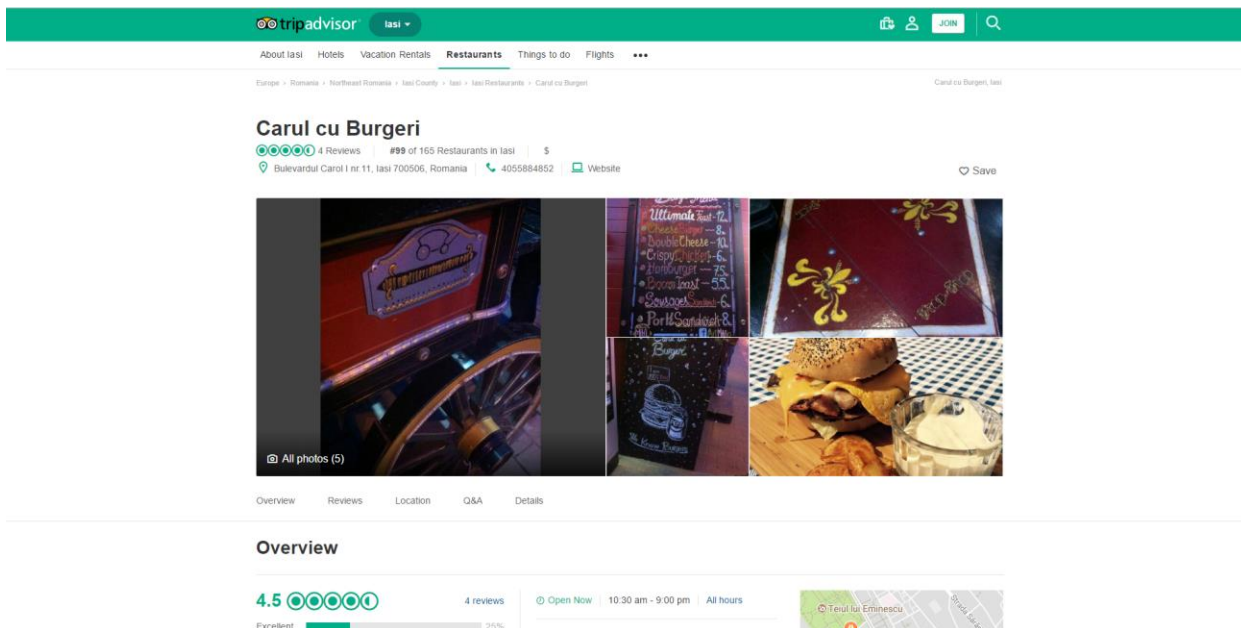


Fig. I.9 TripAdvisor - Vizualizare informații despre restaurante

După cum se poate vedea în Fig. I.10, aceste recenzii sunt foarte bine clasificate: în funcție de servire, atmosferă, preț și calitatea mâncării. Totodată, recenziiile sunt procesate și ordonate în funcție de numărul de puncte oferite de utilizator.

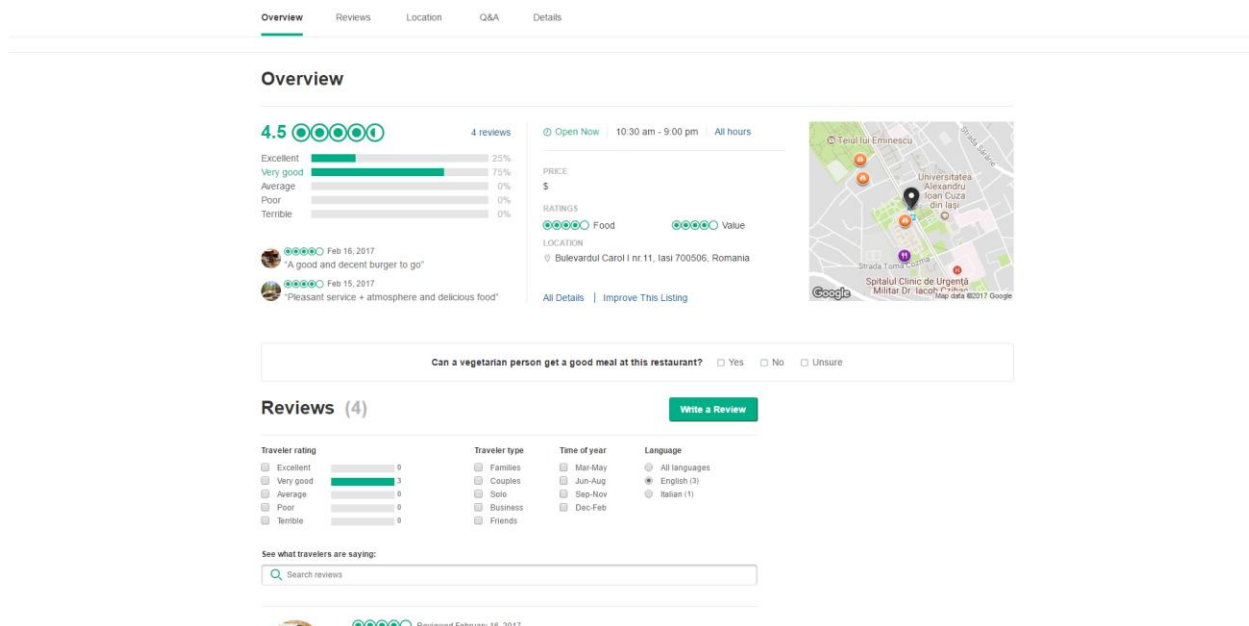


Fig. I.10 - TripAdvisor - Vizualizarea recenziilor

După părerea mea, *TripAdvisor* este o aplicație foarte utilă, mai ales când o persoană este într-un oraș necunoscut și vrea să își petreacă timpul vizitând cât mai multe obiective turistice sau încercând noi mâncăruri în restaurantele cu specific căutate. Datorită posibilității scrierii de recenzii privind călătoriile din toată lumea, un utilizator poate afla informații din punctul de vedere al unui client, și nu din punctul de vedere al angajaților care pot fi subiectivi. Deoarece este foarte practică, *TripAdvisor* este disponibilă atât pe desktop, cât și pe mobil.

C. BoogIt

*BoogIt*⁵ este o aplicație disponibilă atât pe desktop cât și pe mobil. Cu ajutorul ei, orice utilizator poate organiza simplu și rapid un eveniment cum ar fi o nuntă, un botez, o aniversare sau orice alt eveniment ce necesită mult efort pentru a reuși rezervarea unei săli. Aplicația a fost creată de un grup de tineri din orașul Brașov, iar, după cum a spus într-un interviu unul dintre membri, „ideea a venit după ce unul dintre membrii fondatori a vrut să-și organizeze un eveniment și a întâmpinat numeroase dificultăți, printre care lipsa de furnizori, lipsa de date de

⁵ <https://www.boogit.ro/>

*contact și i-a fost foarte dificil să-ți găsească ce-i trebuia.”*⁶. Astfel, după cum se poate observa în Fig. I.11, aplicația permite căutarea de restaurante, fotografi, formație muzicală, florărie, precum și alte servicii necesare creării unui eveniment reușit, iar căutarea se poate face pentru orice oraș din România.

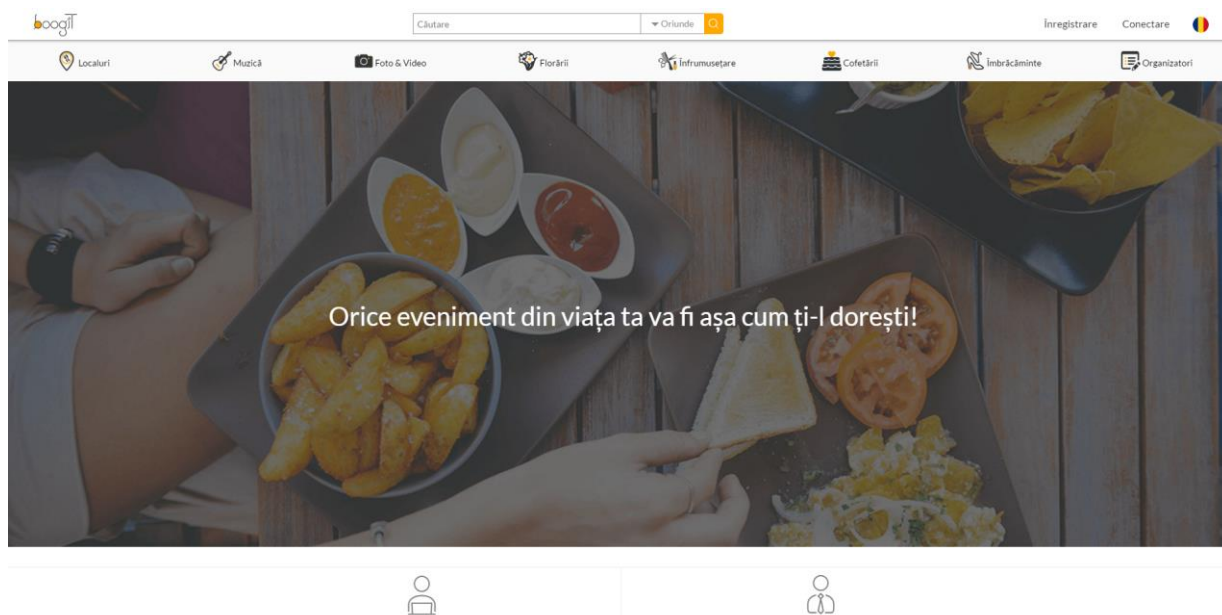


Fig. I.11 BoogIt - Pagina principală

În Fig. I.12 se observă că pentru un serviciu ales din meniu se afișează o listă cu furnizorii acestuia. În ceea ce privește partea de localuri, se afișează toate tipurile: cafenea, restaurant, club sau pizzerie.

⁶ <http://www.brasovultau.ro/articol/stiri/aplicatie-unica-in-lume-creata-de-brasoveni.html>

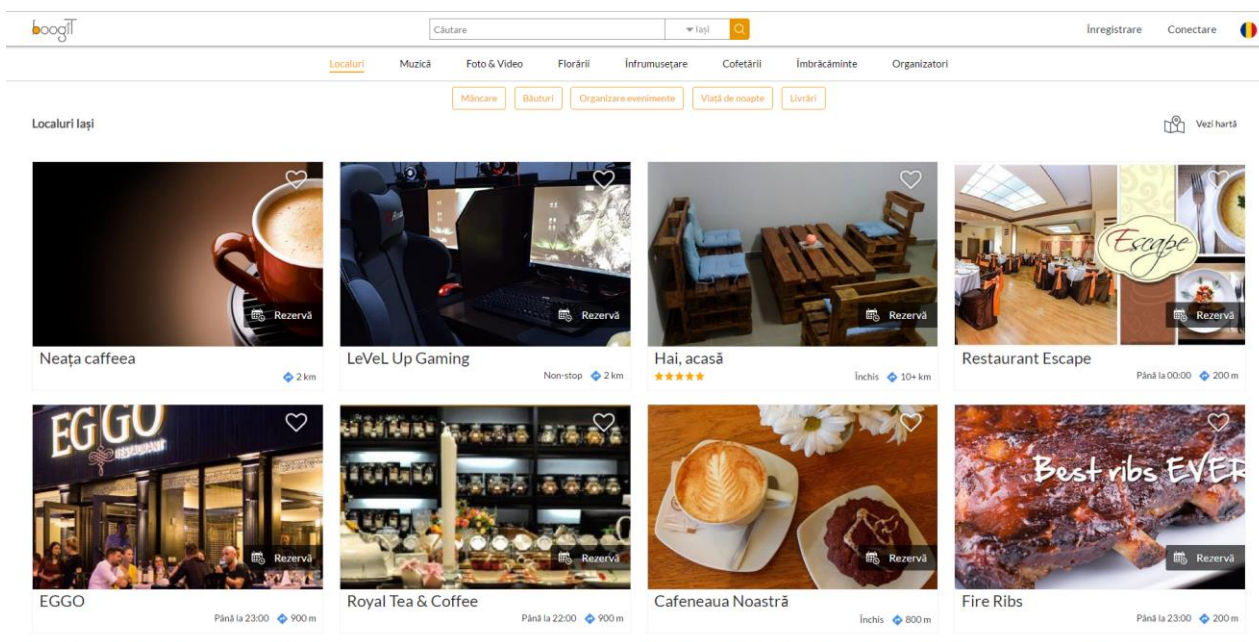


Fig. I.12 BoogIt - Afișarea furnizorilor de servicii

Pentru fiecare local se oferă informații despre adresa unde este amplasat precum și poziția lui pe hartă, programul de funcționare, fotografiile sau numărul de telefon la care poate fi contactată persoana responsabilă de rezervări. *BoogIt* aduce o funcționalitate nouă față de alte aplicații: posibilitatea rezervării online. Pentru a putea rezerva online un utilizator are nevoie de un cont creat în aplicație, iar apoi este necesară completarea unui mic formular cu date specifice unei rezervări. Aceste informații se pot vedea în Fig. I.13.

