

**UNIVERSITATEA ALEXANDRU IOAN CUZA IAȘI**  
**FACULTATEA DE INFORMATICĂ**



**LUCRARE DE LICENȚĂ**

**Bus Reservation by using Optimal Routes**

propusă de

**Narcis-Constantin Diaconu**

**Sesiunea: Iulie, 2019**

Coordonator științific

**Prof. Colab. Olariu Florin**

**UNIVERSITATEA ALEXANDRU IOAN CUZA IAȘI**  
**FACULTATEA DE INFORMATICĂ**

**Bus Reservation by using Optimal Routes**

**Narcis-Constantin Diaconu**

**Sesiunea:** Iulie, 2019

Coordonator științific

**Prof. Colab. Olariu Florin**

## DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „Bus Reservation by using Optimal Routes”, 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, 26.06.2019

Absolvent Narcis-Constantin Diaconu

---

(semnătura în original)

## Cuprins

<b>Introducere</b>	<b>5</b>
Motivație	5
Noutate	5
<b>Contribuții</b>	<b>7</b>
<b>Capitolul 1. Descrierea problemei</b>	<b>8</b>
<b>Capitolul 2. Abordări anterioare</b>	<b>9</b>
2.1 Bileteria	9
2.2 Teisa	12
<b>Capitolul 3. Descrierea soluției</b>	<b>13</b>
3.1 Aplicația principală	13
3.1.1 JHipster Registry	15
3.1.2 Microservicii	15
3.1.3 JHipster Gateway	17
3.2 Rute optime	19
3.2.1 Algoritmul A*	19
3.2.2 Integrare	20
<b>Capitolul 4. Prezentarea Aplicației</b>	<b>21</b>
4.1 Vizualizare informații rute optime	21
4.2 Rezervarea unui loc	26
4.3 Vizualizare rezervări făcute	28
4.4 Vizualizarea tuturor rezervărilor la un autobuz	29
<b>Concluzii</b>	<b>30</b>
Concluzii generale	30
Direcții viitoare	31
<b>Bibliografie</b>	<b>32</b>
<b>Anexe</b>	<b>33</b>
Anexa 1 - JHipster	33
Anexa 2 - Java Spring	34
Anexa 3 - Python	35

# Introducere

## Motivație

Transportul în comun este un serviciu public de transport disponibil pentru întreaga populație. Majoritatea sistemelor de transport public se derulează de-a lungul unor rute fixe cu puncte de îmbarcare și debarcare setate într-un orar prealabil și percep o taxă pentru fiecare călătorie. Principalele ramuri ale acestui sistem sunt transportul aerian, rutier, feroviar și naval.

Acest serviciu este oferit persoanelor de către diverse companii de transport. În general, fiecare companie de transport se axează pe o anumită ramură, și în România, cele mai multe companii de transport oferă servicii de transport rutier între orașe folosind ca și vehicule autobuze și microbuze. Datorită numărului limitat de locuri, timpului de călătorie ridicat și al distanțelor mari, marea majoritate a companiilor oferă persoanelor posibilitatea de a face rezervări. Rezervarea unui loc în prealabil garantează că va exista un loc în autobuz la data și ora specifice pentru călătoria dorită.

În momentul de față, cele mai multe rezervări se fac prin apel telefonic. Acest lucru necesită ca cel puțin o persoană, din partea companiei, să fie disponibilă telefonic și să gestioneze rezervările.

Doresc realizarea unei aplicații care să simplifice acest proces atât pentru o companie de transport cât și pentru clienți. Aplicația oferă unui utilizator posibilitatea de a afla informații despre autobuzele disponibile la o anumită dată, de a vizualiza ruta pe care autobuzul o va urma pe o hartă și de a face o rezervare la călătoria dorită.

Aplicația oferă și posibilitatea de configurare a rutelor și autobuzelor disponibile și de verificare a rezervărilor făcute până în acel moment la orice autobuz.

## Noutate

Transportul în comun nu acoperă însă toate rutele dorite. În unele cazuri, nu există un mijloc de transport în comun între două locații, iar în alte cazuri orarul mijloacelor de transport disponibile nu este convenabil.

Pentru a rezolva această problemă, aplicația dispune de un algoritm ce generează rute optime. Scopul acestui algoritm este de a oferi utilizatorului o secvență de mijloace de transport în comun pentru a ajunge la destinația dorită în intervalul orar dorit de acesta.

Lucrarea de față este structurată pe 4 capitole ce vor ajuta la prezentarea aplicației, a detaliilor legate de implementare și a tehnologiilor folosite.

În **Capitolul 1** este prezentate principalele probleme pe care aplicația dezvoltată își propune să le rezolve.

**Capitolul 2** prezintă câteva aplicații ce au fost create pentru a rezolva problemele prezentate, modul în care acestea funcționează și principalele funcționalități pe care acestea le oferă.

**Capitolul 3** are rolul de a prezenta aplicația implementată prin detalierea arhitecturii utilizare, a tehnologiilor ce au ajutat în dezvoltarea aplicației și a modului în care aplicația abordează problema.

**Capitolul 4** prezintă principalele funcționalități ce au fost implementate în aplicație și modul în care utilizatorul poate să interacționeze cu aplicație prin diverse exemple și imagini.

În ultima parte vor fi prezentate concluziile lucrării și câteva direcții de dezvoltare în viitor dar și îmbunătățiri prin care aplicația se poate dezvolta.

## Contribuții

Acest proiect îmbină idei personale cu cele ale domnului coordonator în vederea realizării unei aplicații, diferită de cele existente, cu scopul de a facilita rezervarea unui loc la călătoria dorită și de a oferi informații utilizatorilor despre mijloacele de transport în comun disponibile.