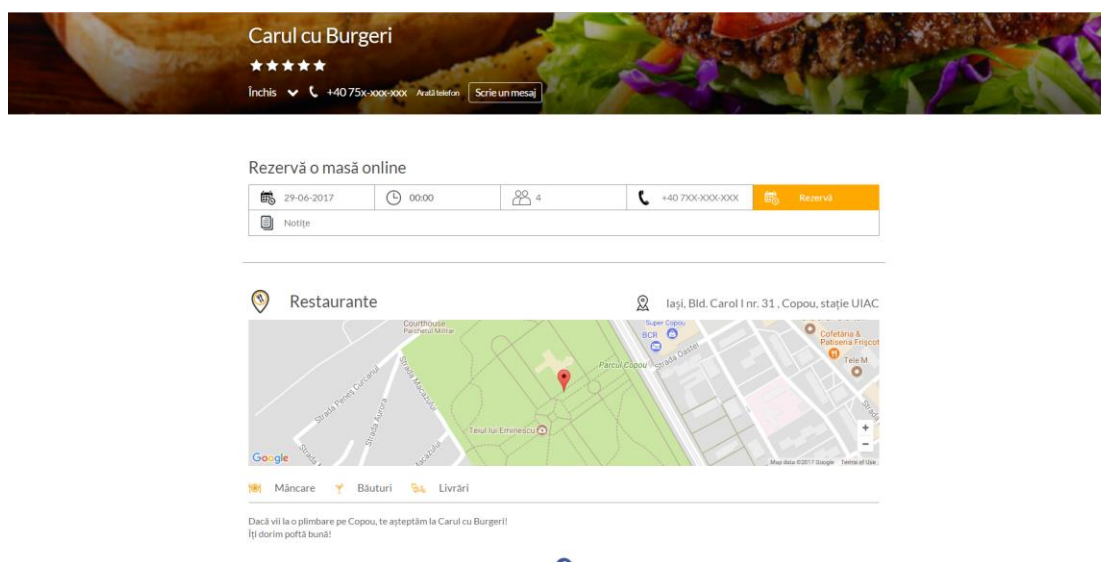


Fig. I.13 BoogIt - Informații despre “Carul cu Burgeri”

Totodată, *BoogIt* oferă unui utilizator autentificat posibilitatea de a împărtăși și altora experiența lui într-un local. Pentru aceasta, după cum se poate vedea în Fig. I.14, este disponibilă opțiunea *Scrie o recenzie*, în urma căreia părerile utilizatorului vor fi afișate în secțiunea de *Recenzii* pentru a fi citite și de alți utilizatori care vor doar să caute un local pe gustul fiecăruia.

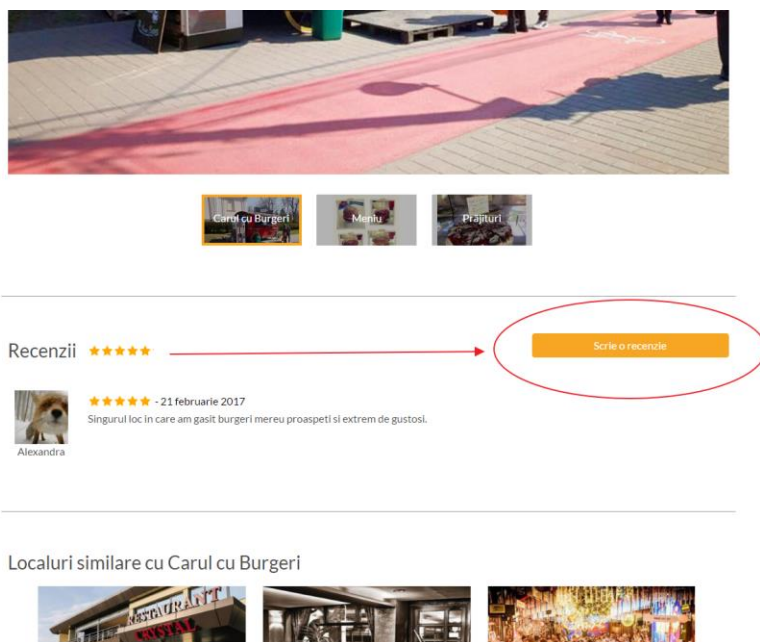


Fig. I.14 BoogIt - Scrierea unei recenzii

Fie că persoana ce folosește aplicația este un utilizator obișnuit ce dorește organizarea unui eveniment sau este un furnizor de servicii, *BoogIt* este adaptată pentru a putea fi folosită de oricine, este ușoară de utilizat și oferă informații concrete, fără a încărca pagina.

Cap. II Tehnologii utilizate

A. MySQL

Conform documentației oficiale, „MySQL este un sistem de gestionare a bazelor de date relaționale”, componentă integrată a platformelor XAMPP sau WAMP.

- O bază de date reprezintă o colecție de date structurate;
- O bază de date relațională nu stochează toate datele într-un singur fișier, ci se creează tabele separate în care se memorează datele. Aceste tabele sunt legate între ele prin relații. Astfel se adaugă viteză și flexibilitate;
- XAMPP și WAMP sunt aplicații ce conțin mai multe servicii cum ar fi severe MySQL. Ele pot fi folosite atât local, cât și pe servere web.

Conform site-ului oficial, MySQL este „cel mai cuprinzător set de caracteristici avansate, instrumente de gestionare și suport tehnic pentru a atinge cele mai înalte niveluri de scalabilitate, securitate, fiabilitate și timp de funcționare.”⁷. Astfel, MySQL permite stocarea, căutarea, sortarea și găsirea datelor în mod eficient. Serverul MySQL controlează accesul la date pentru a garanta că mai mulți utilizatori pot lucra simultan cu acestea. Așadar, MySQL este un server multi-user (mai mulți utilizatori) și multi-thread (mai multe fire de execuție)⁸.

O bază de date MySQL conține tabele, în care există rânduri și coloane. Pentru fiecare celulă de la intersecția fiecărui rând cu coloană există o informație de un anumit tip: *int*, *varchar*, *date*. O structură de ansamblu a unei baze de date MySQL este prezentată în Fig. II.1.

⁷ <https://www.mysql.com/>

⁸ http://www.cursuri-online.info/php_mysql/CeestePHPsiMySQL.htm

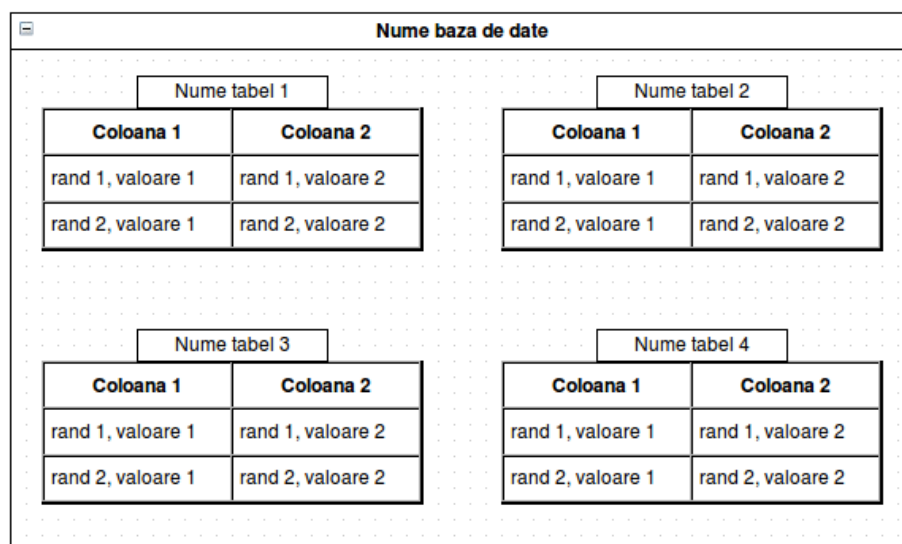


Fig. II.1 Structura unei baze de date MySQL⁹

Fiecare bază de date conține relații între tabele. Există 3 tipuri de relații:

- *unu-la-unu* (*en. one-to-one*): unui rând dintr-un tabel îi corespunde un singur rând din alt tabel;
- *unu-la-mai-mulți* (*en. one-to-many*): unui rând dintr-un tabel îi corespund mai multe rânduri din alt tabel;
- *mai-mulți-la-mai-mulți* (*en. many-to-many*): mai multor rânduri dintr-un tabel le corespund mai multe rânduri din alt tabel.

O bază de date poate fi reprezentată schematic printr-o diagramă UML (limbaj vizual de modelare). Această schemă conține tabelele existente în baza de date, numele coloanelor existente în fiecare tabel, precum și relațiile dintre tabele. În Fig. II.2 este oferită structura unei baze de date modelată conform acestui limbaj.

⁹ <https://programam.ro/posts/view/notiuni-de-baza-in-lucrul-cu-mysql>

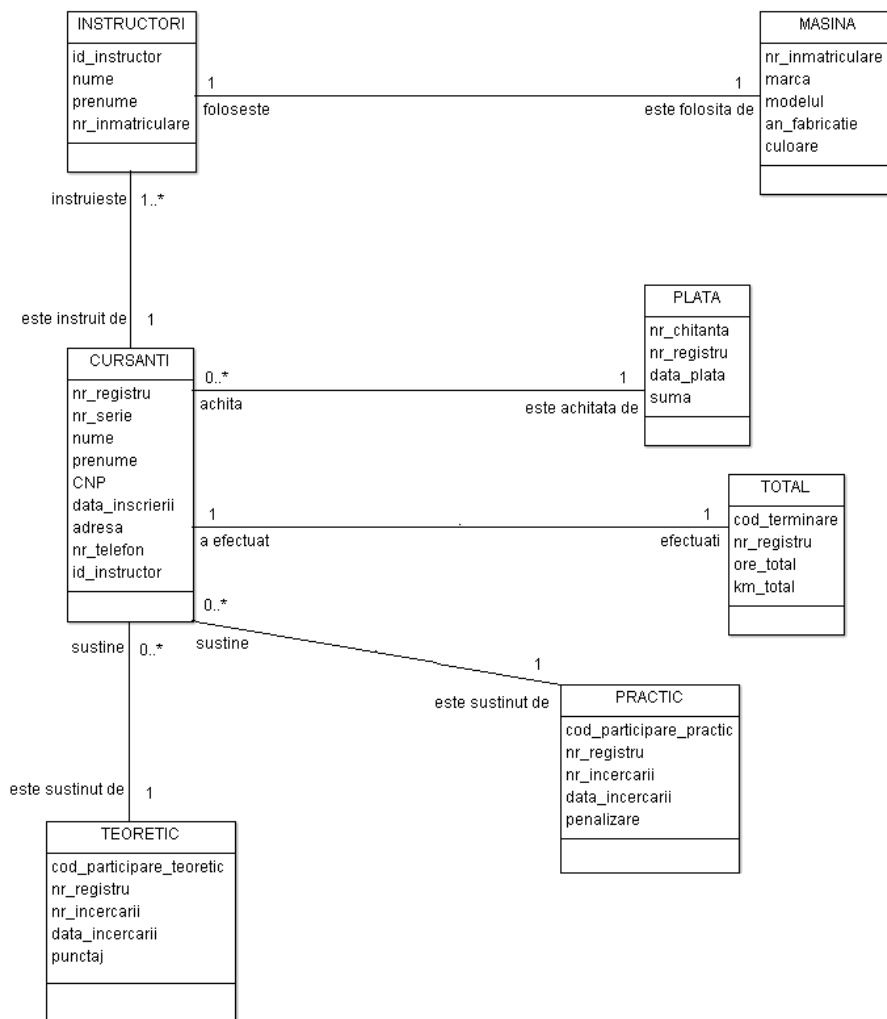


Fig. II.2 Exemplu de bază de date

MySQL utilizează limbajul SQL (en. Structured Query Language, ro. limbaj de interogare structurată), un limbaj neprocedural deoarece în cererea trimisă la server se specifică ce informații se doresc, și nu cum să fie obținute¹⁰. O cerere se realizează prin interogări. Utilizând limbajul SQL, conform documentației oficiale SQL¹¹, aceste interogări pot fi de mai multe tipuri, iar cele mai importante și mai des folosite sunt:

¹⁰ http://vega.unitbv.ro/~cataron/Courses/BD/BD_Cap_5.pdf

¹¹ <https://dev.mysql.com/doc/refman/5.7/en/sql-syntax.html>

- CREATE: pentru crearea de tabele (în Fig. II.3 este un exemplu de interogare pentru creare date);

```
CREATE TABLE note(  
    prenume VARCHAR(20),  
    nume VARCHAR(25),  
    nota INT(2)  
);
```

Fig. II.3 Crearea tabelii "Note"

- INSERT: pentru inserarea de date în tabelele din baza de date (în Fig. II.4 este un exemplu de interogare pentru inserare date);

```
INSERT INTO note(prenume,nume,nota) VALUES ('ana','popescu',10);
```

Fig. II.4 Înserarea unei înregistrări

- SELECT: pentru obținerea datelor din tabelele din baza de date (în Fig. II.5 este un exemplu de interogare pentru obținere de date);

```
SELECT nume,prenume FROM note WHERE nota=6;
```

Fig. II.5 Comanda select

- UPDATE: pentru a schimba date deja existente în tabelele din baza de date (în Fig. II.6 este un exemplu de interogare pentru schimbare de date);

```
UPDATE note SET nota=10 WHERE nota=5;
```

Fig. II.6 Actualizarea informațiilor din tabela Note

- DELETE: pentru a șterge date sau tabele din baza de date (în Fig. II.7 este un exemplu de interogare pentru ștergere de date);

```
DELETE FROM note WHERE nume='ana';
```

Fig. II.7 Ștergerea de înregistrări

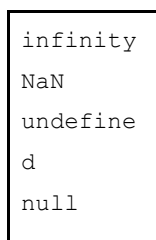
B. JavaScript

JavaScript este un limbaj de tip scripting destinat să opereze în mai multe sisteme de operare¹². Acesta poate fi folosit atât pe partea de server, cât și pe partea de client pentru că este ușor de integrat și utilizat.

Conform Cornel Mironel Niculae¹³, profesor la Facultatea de Fizică din cadrul Universității București, pentru partea de server *JavaScript* extinde nucleul limbajului furnizând obiecte pentru a rula programe *JavaScript* pe un server. Pentru partea de client *JavaScript* extinde nucleul limbajului furnizând obiecte pentru a controla un browser (Navigator sau un alt tip de browser) și modelul său de obiect document (*en. Document Object Model - DOM*).

JavaScript conține, la fel ca alte limbaje de programare, tipuri de date. O parte din acestea¹⁴ sunt:

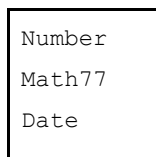
- valoare: acestea sunt simple valori și nu au proprietăți sau metode. În Fig. II.8 sunt exemplificate tipurile de date valoare;



```
infinity
NaN
undefined
d
null
```

Fig. II.8

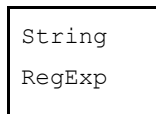
- număr și dată: acestea sunt obiecte ce reprezintă numere, date și calcule matematice. În Fig. II.9 sunt exemplificate tipurile de date număr și dată;



```
Number
Math77
Date
```

Fig. II.9

- pentru procesare de texte: În Fig. II.10 sunt exemplificate tipurile de date pentru procesare de texte;



```
String
RegExp
```

Fig. II.10

¹² https://developer.mozilla.org/ro/docs/Web/JavaScript/O_re-introducere_in_JavaScript

¹³ <http://fpce9.fizica.unibuc.ro/jsjava/>

¹⁴ <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>

La adresa <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference> se găsesc mai multe informații despre tipurile de date din *JavaScript*, precum și exemple detaliate pentru fiecare tip în parte.

C. Node.js

Node.js este un mediu de programare disponibil gratuit, ce permite dezvoltarea de aplicații web la nivel de server atât pentru Windows și OS X, cât și pentru Linux. Acesta folosește JavaScript ca limbaj de programare, iar o aplicație Node.js rulează într-un singur proces și tratează în mod asincron evenimentele de intrare/ieșire.¹⁵ În Fig. II.11 este prezentat schematic modul de tratare a acestor evenimente într-un web server obișnuit și într-un web server Node.js.

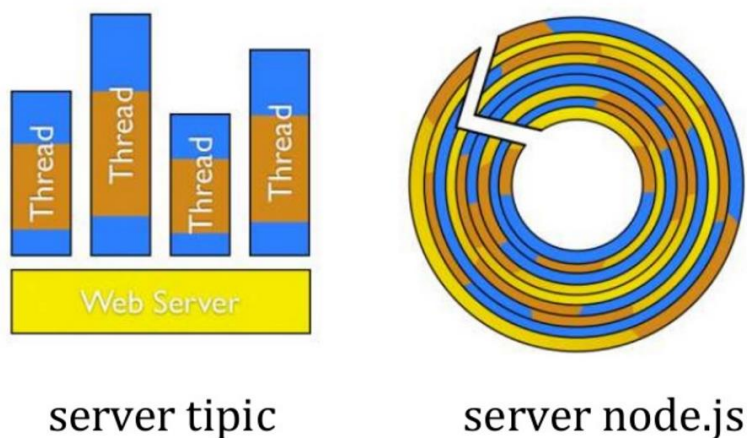


Fig. II.11 Modul de tratare a evenimentelor de intrare/ieșire¹⁶

Pentru a crea diverse aplicații, *Node.js* folosește module. Aceste module sunt considerate librării și se importă în proiect utilizând comanda `require`. Se pot folosi cele deja create din aplicație sau un utilizator își poate crea propriul modul. În Fig. II.12 este exemplificată comanda completă de importare a unui modul în Node.js.

```
var nume_folosit_in_program =  
require('nume_modul');
```

Fig. II.12 Comanda de importare a unui modul

Pentru a putea fi importat, în cazul în care nu este creat de utilizator, fiecare modul trebuie să fie instalat înainte cu ajutorul comenzii `npm`. În Fig. II.13 este exemplificată comanda completă de instalare a unui modul.

¹⁵ Slide 12 din <https://www.slideshare.net/busaco/nodejs-aspecte-esentiale>

¹⁶ Slide 14 din <https://www.slideshare.net/busaco/nodejs-aspecte-esentiale>

```
npm install nume_modul --  
save
```

Fig. II.13 Comanda de instalare a unui modul

Aceste comenzi se execută în linia de comandă după ce a fost instalat *Node.js* de pe site-ul oficial¹⁷, iar o listă cu câteva dintre modulele utile pentru crearea unei aplicații se găsesc la adresa <https://www.codementor.io/ashish1dev/list-of-useful-nodejs-modules-du107mcv3>.

Pentru a crea un server web în *Node.js*, fără a folosi nicio librărie externă, se poate folosi modulul `http`. El nu trebuie instalat înainte pentru că se instalează o dată cu *Node.js*, făcând parte din acesta. În Fig. II.14 este prezentat un exemplu de creare a unui server cu ajutorul `http`.

```
// importarea modului deja instalat pentru crearea unui server  
var http = require('http');  
  
const hostname = '127.0.0.1';  
const port = 8000;  
  
// crearea unui server care răspunde la toate cererile cu un status cod 200 și  
// textul // Hello World  
var server = http.createServer(function (request, response) {  
  response.statusCode = 200;  
  response.setHeader("Content-Type": "text/plain");  
  response.end("Hello World!\n");  
});  
  
// serverul ascultă la portul 8000 și adresa IP 127.0.0.1  
// după ce a pornit serverul se afișează un mesaj în consolă  
  
server.listen(port, hostname, () => {  
  console.log(`Server running at http://${hostname}:${port}/`);  
});
```

Fig. II.14 Modalitate de creare a unui server `http`

În Fig. II.15 este prezentată comanda completă ce trebuie executată pentru a porni un server creat în *Node.js*. Această comandă se execută în linia de comandă,

```
node nume_fisier.js
```

Fig. II.15 Comanda de pornire a serverului

¹⁷ <https://nodejs.org/en/download/>


O parte din beneficiile¹⁸ folosirii *Node.js* sunt:

- Codul este scris într-un singur limbaj, dar poate rula pe mai multe platforme;
- Este compatibil cu multe module, librării și extensii gratuite, disponibile în cadrul comunităților puternice formate în jurul *Node.js* și înregistrează o rată de adopție în piața în continuă creștere;
- E foarte rapid: în comparație cu alte limbaje, aplicațiile scrise în Node necesită mai puține linii de cod, mai puține fișiere, pot fi construite mai rapid și cu mai puțini programatori. Aplicațiile în *Node.js* nu sunt doar construite mai rapid, ci și rulează mai repede, având timpi de răspuns mult reduși și procesând mai multe cereri pe secundă, în comparație cu majoritatea alternativelor.

D. HTML

HTML (en. *HyperText Markup Language*) este un limbaj cu ajutorul căruia se pot construi pagini html. Acestea sunt construite folosind etichete sau taguri și au extensia „.html”.

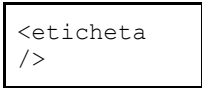
O etichetă trebuie să existe neapărat pereche, după cum se poate vedea în Fig. II.16.



```
<eticheta> conținutul din interiorul etichetei </eticheta>
```

Fig. II.16 Scrierea unui etichete cu conținut

Dacă o etichetă nu are conținut atunci ea poate fi scrisă și cum se vede în Fig. II.17.



```
<eticheta  
/>
```

Fig. II.17 Scrierea unei etichete fără conținut

O parte din etichetele și tagurile des folosite în crearea unei pagini html sunt următoarele:

- *html*: folosit pentru crearea efectivă a paginii web și anunță că blocul este de tip HTML;
- *head*: folosit pentru a realiza antetul paginii web;
- *meta*: folosit pentru a defini metainformații cum ar fi stiluri pentru formatarea textului;
- *link*: folosit pentru a face legătura cu un fișier;
- *title*: folosit pentru a da un titlu paginii web;
- *body*: folosit pentru a defini efectiv corpul paginii web;

¹⁸ <http://www.way2web.ro/tehnologii/node-js/>

O parte din tagurile definite mai sus fac parte din structura de bază a unei pagini web, de aceea ele trebuie folosite pereche, cu conținut în interior. În Fig. II.18 este prezentată o astfel de structură.

```
<html>
  <head>
    <title>Acesta este titlul paginii</title>
  </head>
  <body>
    Acest text este cel care va apărea în pagina propriu-
    zisă.
  </body>
</html>
```

Fig. II.18 Structura de bază a unei pagini html

Alte taguri și etichete cu ajutorul cărora se construiesc pagini html sunt:

- *a*: marchează legătura către o altă pagină web;
- *b*: formatează textul și îl face îngroșat;
- *br*: marchează sfârșitul de rând;
- *div*: structurează într-un bloc conținutul;
- *h1*, *h2*, *h3*, *h4*, *h5*, *h6*: mărește fontul și îl face îngroșat în funcție de cifră;
- *input*: inserează un câmp în care se poate insera și șterge text;
- *img*: inserează o imagine;
- *p*: inserează un paragraf nou etc.

Unele taguri au atribute predefinite și pot fi utilizate în funcție de ceea ce se dorește în pagină, dar toate tagurile pot avea aceleași atribute cum ar fi: *class*, *id*, *title*, *style* etc. De reținut faptul că mai multe etichete pot folosi aceeași clasă, dar un id poate fi folosit de o singură etichetă. În Fig. II.19 este scris un exemplu de cod care nu este corect pentru că nu respectă regula de mai sus, iar în Fig. II.20 este un exemplu de cod corect.

```
<div class="prima-clasa">
  <p id="primul-id">Acesta este primul paragraf</p>
  <p id="primul-id">Acesta este al doilea
  paragraf</p>
</div>
```

Fig. II.19 Atribuirea greșită a id-urilor pentru elemente

```
<div class="prima-clasa">
  <p id="primul-id">Acesta este primul paragraf</p>
  <p id="al-doilea-id">Acesta este al doilea paragraf</p>
</div>
<div class="prima-clasa">
  <p id="al-treilea-id">Acesta este al treilea paragraf</p>
</div>
```

Fig. II.20 Atribuirea corectă a id-urilor și claselor pentru elemente

Așadar, HTML este un limbaj ușor de învățat și utilizat pentru că folosește aceste etichete ce nu necesită depunerea unui efort foarte mare pentru a înțelege utilitatea lor. Totuși, dacă se respectă structura exemplificată mai sus la care se adaugă alte informații și taguri conform cu utilitatea lor exemplificată mai sus, se obține o pagină html ce nu este foarte atractivă pentru un utilizator, fiind asemănătoare cu un text scris în editorul Notepad. Un exemplu de astfel de pagină este în Fig. II.20.

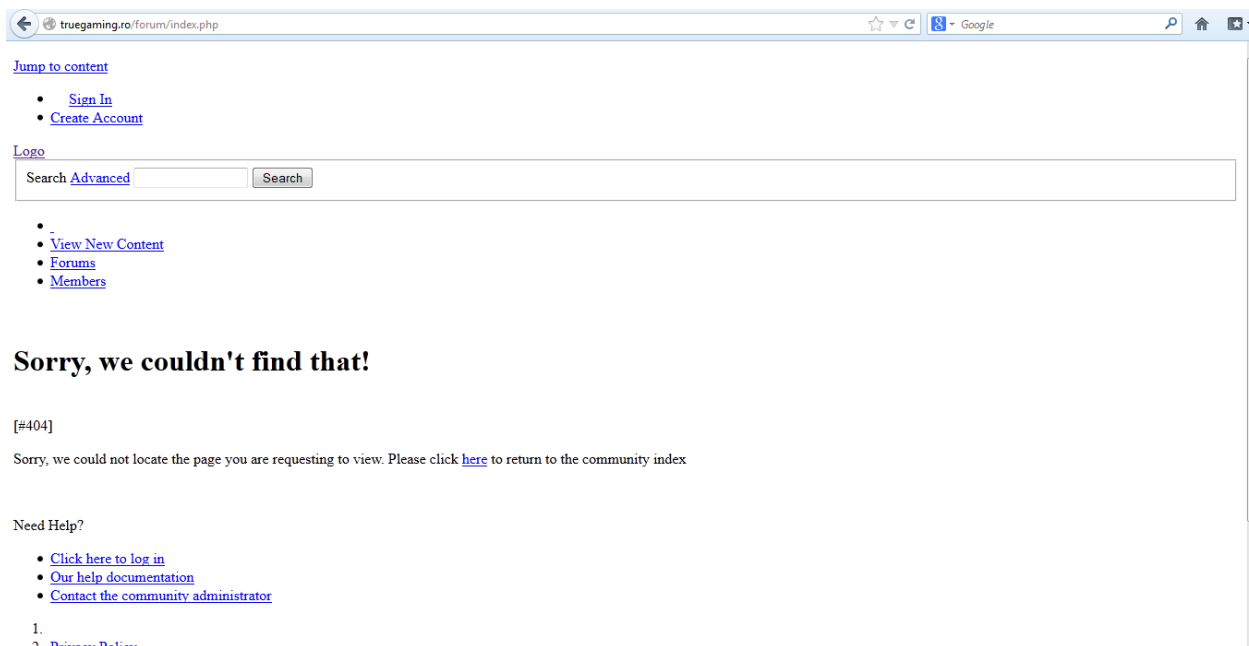


Fig. II.20 Pagină web fără stilizare¹⁹

Un programator ce utilizează html dorește ca experiența unui vizitator al paginii să fie una plăcută și frumoasă care să îl determine să mai revină pe pagină, de aceea aceștia îi trebuie adăugate culori, fonturi, animații, precum și alte elemente cheie în interacțiunea utilizatorului cu pagina. Pentru a putea realiza acestea, programatorul are la îndemână un limbaj numit CSS, descris în subcapitolul următor.

¹⁹https://www.google.ro/search?q=pagina+nestilizata&rlz=1C1CHBF_enRO731RO731&source=lnms&tbm=isch&sa=X&ved=0ahUKEwj_xKj2kebUAhUBuBQKHVA5DpsQ_AUICigB&biw=950&bih=930#imgsrc=pgadEAOLCsAzXM:

E. CSS

CSS (*ro. foi de stil în cascadă, en. Cascading Style Sheets*) este un limbaj de stilizare al elementelor²⁰ conținute de o pagină html descrisă în subcapitolul anterior. Limbajul CSS este folosit atunci când un programator dorește să stilizeze o astfel de pagină și vine cu următoarele avantaje²¹:

- Separă conținutul de prezentare;
- Permite definirea modului de afișare și așezarea în pagină într-un singur fișier a tuturor elementelor din mai multe pagini html;

Folosind CSS, se poate seta tot ce ține de estetica unei pagini web: ce culoare sau imagine să aibă fundalul paginii, ce culoare, dimensiune și stil să aibă textul din pagină, cum să apară sau nu un element când se dă click pe un buton și multe altele. Astfel, tot ce ține de design se poate seta în CSS.

În CSS un element se poate stiliza în funcție de numele clasei și astfel se modifică toate elementele care au clasa respectivă, în funcție de id-ul elementului sau în funcție de tag și se vor modifica toate elementele cu acel tag.

- Pentru a selecta o clasă se pune un punct și apoi numele clasei. Acestea se scriu legat. În Fig. II.21 este reprezentat cum se selectează în CSS o clasă;

```
.nume-clasa {  
    color: red;  
    font-size: 14px;  
}
```

Fig. II.21 Selectarea unei clase în CSS

- Pentru a selecta un id se pune # și apoi numele id-ului. Acestea se scriu legat. În Fig. II.22 este reprezentat cum se selectează în CSS un id;

```
#nume-id {  
    color: blue;  
    font-size: 16px;  
}
```

Fig. II.22 Selectarea unui id în CSS

- Pentru selectarea unui tag se scrie doar numele tagului. Fig. II.23 este reprezentat cum se selectează în CSS un tag;

²⁰ <http://it.webdesign-galaxy.ro/ce-este-css/>

²¹ Slide 1 din <http://inf.ucv.ro/~mihaiug/courses/web/slides/Curs%204%20-%20CSS.pdf>

```
nume-tag {  
    color: green;  
    font-size: 10px;  
}
```

Fig. II.22 Selectarea unui id în CSS

Când se dorește stilizarea în CSS trebuie să se țină cont de faptul că dacă în fișierul CSS se stilizează de mai multe ori un element, atunci nu se va lua decât ultimul. Trebuie atenție sporită la acest aspect deoarece de multe ori un element poate apărea diferit de cum ar fi crezut programatorul și e posibil să dureze un timp până găsește cauza. De exemplu, în Fig. II.23 sunt prezentate mai multe elemente, care sunt stilizate:

```
.mar {  
    color: green;  
    font-size:  
14px,  
}  
  
#portocala {  
    color: orange;  
}  
  
.mar {  
    font-size:  
10px;  
}
```

Fig. II.23 Stilizarea mai multor elemente în CSS

Dacă se dorește ca fontul textului din interiorul clasei *mar* să fie *14px*, acesta va avea de fapt dimensiunea de *10px*, deoarece a fost suprascris fontul pentru acea clasă, iar *px* reprezintă o unitate de măsură în CSS. Din această cauză este de preferat să se evite suprascrierea pentru a nu se crea confuzii.

Când se dorește stilizarea mai multor elemente ca în Fig. II.23 de mai sus, atunci se vor selecta clasele, id-urile și tag-urile care se doresc a fi modificate și se vor scrie una sub alta. Se pot scrie în orice ordine, dar trebuie ținut cont de cazul de suprascriere detaliat mai sus.

Așadar, este de preferat ca paginile web să fie stilizate cât mai frumos pentru a-i oferi utilizatorului o experiență cât mai plăcută. Fig. II.24 reprezintă un imagine a unei pagini web în care s-a folosit CSS.

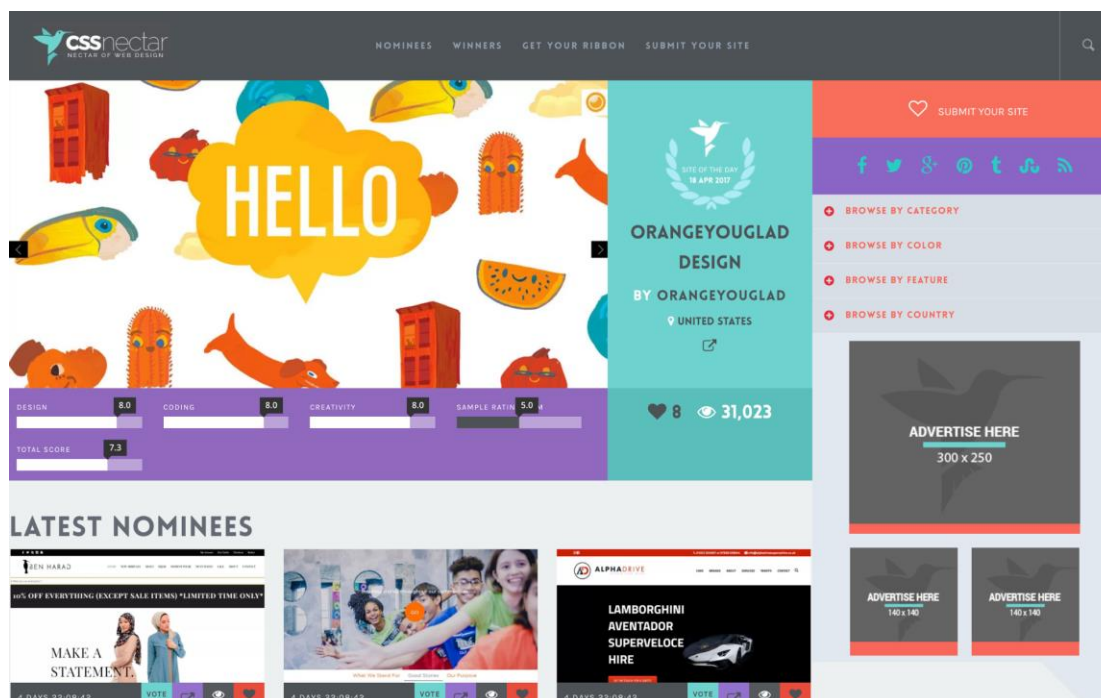


Fig. II. 24 Pagină web realizată cu CSS ²²

Făcând comparație între Fig. II.20 din subcapitolul anterior și Fig. II.24 din acest subcapitol, se poate observa o diferență majoră de design, așadar utilizarea CSS este benefică pentru utilizatori.

F. SASS

SASS (en. Syntactically Awesome Style Sheets) reprezintă un preprocesator (sau extensie) de fișiere CSS despre care am discutat în subcapitolul anterior. Așadar, *SASS* folosește CSS ca limbaj, dar permite organizarea mai bună a blocurilor pentru stilizarea paginii web. Când se compilează un fișier *SASS* se crează automat și fișierul CSS de unde pagina web își va lua datele pentru stilizare. Astfel, folosind *SASS* nu se va vedea o diferență la nivelul paginii web, dar are, pentru programatori, avantajul creării unui cod lizibil, bine structurat și fără repetiții.

SASS a fost dezvoltat în Ruby (un limbaj de programare). Așadar, pentru a putea fi folosit pe Windows, este necesară instalarea Ruby și apoi instalarea efectivă a preprocesatorului. La

²²https://www.google.ro/search?q=pagina+stilizata+vs+nestilizata&rlz=1C1CHBF_enRO731RO731&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiS4JHskebUAhVEvBQKHSouDrUQ_AUICigB&biw=1920&bih=901#tbm=isch&q=pagina+web+css+design&imgrc=V6jCvIZUooJgwM:

adresa <http://sass-lang.com/install> se pot găsi mai multe informații despre instalare. Dacă se folosește OS X atunci Ruby este deja instalat și se instalează doar SASS.

Un fișier SASS poate folosi 2 tipuri de sintaxe²³: SASS și SCSS. Diferența dintre cele 2 este doar la nivel de scriere întrucât ele fac același lucru ca și funcționalitate. Astfel, sintaxa SASS este identificată prin extensia *.sass* și se folosesc tab-uri și rânduri noi pentru delimitarea definițiilor CSS și definirea blocurilor, iar sintaxa SCSS este identificată prin extensia *.scss* și folosește acoladele pentru definirea de blocuri de definiții și delimitarea definițiilor CSS. În Fig. II.21 sunt exemple de cod scris în cele 2 sintaxe, precum și rezultatul CSS obținut în urma compilării.

/* SCSS */	/* SASS */	/* CSS */
<pre>\$rosu: red; \$negru: black; \$verde: green; \$marime: 16px; .continut { .titlu { color: \$rosu; font-size: \$marime; } .descriere { border-color: \$negru; color: \$verde; } }</pre>	<pre>\$rosu: red \$negru: black \$verde: green \$marime: 16px .continut .titlu color: \$rosu font-size: \$marime .descriere border-color: \$negru color: \$verde</pre>	<pre>.titlu { color: red; font-size: 16px; } .descriere { border-color: black; color: green; }</pre>

Fig. II.21 Exemple de cod pentru SCSS, SASS și rezultatul în CSS

Se poate observa că în Fig. II.21 de mai sus s-a utilizat cod precum `$rosu: red;` ce semnifică atribuirea culorii `red` variabilei `rosu`. Acest procedeu reprezintă declararea de variabile. Ele se comportă ca niște constante și pot fi folosite în cod de câte ori este nevoie. Totodată se observă și existența de blocuri în blocuri, iar acest procedeu reprezintă încuibarea sau imbricarea selectorilor (*en. nesting*). Încuibarea permite crearea de stilizări ierarhice ce respectă structura codului html prin care este alcătuită pagina web.

Utilizând SASS este permisă modularizarea codului și, astfel, crearea de fișiere mai mici ce pot fi ușor accesate și modificate. Împărțirea codului CSS în multiple zone este benefică pentru că se permite astfel o mai bună structurare și ierarhizare a codului, deci acesta va deveni mai ușor de accesat, înțeles și modificat. În plus, se poate separa partea dinamică, în care sunt

²³ <http://www.adibarbu.ro/2017/05/introductie-instalare-sass.html>

definite variabile, de partea statică, în care există codul propriu-zis, facilitând creșterea gradului de actualizare și mentenanță al codului.²⁴ În Fig. II.22 este comanda pentru importarea unui astfel de fișier.

```
@import 'nume_fisier_mai_mic';
```

Fig. II.22 Comanda pentru import-ul de fișiere în SASS

Spre deosebire de CSS fără preprocesator, dacă se dorește utilizarea aceleiași cod de mai multe ori, SASS are opțiunea de a crea o clasă în acel fișier care conține stilizarea dorită și apoi poate fi preluată în alte clase folosind comanda `@extend`. În Fig. II.23 este exemplificată o astfel de situație.

```
.capsula {  
  border: 1px solid black;  
  background-color: grey;  
  color: green;  
}  
  
.continut1 {  
  @extend .capsula;  
  font-size: 16px;  
  
  .continut2 {  
    @extend .capsula;  
    height: 100px;  
  }  
}  
  
.continut3 {  
  @extend .capsula;  
}
```

Fig. II.23 Folosirea @extend pentru evitarea de cod duplicat

În SASS se pot efectua operații matematice direct în cod. În Fig. II.24 este un astfel de exemplu.

```
$dark: #222;  
$light: $dark + #333;  
  
.content {  
  height: 200px / 360px * 100%;  
}
```

Fig. II.24 Efectuare operații matematice direct în codul CSS în SASS

²⁴ <http://www.adibaru.ro/2017/05/introductie-instalare-sass.html>

În Fig. II.25 se folosește operatul de referire la părinte. Codul descris în aceasta se interpretează astfel: atunci când cursorul de la mouse este deasupra unui tag *div* se va schimba fontul la 14px.

```
body {  
  color: green;  
  
  .div {  
    font-size: 10px;  
    &:hover {  
      font-size: 14px;  
    }  
  }  
}
```

Fig. II.25 Utilizarea în SASS a operatorului de referire la părinte

La adresa http://sass-lang.com/documentation/file.SASS_REFERENCE.html se poate găsi documentația completă a SASS.

Așadar, acest limbaj de preprocesare SASS este util pentru programatori pentru că pentru structura claselor se poate folosi aceeași structură ierarhizată ca în pagina HTML.

G. ReactJS

Conform documentației oficiale²⁵, *ReactJS* este o librărie gratuită JavaScript, folosită pentru a construi interfețe web în special pentru aplicațiile alcătuite dintr-o singură pagină și întreținută în mare parte de *Facebook* și *Instagram*. Această librărie permite crearea de aplicații mari fără a reîncărca pagina. *ReactJS* este orientat spre refolosirea componentelor deja create pentru a nu scrie cod duplicat, oferindu-se, astfel, modularizare. De exemplu, dacă într-o pagină web sunt 3 formulare de înregistrare, atunci în toate cele 3 locuri trebuie scris același cod. Folosind *ReactJS*, se poate scrie crea o componentă și importată acolo unde este nevoie, evitându-se, astfel, încărcarea codului cu bucăți deja scrise.

Pentru a putea folosi această librărie trebuie instalat *Babel* și *ReactJS* din linia de comanda. La adresa https://www.tutorialspoint.com/reactjs/reactjs_environment_setup.htm se găsesc mai multe informații despre configurare și instalare.

²⁵ <https://facebook.github.io/react/docs/hello-world.html>

ReactJS folosește sintaxa JSX. Aceasta folosește codul simplu JavaScript și permite imbricarea elementelor HTML cu scopul de a crea subcomponente²⁶. În Fig. II.26 este un exemplu de astfel de cod.

```
return (
  <div>
    {
      persoane.map((pers) => {
        return <p>Numele este {pers.num};</p>
      })
    }
  </div>
);
```

Fig. II.26 Exemplu pentru sintaxa JSX

În *ReactJS* datele sunt trimise de la părinte la copil și nu invers. Părintele este cel care îi spune copilului să facă, iar acesta doar execută și trimite înapoi răspunsul și nu știe pentru ce îi trebuie acel răspuns. Pentru a realiza această comunicare între componente sunt mai multe modalități²⁷, iar cele mai importante și mai des folosite dintre ele sunt prezentate în continuare.

1. Comunicarea de la părinte la copil se realizează prin *props*. Părintele trimite date prin intermediul acestora. În Fig.II.27 sunt prezentate părți de cod din componentele în care se face astfel de comunicare, în care *date_personale* reprezintă *props*. Funcționalitatea îndeplinită de ele este următoarea: pentru fiecare persoană din vectorul *persoane* din componenta părinte se va afișa numele persoanei prin intermediul componentei copil.

```
/* parinte */
return (
  <div>
    {
      persoane.map((pers) => {
        return <Persoana date_personale={pers} />
      })
    }
  </div>
);

/* copil */
return (
  <div>
    <p>Numele persoanei este {this.props.date_personale.num}</p>
  </div>
);
```

Fig. II.27 Comunicarea părinte-copil prin *props*

²⁶ <http://www.c-sharpcorner.com/article/what-and-why-reactjs/>

²⁷ <http://andrewfarmer.com/component-communication/>

2. Comunicarea de la copil la părinte se realizează prin *callback functions*. Acestea sunt funcții ce se execută în părinte, dar lansate în execuție din copil și pot avea parametri sau nu. În Fig. II.28 este prezentat un exemplu pentru astfel de comunicare, iar funcționalitatea este: după ce din părinte s-a accesat copilul și s-a afișat mesajul „Numele persoanei este ...” se lansează în execuție și funcția *afiseazaMesaj* din părinte și se va afișa în consolă mesajul „S-a afișat numele ...”.

```
/* parinte */
afiseazaMesaj(nume_persoana) {
  console.log('S-a afișat numele', nume_persoana);
}
return (
  <div>
    {
      persoane.map((pers) => {
        return <Persoana
          date_personale={pers}
          functieCallback={this.afiseazaM}
        />
      })
    }
  </div>
);

/* copil */
let nume_persoana = this.props.date_personale.nume;
return (
  <div>
    <p>Numele persoanei este {nume_persoana}</p>
    {this.props.functieCallback(nume_persoana)}
  </div>
);
```

Fig. II.28 Comunicarea copil-părinte prin *funcții callback*

Se poate observa că, în Fig. II.28 de mai sus, părintele i-a transmis copilului prin *props* funcția ce trebuie să o execute. În același timp, în Fig. II.27 și Fig. II.28 se observă elemente imbricate de JavaScript și HTML care sunt specifice sintaxei JSX folosită de *ReactJs*.

Pe lângă aceste 2 comunicări mai există comunicare între copiii aceluiași părinte și comunicare între copiii care au părinți diferiți, dar este de preferat evitarea folosirii lor mai ales dacă acestea se pot realiza prin combinarea celor 2 metode de comunicare între părinte-copil și copil-părinte explicate mai sus. De exemplu, comunicarea dintre copiii aceluiași părinte, folosind doar pe cele 2 uzuale, se poate face astfel: se transmit datele de la copil la părinte printr-o funcție de callback, iar părintele le transmite la celălalt copil. Pentru a putea jongla cu comunicările trebuie știută foarte bine ierarhia părinților și copiilor.

O imagine de ansamblu a acestor comunicări este prezentată în Fig. II.29.

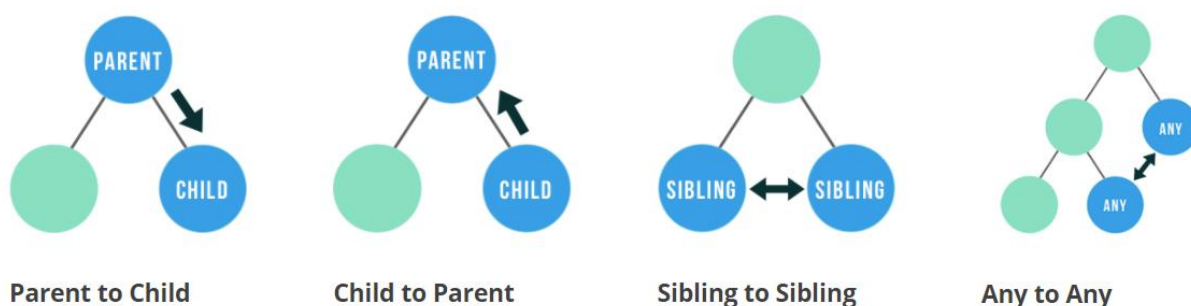


Fig. II.29 Imagine de ansamblu a comunicării dintre componente²⁸

O componentă în *ReactJs* este de fapt o clasă pentru că ea poate avea constructor și funcții. O caracteristică esențială a *ReactJs* este posibilitatea deținerii de către fiecare componentă a unui *state*. Acest *state* este deținut de constructor și poate fi privit ca un obiect ce conține informații temporare. Este de preferat, dacă este posibil, încercarea de a se utiliza cât mai puțin *state*-ul. De exemplu, putem utiliza *state*-ul în momentul în care dorim ca în momentul în care se apasă pe un buton să apară sau să dispară un element din pagină, iar în Fig. II. 30 este o parte din codul necesar.

```
class MakeButton extends Component() {
  constructor() {
    super();
    this.state = {
      elementAfisat: false
    }
    this.schimbaVizibilitateaElementului =
      this.schimbaVizibilitateaElementului.bind(this);
    this.afiseazaElement = this.afiseazaElement.bind(this);
  }
  schimbaVizibilitateaElementului() {
    this.setState({elementAfisat: !this.state.elementAfisat});
  }
  afiseazaElement() {
    if(this.state.elementAfisat === true) {
      return (<div>Acesta este elementul ce trebuie ascuns!</div>);
    }
  }
  render() {
    return (
      <div>
        {this.afiseazaElement()}
        <button
          onClick={this.schimbaVizibilitateaElementului}>Schimba</button>
      </div>
    );
  }
}
```

Fig. II.30 Exemplu de folosire a *state*-ului

²⁸ <http://andrewfarmer.com/component-communication/>

Componentele din *ReactJS* pot fi de 2 feluri: *smart* sau *dumb*. Diferența între cele 2 este că cea *smart* are constructor și *state*, iar cea *dumb* nu are și este privită doar ca o funcție. În Fig.II.31 este codul pentru crearea unei componente *dumb*, iar în Fig. II.32 este codul pentru crearea unei componente *smart* ce utilizează componenta *dumb* din Fig. II.33.

```
1 export default function Tab({tab, callback, isSelected}) {
2   let {id, name} = tab,
3     tabClass = isSelected ? 'tab selected' : 'tab';
4   return (
5     <div key={id} className={tabClass} onClick={callback.bind(this,id)}>
6       {name || 'Tab'}
7     </div>
8   );
9 }
10
```

Fig. II.32 Codul pentru realizarea unei componente *dumb*

```
1 import Tab from './tab';
2
3 export default function Tabs({tabs, callback, selectedTab}) {
4   function isSelected(tabId) {
5     return selectedTab === tabId;
6   }
7
8   return (
9     <div className="tabs-wrapper">
10      {
11        tabs.map( item => {
12          return (
13            <Tab tab={item} callback={callback} isSelected={isSelected(item.id)}/>
14          );
15        })
16      }
17    </div>
18  );
19 }
20
```

Fig. II.32 Codul pentru realizarea unei componente *smart*

Așadar, *ReactJs* este simplu de învățat, oferă modularitate codului și este preferat de folosit pentru aplicațiile web realizate dintr-o singură pagină pentru că o componentă este scrisă o dată și bine și folosită oriunde.

Cap. III Soluția propusă

Aplicația *Recenzii Restaurante Iași* vine în ajutorul studenților din Iași, dar mai ales studenților de la Facultatea de Informatică deoarece toate restaurantele existente în baza de date a acestei aplicații sunt din jurul ei.

În momentul în care un student (pe care îl vom denumi în continuare utilizator) dorește să afle informații despre un anumit restaurant existent în aplicație, el nu trebuie decât să deschidă *Recenzii restaurante Iași*. Are posibilitatea de a vizualiza informații despre toate restaurantele din baza de date sau informații despre un anumit restaurant. Din aceste informații fac parte recenziile date de alți utilizatori, adresa localului, programul de funcționare precum și, dacă deține, adresa site-ului web. Recenziile sunt ordonate și evidențiate în funcție de gradul de credibilitate al fiecăreia, astfel un vizitator se poate folosi de ele în aventura căutării unui local pe gustul lui.

În același timp o persoană care folosește aplicația, poate vizualiza pe hartă restaurantele, de unde poate accesa informații despre un anumit local. În cazul în care utilizatorul dorește să vizualizeze doar anumite restaurante care au un anumit tag (cum ar fi: *food delivery*, *coffe to go*, *food* etc.) el poate pune un filtru care îi va arăta doar restaurantele cu acel tag.

Totodată, dacă un utilizator știe deja numele unui local, dar vrea să vadă recenziile acestuia, el le poate căuta în funcție de numele localului.

Dacă o persoană a vizitat un anumit restaurant și dorește să împărtășească experiența lui de acolo și, astfel, să ajute alte persoane în luarea deciziei alegerii unui restaurant, ea are posibilitatea de a-și crea un cont cu ajutorul căruia poate scrie aceste recenzii. O dată autentificat, un utilizator poate să vadă restaurantele la care a scris cel puțin o recenzie, toate recenziile scrise de acesta, toate restaurantele existente în aplicație, precum și harta unde acestea apar.

Aplicația oferă și posibilitatea de a-i arăta unui utilizator autentificat drumul de la locația curentă a acestuia și până la restaurantul dorit de el.

A. Arhitectura aplicației

Pentru realizarea aplicației *Recenzii Restaurante Iași* am folosit o bază de date MySQL, un server creat în Node.js cu ajutorul framework-ului Express, precum și un client ce reprezintă pagina web efectivă. În Fig.III.1 este o imagine de ansamblu a aplicației.

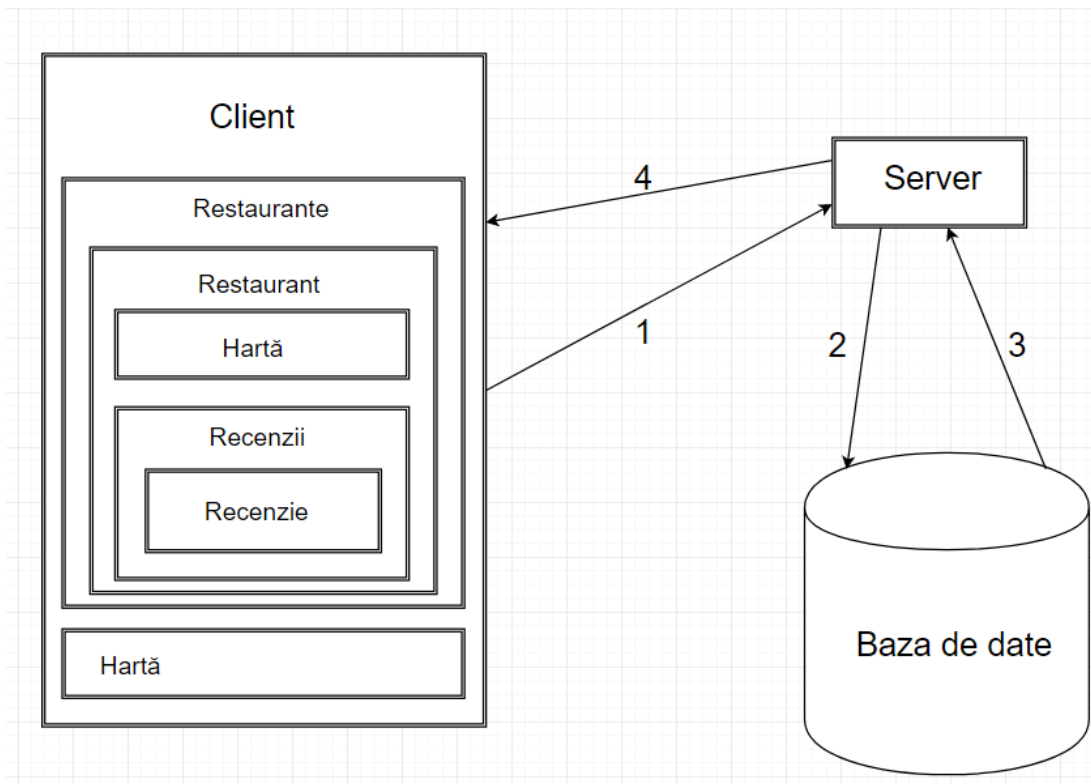


Fig. III.1 Imagine de ansamblu a aplicației

De exemplu, în momentul în care este pornit clientul se face o cerere către server pentru a-i arăta toate restaurantele. La rândul lui, serverul face o cerere către baza de date care îi trimite acestuia toate restaurantele, iar apoi acesta le trimite înapoi la client. În Fig. III.2 este prezentată o imagine schematică a acestor cereri.

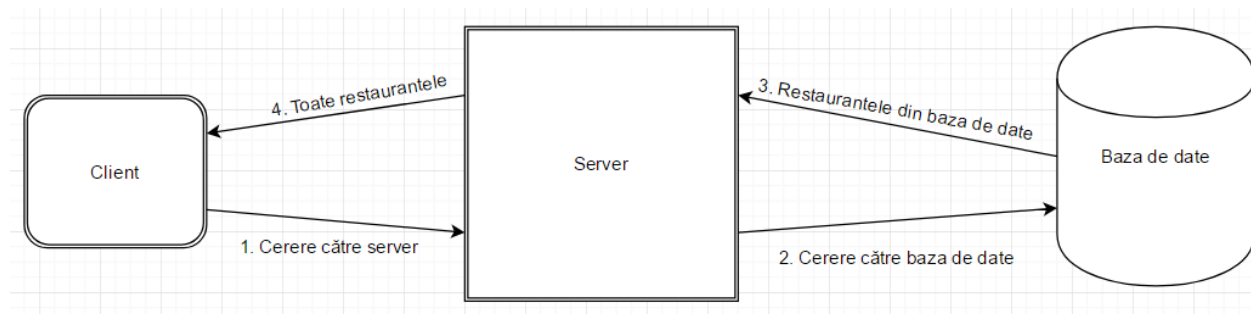


Fig. III.2 Cum se realizează cererile pentru exemplul dat

1. Baza de date

Baza de date a aplicației este construită în MySQL. O imagine de ansamblu a acestei baze de date este prezentată în Fig. III.3.

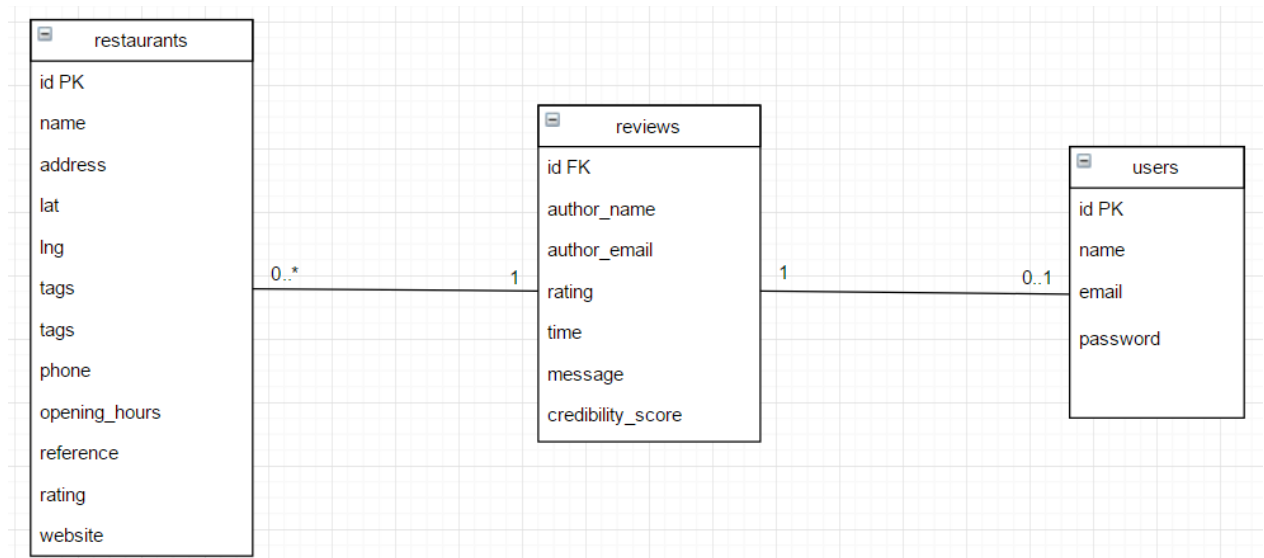


Fig. III.3 Imagine de ansamblu a bazei de date

Pentru popularea ei cu date reale am folosit *Google Places*. După ce mi-am creat cont de *Gmail*, am obținut o cheie pentru a o putea folosi în link-ul cererii care se face pentru a găsi restaurantele relevante pentru aplicație. Pentru fiecare restaurant găsit am făcut o a doua cerere, folosind aceeași cheie, către un alt link tot de la *Google Places* din care am obținut informații necesare cum ar fi adresa fiecărui local, orarul de funcționare și altele. În Fig.III.4 este o parte din codul folosit pentru cererile efectuate către *Google Places*.

```
request('https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=47.1742073,27.5707641&radius=500&type=restaurant&key=AIzaSyBmPVCIX0mMesnQ0-Bw_I
if (error && response.statusCode === 200) {
  restaurants = restaurants.concat(functionsForDatabase.getRestaurantsReference(response.body));
  fs.writeFileSync('./scripts/insert_datas_into_restaurant.sql', '');
  fs.writeFileSync('./scripts/insert_datas_into_reviews.sql', '');
  fs.writeFileSync('./scripts/insert_datas_into_photos.sql', '');
  if(restaurants.length > 0) {
    for(var i = 0; i < restaurants.length; i++) {
      let link = 'https://maps.googleapis.com/maps/api/place/details/json?reference=' + restaurants[i].reference + '&sensor=true&key=AIzaSyBmPVCIX0m
      request(link, (error, response) => {
        if (error && response.statusCode === 200) {
          functionsForDatabase.parseResponse(response.body);
        }
        else {
          console.log("Error at API request! ");
          process.exit();
        }
      });
    }
  }
  else {
    console.log("Error at API request! ");
    process.exit();
  }
});
```

Fig. III.4 O parte din codul pentru cererile efectuate pentru popularea bazei de date

Aplicația *Recenzii Restaurante Iași* este orientată către credibilitatea fiecărei recenzii. De aceea pentru fiecare în parte am calculat un scor cu ajutorul unui algoritm. Deoarece am folosit Node.js, am instalat o librărie de la *Google Translate* ca în Fig. III.5 care îmi permite traducerea mesajului recenziei în limba engleză din orice limbă este.

```
npm install @google-cloud/translate --save
```

Fig. III.5

După ce am obținut textul tradus, am apelat la un alt API de la Watson²⁹: *NaturalLanguageUnderstanding*. Cu ajutorul unei cereri către acest API care parsează textul și verifică sentimentele conținute de acesta, am obținut un scor. Acest scor împreună cu numărul de puncte pe care le-a dat persoana care a scris recenzia le-am folosit într-un algoritm pentru determinarea credibilității recenziei. Codul algoritmului este prezent în Fig. III.6 și în continuare îl voi explica pe scurt. În cazul în care numărul de puncte, care pot fi de la 1 la 5, pe care le vom denumi în continuare *rata*, este mai mic decât 3, le transform în număr negativ și le procesăm mai departe după cazuri:

- a. Dacă rata și scorul mesajului sunt numere pozitive atunci doar le adun și obțin scorul de credibilitate al mesajului;
- b. Dacă rata și scorul mesajului sunt numere negative atunci rata o transform în număr pozitiv și o modific în numărul 5 sau 4 dacă ea este 1 sau, respectiv, 2. Apoi transform și scorul mesajului în număr pozitiv și adun noua rată cu noul scor și obțin scorul de credibilitate al mesajului;
- c. Dacă rata este număr pozitiv și scorul este număr negativ doar transform rata în număr negativ și o adun cu scorul și obțin scorul de credibilitate al mesajului;
- d. Dacă rata este număr negativ și scorul este număr pozitiv atunci aplic același algoritm de transformare a ratei ca în cazul b și apoi o fac număr negativ și transform și scorul în număr negativ. Rata nouă și scorul nou le adun și obțin scorul de credibilitate al mesajului.

²⁹ <https://www.ibm.com/watson/developercloud/>

```

let newRate = null;
if(rate < 3) {
    newRate = (-1) * rate;
} else newRate = rate;

if(score < 0 && newRate < 0) {
    switch (newRate) {
        case -1:
            newRate = -5;
            break;
        case -2:
            newRate = -4;
        default:
            newRate = newRate;
    }
    credibility_score = (-1) * (score + newRate);
} else if(score >= 0 && newRate > 0) {
    credibility_score = score + newRate;
} else if(score < 0 && newRate > 0) {
    credibility_score = score + (-1) * newRate;
} else if(score >= 0 && newRate < 0) {
    switch (newRate) {
        case -1:
            newRate = -5;
            break;
        case -2:
            newRate = -4;
        default:
            newRate = newRate;
    }

    credibility_score = (-1) * score + newRate;
}

```

Fig. III.6 Codul algoritmul de determinare al credibilității mesajului

După ce am calculat acest scor de credibilitate se inserează în baza de date împreună cu informațiile particulare. După ce am realizat acest algoritm pentru fiecare în parte, am populat baza de date.

2. Server-ul

Serverul aplicației a fost creat în Node.js cu ajutorul framework-ului Express. Pentru a-l putea importa și utiliza în aplicație, el a trebuie mai întâi instalat cu comanda din Fig. III.7.

```
npm install express --save
```

Fig. III.7 Comanda de instalare a framework-ului Express

Din client se lansează o cerere către server, iar aceasta o manageriază în funcție de fiecare cerere în parte. În aplicația *Recenzii Restaurante Iași* fiecare cerere accesează baza de date și aduce informațiile necesare în funcție de tipul și parametrii cererii. Tipurile folosite sunt GET și POST. GET se folosește când se dorește doar aducerea de informații de la baza de date cum ar fi toate restaurantele, iar POST se folosește atunci se dorește inserarea unor date cum ar fi atunci când se crează un cont nou. Un exemplu de manageriere în server a unei cereri se găsește în Fig.III.8.

```
15 router.get('/', function(req, res) {
16   var connection = mysql.createConnection({
17     host: "localhost",
18     user: "root",
19     password: "",
20     database: "restaurants",
21     debug: false,
22     multipleStatements: true
23   });
24   connection.connect((err) => {
25     if(err) {
26       console.log('Error connecting to database! \nExecution stopped! \n' + 'Error message: ' + err.message);
27       res.json({typeError: 'DBconnect', text: 'Error connecting to database! \nExecution stopped! \n' + 'Error message: ' + err.message});
28     }
29     id = connection.threadId;
30     console.log('Connection established!\nConnected with id: ' + id);
31   });
32
33
34   connection.query('SELECT * FROM restaurants ORDER BY rating DESC', (err, result) => {
35     var results = null;
36     if(err) {
37       console.log('Error searching in database! \nExecution stopped! \n' + 'Error message: ' + err.message);
38       res.json({typeError: 'DBselect', text: 'Error searching in database! \nExecution stopped! \n' + 'Error message: ' + err.message});
39       connectionEnd(connection, id);
40     } else {
41       results = result.length;
42       if(results > 0) {
43         var obj_response = [];
44         for(var i=0; i < result.length; i++) {
45           var obj = {};
46           obj.id = result[i].id;
```

Fig. III.8 Exemplu de manageriere a unei cereri

Fiecare cerere este manageriată într-un fișier separat, ceea ce oferă modularitate și ușurință în înțelegerea a ceea ce face server-ul. Așadar, acesta nu face decât să primească o cerere de la client, se uită ce fel de cerere este, se duce în fișierul specific, execută codul cu ajutorul

căruia se conectează la baza de date de unde ia informațiile pe care le transformă într-un obiect și le trimite înapoi la client.

3. Clientul

Clientul reprezintă partea cea mai importantă pentru un utilizator. Acesta este locul în care el vede efectiv ceva vizual. Aici se află toate informațiile venite de la server, cu care comunică direct. Fără client nu se pot efectua cereri și astfel nu mai există informații. Clientul aplicației *Recenzii Restaurante Iași* a fost realizat în ReactJS și este bazat pe mai multe componente pe care le-am creat într-un singur loc și pe care, pe unele dintre ele, le-am folosit în mai multe locuri. Astfel, în continuare voi prezenta principalele componente ale aplicației.

a. Componenta Restaurante

Componenta pentru restaurante conține informații stilizate pentru toate restaurantele existente în baza de date. Acestea au fost obținute printr-o cerere de tip GET la server, iar în momentul când un utilizator accesează aplicația i se va afișa aceste restaurante ca în Fig. III.9.

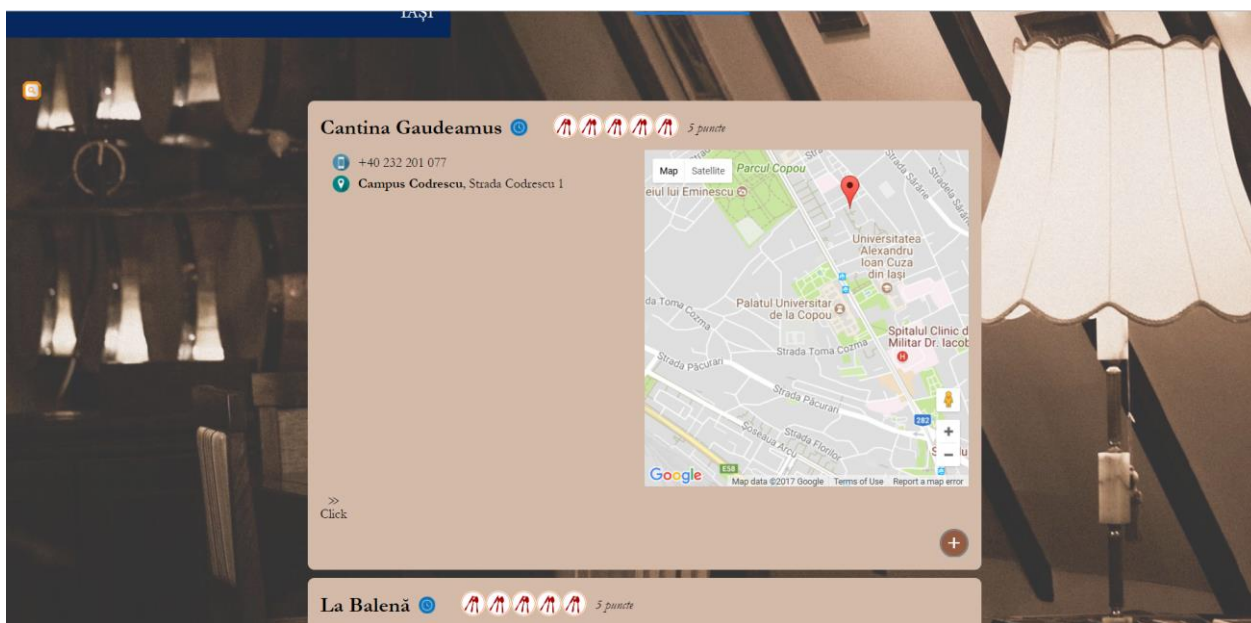


Fig. III.9 Afișarea tuturor restaurantelor

Utilizatorul poate filtra restaurantele în funcție de numele pe care îl dorește. În Fig. III.10 se poate observa rezultatul unei astfel de filtrări.

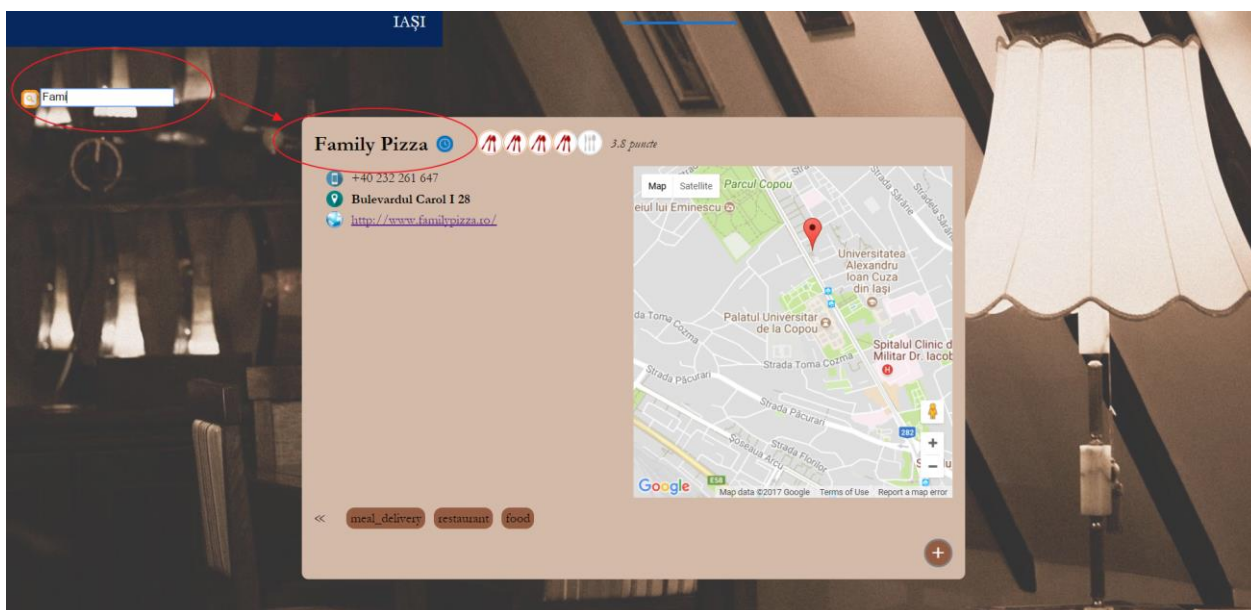


Fig. III.10 Rezultatul unei filtrări a restaurantelor

Totodată o filtrare se mai poate face și după tag-ul pe care restaurantele îl au. De exemplu, dacă un utilizator dorește să vadă doar acele localuri în care se poate mânca, atunci se poate selecta tag-ul *food* din lista de tag-uri a unui restaurant ca în Fig. III.11.

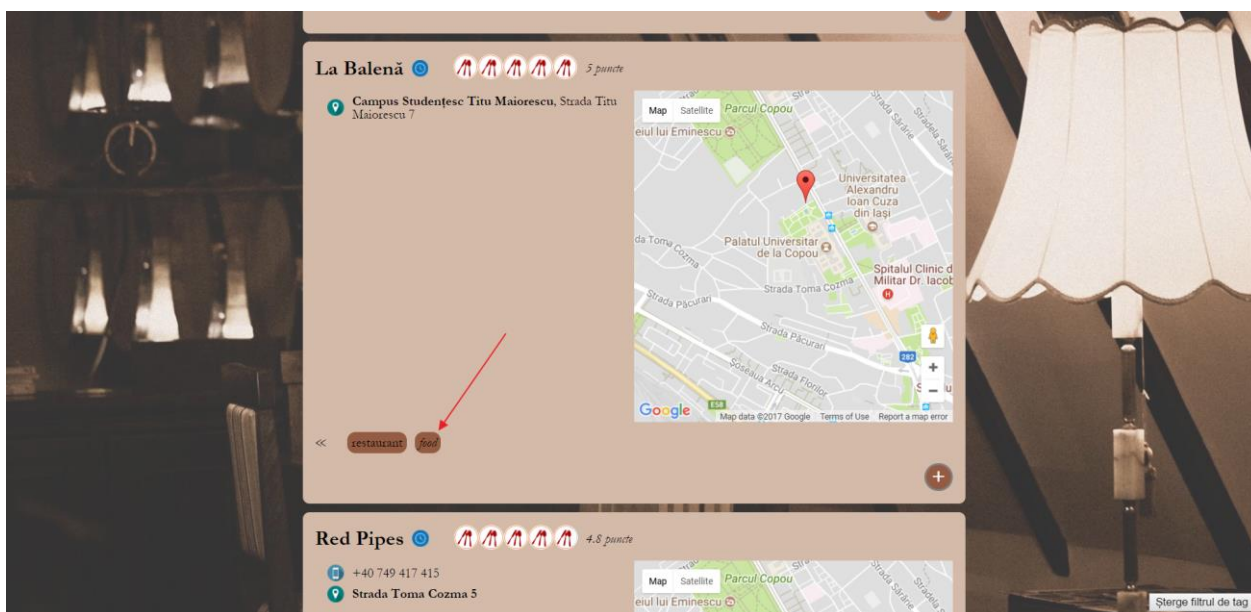


Fig. III.11 Filtrarea restaurantelor în funcție de tag-ul dorit

Din aceeași componentă un utilizator are opțiunea de a adăuga o recenzie. Pentru aceasta el trebuie să dea click pe butonul de adăugare din colțul din dreapta al unui restaurant. Dacă el nu este autentificat atunci i se va afișa un modal pentru autentificare sau înregistrare dacă nu are cont, ca în Fig. III.12.

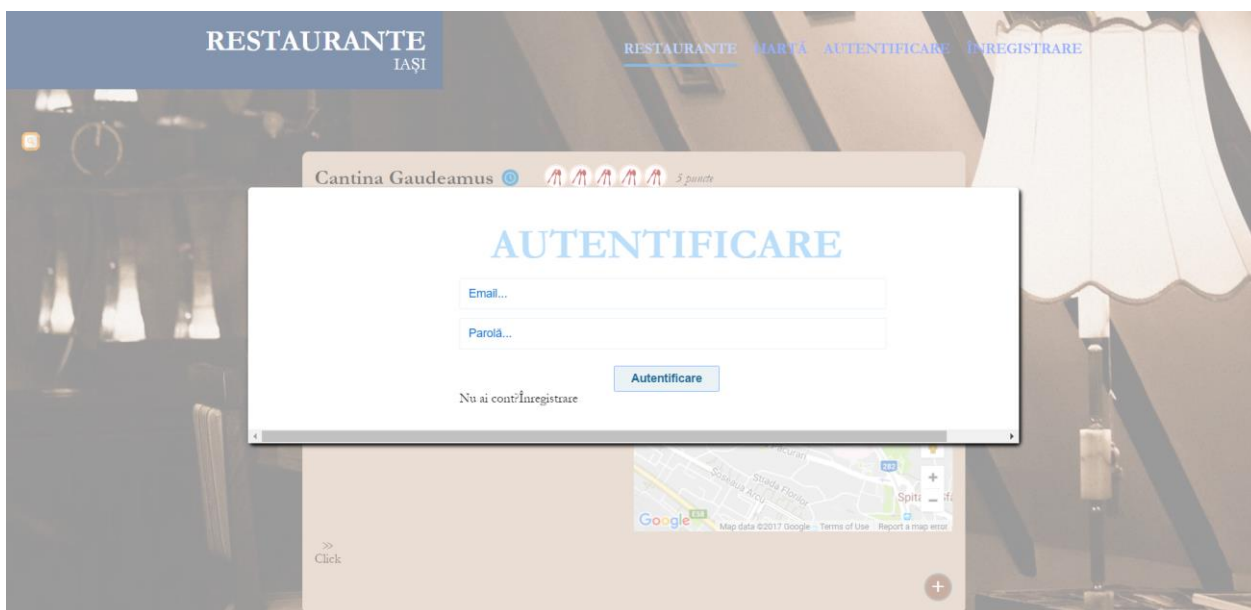


Fig. III.12 Afișarea modalului pentru autentificare sau înregistrare

Tot din aceeași componentă utilizatorul are opțiunea, dacă dă click pe numele unui restaurant, de a vedea toate recenziile acestuia ca în Fig. III.13. Totodată, pe lângă recenzii, dacă acel utilizator este autentificat, îi apare câmpul pentru adăugare a unei recenzii, iar dacă nu este autentificat i se afișează un buton de adăugare ca cel din colțul din dreapta jos al fiecărui restaurant.

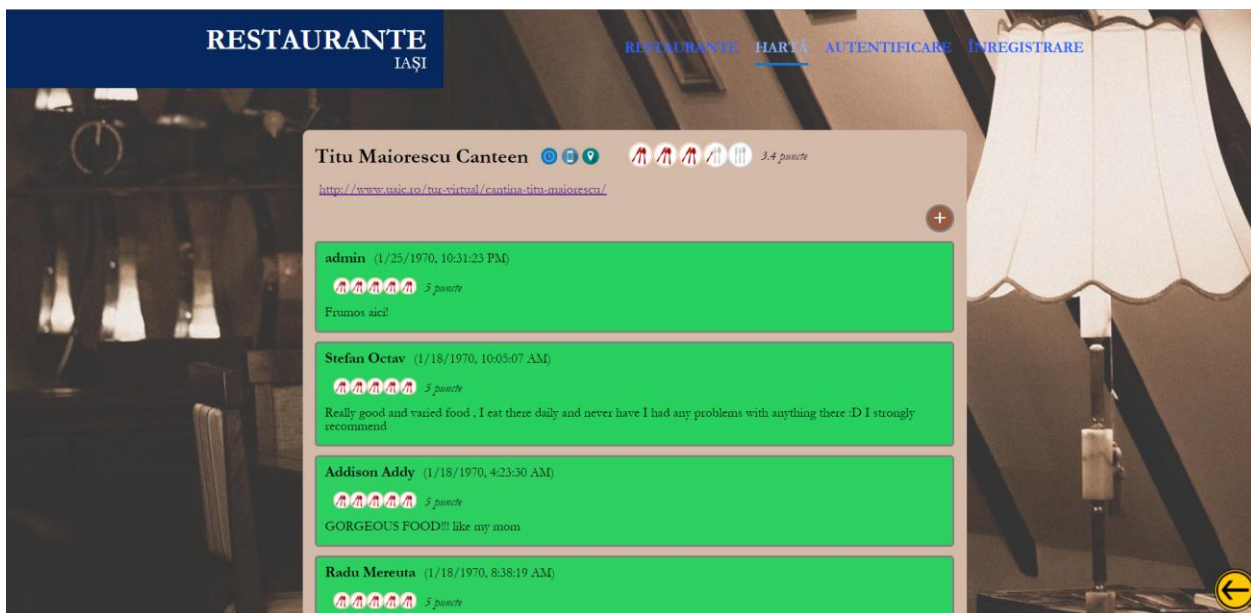


Fig. III.13 Afișarea recenziilor a unui restaurant

Condiția pentru adăugare de recenzii este ca utilizatorul să fie autentificat pentru că altfel, indiferent pe ce butoane se dă click pentru le scrie, dacă el nu este autentificat îi apare modalul pentru autentificare. În momentul în care un utilizator se autentifică îi apar mai întâi, în același

format ca în Fig. III.9, restaurantele la care el a scris recenzii și poate efectua aceleași acțiuni ca cele descrise mai sus, dar cu îndeplinirea condiției de autentificare. Totodată, în cazul în care dorește să scrie o recenzie la un restaurant la care nu a scris până în acel moment, are și opțiunea de a vizualiza toate restaurantele.

b. Componenta Hartă

Componenta pentru hartă reprezintă efectiv o hartă *Google Maps* pe care am folosit-o prin instalarea librăriei *react-google-maps* care este gratuită. Această hartă am folosit-o mai întâi pentru fiecare restaurant în parte. La afișarea informațiilor generale despre un restaurant se afișează și harta pe care este marcată poziția acestuia, iar când se merge cu mouse-ul peste acest marcator apare afișat numele restaurantului. Fiind importată de la *Google Maps*, această hartă este interactivă și se pot efectua același acțiuni obișnuite de mutare, mărire și micșorare, de schimbare a centrului hărții și așa mai departe. A doua folosire a componentei pentru hartă este la afișarea la un loc pe aceasta a tuturor locațiilor restaurantelor cu marcajul corespunzător. La fel ca și la prima utilizare a hărții, și pe aceasta se pot efectua diferite acțiuni, precum și dacă se dă cu mouse-ul peste un marcator apare numele restaurantului împreună cu punctele pe care le-a obținut în urma recenziilor scrise, ca în Fig. III.14.

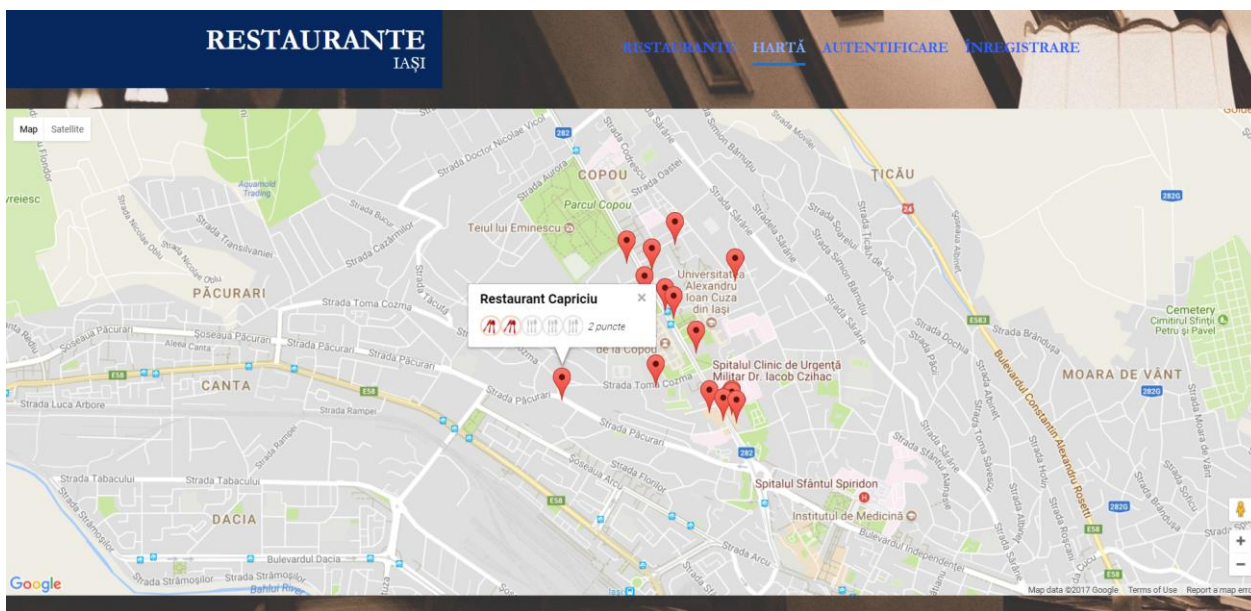


Fig. III.14 Afișarea tuturor marcajelor pentru locațiile restaurantelor

O altă opțiune pe care harta o are este că, atunci când un utilizator este autentificat, i se afișează cu un marcaj diferit restaurantele la care el a scris deja recenzii, cu un marcaj cu steguleț îi apare locația curentă a utilizatorului, iar dacă se dă click pe marcajul oricărui restaurant, i se

afișează acestuia drumul pe care îl are de parcurs cu mașina de la locația curentă până la restaurantul respectiv, ca în Fig. III.15.

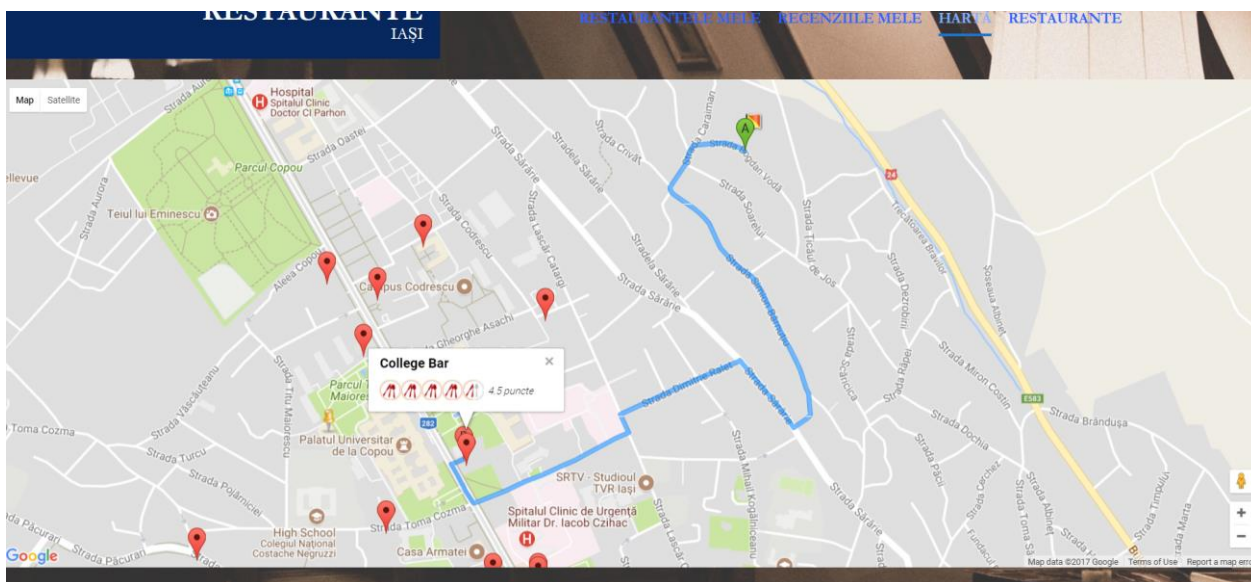


Fig. III.15 Afișarea drumului de la locația curentă la un restaurant

Totodată, dacă în harta în care se afișează toate restaurantele se dă click pe numele unui restaurant, atunci se va afișa restaurantul respectiv împreună cu informațiile și recenziile sale ca în Fig. III.13.

c. Componenta Recenzii

Componenta pentru recenzii este componenta pe care se bazează cel mai mult aplicația. Orice utilizator autentificat are posibilitatea de a scrie recenzii la orice restaurant, indiferent dacă a mai scris sau nu la acel restaurant. Totodată, acesta are posibilitatea de a vedea toate recenziile pe care le-a scris de când și-a creat contul. Așadar, în momentul în care este adăugată o recenzie în câmpul corespunzător ca în Fig. III.16, trebuie selectate numărul de puncte pe care utilizatorul le oferă, precum și textul recenziei pentru că acesta nu poate fi gol. Dacă utilizatorul omite selectarea punctelor, atunci acestea vor fi selectate automat pe punctajul 5.

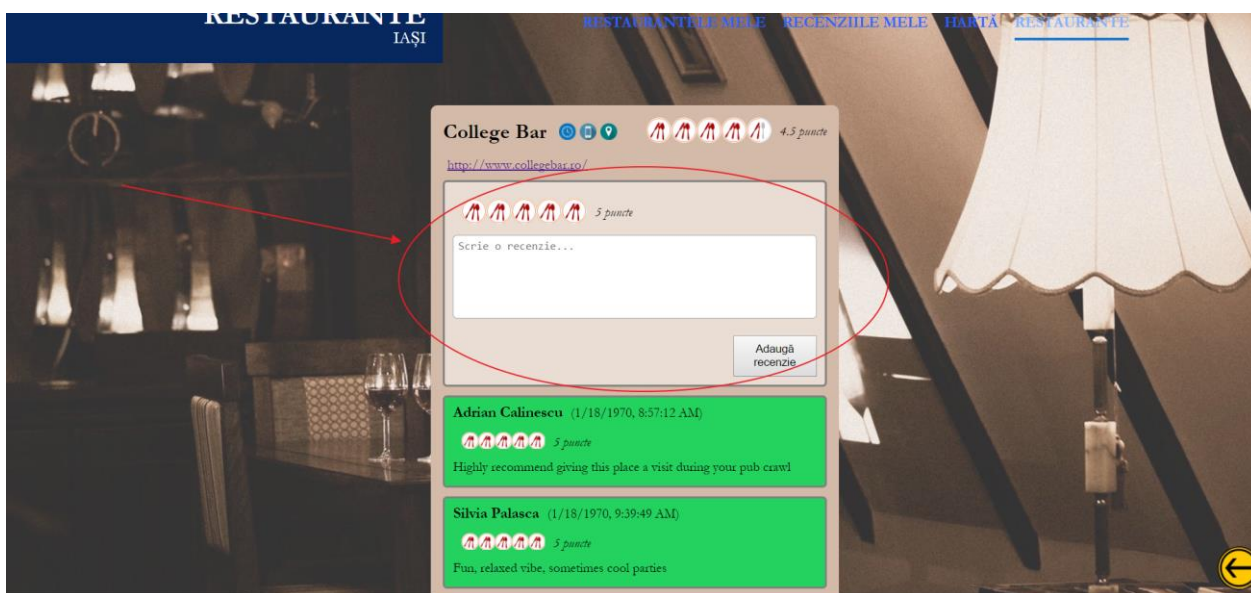


Fig. III.16 Locul unde se scrie o recenzie

Se poate observa că recenziile deja scrise au fundalul colorat. Acest lucru se întâmplă datorită scorului de credibilitate pe care o recenzie îl obține. Acest scor este calculat cu ajutorul algoritmului descris mai sus la subcapitolul *Baza de date*. Așadar mesajul poate fi în orice limbă pentru că el va fi tradus în engleză și apoi i se calculează scorul de credibilitate. Totuși, calcularea scorului se face la server pentru că în client utilizatorul doar apasă pe butonul de adăugare recenzie. O dată calculat, scorul este inserat în baza de date în tabela destinată recenziilor. După ce se apasă pe butonul de adăugare, apare automat și recenzia proaspăt scrisă, cu fundalul colorat. Pentru fiecare recenzie, acest fundal se colorează cu nuanțe de verde sau de roșu în funcție de mesajul ei și numărul de puncte. Astfel dacă mesajul este negativ și rata este tot negativă sau dacă mesajul este pozitiv și rata tot pozitivă atunci recenzia este credibilă și se va avea fundalul în nuanțe de verde. Dacă mesajul este pozitiv și rata este negativă sau dacă mesajul este negativ și rata este pozitivă atunci recenzia nu este credibilă și se va avea fundalul în nuanțe de roșu. Nuanța culorii de fundal este stabilită în funcție de scorul de credibilitate după formula din Fig.III.17.

$$\text{nuanta} = (((\text{scorul_de_credibilitate_formatat} - 1) * 2) / 10;$$

Fig. III.17 Formula de calculare a nuanței de fundal

În Fig.III.17 *scorul_de_credibilitate_formatat* reprezintă scorul de credibilitate dacă el este pozitiv sau scorul de credibilitate înmulțit cu -1 dacă el este negativ. Pentru a pune culoarea de fundal am folosit proprietatea *rgba* specifică stilizării în CSS, unde *r* reprezintă nuanța de roșu, *g* reprezintă nuanța de verde, *b* reprezintă nuanța de albastru, iar *a* reprezintă opacitatea.

Nuanța calculată cu formula din Fig. III.17 reprezintă de fapt opacitatea. Am efectuat această transformarea nuanței în număr pozitiv deoarece nu se acceptă valori negative pentru opacitate.

Toate recenziile afișate vor fi ordonate în funcție de acest scor de credibilitate și în Fig.III.18 se poate observa un exemplu pentru nuanțele despre care am vorbit mai sus.

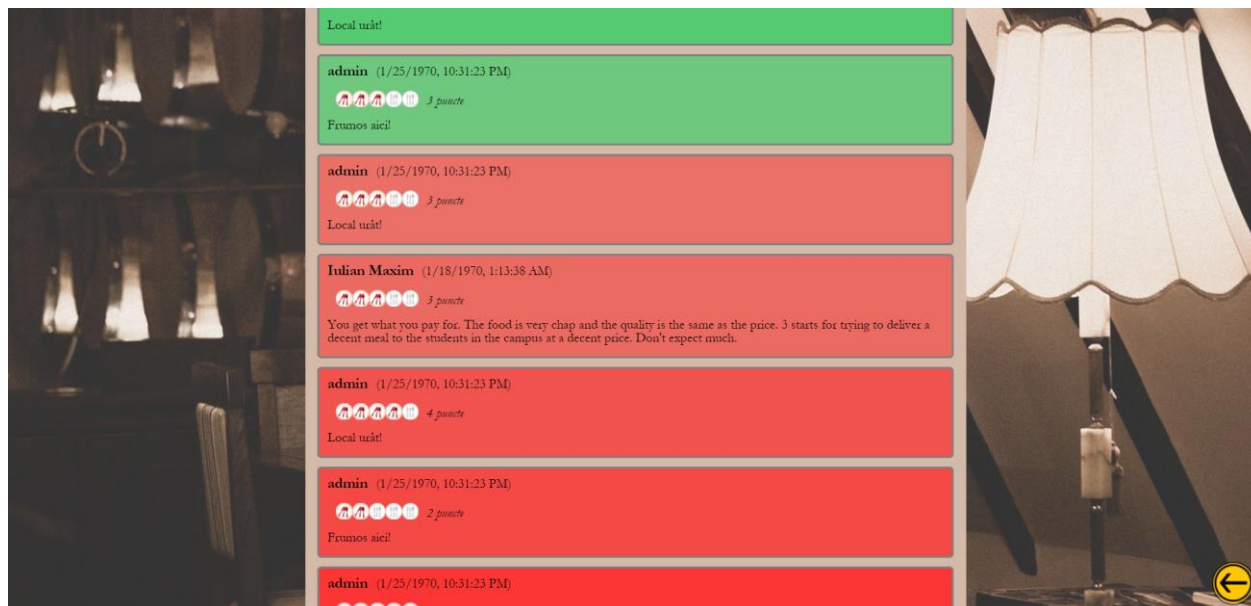


Fig. III.18 Exemplu pentru nuanțele recenziilor

Cap. IV Concluzii

Aplicația *Recenzii Restaurante Iași* este o aplicație pentru scriere de recenzii pentru restaurante. La fel ca aplicațiile similare descrise în capitolul I de mai sus, pentru a putea scrie o recenzie, un utilizator trebuie să aibă cont creat și să fie autentificat. Aceste recenzii conțin un mesaj și un număr de puncte.

Totuși, aplicația *Recenzii Restaurante Iași* este diferită. Ea este concepută în special pentru studenții de la Facultatea de Informatică pentru că deține informații despre restaurantele din jurul acesteia. Este diferită mai ales pentru posibilitatea utilizatorului, autentificat sau nu, de a vizualiza recenziile în funcție de credibilitatea lor. Astfel el poate lua decizia de a merge sau nu în acel local ghidându-se după aceste recenzii.

Totodată, aplicația vine cu posibilitatea utilizatorului autentificat de a vedea drumul pe care îl are de parcurs cu mașina de la locația curentă până la restaurantul pe care îl dorește.

Așadar, aplicația *Recenzii Restaurante Iași* merită consultată înainte ca o persoană să aleagă un local în care să-și petreacă timpul liber.

Bibliografie

- [1] <https://www.google.ro/maps>
- [2] <https://www.google.ro/>
- [3] [https://en.wikipedia.org/wiki/Lars_Rasmussen_\(software_developer\)](https://en.wikipedia.org/wiki/Lars_Rasmussen_(software_developer))
- [3] <https://www.tripadvisor.com/>
- [4] <https://www.boogit.ro/>
- [5] <http://www.brasovultau.ro/articol/stiri/aplicatie-unica-in-lume-creata-de-brasoveni.html>
- [6] <https://www.mysql.com/>
- [7] http://www.cursuri-online.info/php_mysql/CeestePHPsiMySQL.htm
- [8] <https://programam.ro/posts/view/notiuni-de-baza-in-lucrul-cu-mysql>
- [9] http://vega.unitbv.ro/~cataron/Courses/BD/BD_Cap_5.pdf
- [10] <https://dev.mysql.com/doc/refman/5.7/en/sql-syntax.html>
- [11] https://developer.mozilla.org/ro/docs/Web/JavaScript/O_re-introducere_in_JavaScript
- [12] <http://fpce9.fizica.unibuc.ro/jsjava/>
- [13] <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>
- [14] <https://www.slideshare.net/busaco/nodejs-aspecte-esentiale>
- [15] <https://nodejs.org/en/download/>
- [16] <http://www.way2web.ro/tehnologii/node-js/>
- [17] https://www.google.ro/search?q=pagina+nestilizata&rlz=1C1CHBF_enRO731RO731&source=lnms&tbm=isch&sa=X&ved=0ahUKEwj_xKj2kebUAhUBuBQKHVA5DpsQ_AUICigB&biw=950&bih=930#imgsrc=pgadEAOLCsAzXM:
- [18] <http://it.webdesign-galaxy.ro/ce-este-css/>
- [19] Slide 1 din <http://inf.ucv.ro/~mihaiug/courses/web/slides/Curs%204%20-%20CSS.pdf>

[20] https://www.google.ro/search?q=pagina+stilizata+vs+nestilizata&rlz=1C1CHBF_enRO731RO731&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiS4JHskebUAhVEvBQKHSouDrUQ_AUICigB&biw=1920&bih=901#tbm=isch&q=pagina+web+css+design&imgsrc=V6jCvIzUooJgwM:

[21] <http://www.adibaru.ro/2017/05/introductere-instalare-sass.html>

[22] <http://www.c-sharpcorner.com/article/what-and-why-reactjs/>

[23] <http://andrewfarmer.com/component-communication/>

[24] <https://www.ibm.com/watson/developercloud/>