Interfața aplicației a fost gândită astfel încât să fie ușor de utilizat și intuitivă, dar în același timp să ofere cât mai multe informații ce pot fi de ajutor utilizatorului.

Principala contribuție este algoritmul de rute optime ce a fost conceput astfel încât să ofere utilizatorului cea mai eficientă soluție pentru a ajunge la destinația dorită, la data dorită. În momentul de față, nu există nici o aplicație în România ce oferă rute optime deci acest algoritm poate fi considerat ceva nou și inovativ.

Menționez că toate tehnologiile, librăriile și framework-urile utilizate în dezvoltarea aplicației sunt open-source.

# Capitolul 1. Descrierea problemei

Prima problemă pe care doresc să o rezolv în acest proiect este problema rezervărilor. După cum am menționat în introducere, în momentul de față cele mai multe rezervări se fac prin apel telefonic. Acest mod de a face o rezervare impune unele restricții cum ar fi disponibilitatea persoanei ce se ocupă de rezervări. Disponibilitatea este influențată atât de un interval orar în care persoana este disponibilă telefonic cât și de faptul că prin apel telefonic se poate comunica cu un singur client, acest lucru influențând timpul de așteptare al altor persoane ce vor să facă o rezervare în acel moment sau să afle anumite informații.

A doua problemă pe care aplicația o abordează este lipsa unui mijloc de transport în comun între două locații. Pentru ca o persoană să găsească o secvență de mijloace de transport în comun trebuie ca aceasta să se informeze în legătură cu toate mijloacele de transport ce pot fi folosite, iar apoi va trebui să le grupeze într-un mod favorabil. Din cauză că acest lucru necesită destul de multă informare, unele persoane preferă să meargă cu primul mijloc de transport disponibil până la o anumită locație, iar acolo va fi nevoie să aștepte un alt mijloc de transport favorabil, câteodată fără a cunoaște în prealabil timpul total de așteptare. Acest lucru poate duce la un timp de așteptare mare, timp ce poate fi considerat timp irosit.



## Capitolul 2. Abordări anterioare

În acest capitol voi prezenta câteva aplicații ce oferă posibilitatea de a face o rezervare online la călătoria dorită.

### 2.1 Bileteria

**Bileteria**<sup>1</sup> este o platformă pentru vânzarea de bilete de transport auto interurban și internațional din România. Această platformă oferă agenților de turism și companiilor de transport posibilitatea de a vinde bilete pentru curse pe întreg teritoriul României printr-un contract. Această platformă este împărțită în trei componente:

Autogări.ro<sup>2</sup> este un site ce oferă informații publicului larg despre programul curselor autobuzelor din România. Pentru a vedea lista de autobuze disponibile, utilizatorul trebuie să completeze un formular ce conține locația de plecare, locația unde dorește să ajungă, ziua în care dorește să călătorească și numărul de locuri. Pe baza acestor informații va fi afișată o listă cu autobuzele disponibile, ordonate după ora de plecare. Autobuzele disponibile pot fi filtrate după ora de plecare și sosire, compania ce deține autobuzul și tipul mijlocului de transport. Pentru fiecare autobuz este disponibil și prețul, acesta fiind calculat în funcție de numărul de locuri completat în formular. La autobuzele la care există posibilitatea de rezervare apare un buton care va redirecționa utilizatorul către următoarea componentă. În Figura 1 se poate observa un rezultat al unei căutări pe Autogări.ro.

---

<sup>1</sup> <https://www.bileteria.ro/>

<sup>2</sup> <https://www.autogari.ro/>

## Autobuze Iași → Piatra Neamț

De la Iași la Piatra Neamț circulă 23 autobuze cu plecare din stațiile Iași - Autogara Transbus Codreanu SRL, Iași - Autogara Iași Vest Metchim SA (Pacurari) și sosire în stațiile Piatra Neamț - Autogara Minut Service Company SRL, Piatra Neamț - Autogara Transmoldavia SA, Piatra Neamț - Stație Hotel Ceahlău, Piatra Neamț - Vinători-fostul REMAT.

Primul autobuz pleacă la 06:00. Ultimul este la 18:00. Călătoria durează peste 2<sup>h</sup>.

Mijloc	Plecare	Durata	Sosire	Companie	Zile circulație	Preț total
	06:30	2 <sup>h</sup> 29'	08:59	FYA TRANS	L M M J V S D	34 lei

**Cumpără bilet**

▼ Informații

### Itinerariu

**A Iași** Autogara Transbus Codreanu SRL

06:30 **FYA TRANS** Traseu: Iași - Roman - Piatra Neamț  
Operator: SC FYA TRANS SRL  
Dotări:

La pasul 3 utilizatorul va trebui să adauge câteva informații personale și de contact. Informațiile cerute pot fi vizualizate în Figura 4. Ultimul pas este plata online a biletului. Acest pas nu este întotdeauna necesar, unele companii oferind posibilitatea unei rezervări fără a obliga clientul să plătească în avans biletul. Dacă plata biletului este necesară atunci utilizatorul are la dispoziție câteva minute pentru a face plata, altfel rezervarea va fi anulată.

## Bilete de autocar Iasi Piatra Neamt

► Pasul 1: Selectie destinatie

► Pasul 2: Curse

▼ Pasul 3: Date călător

### Detalii rezervare

**Tur** ☒ Bilet

**cu Fya Trans**  
**FYA TRANS**

**A** 06:30 Iasi  
Autogara Transbus Codreanu SRL

**B** 09:14 Piatra Neamt  
Autogara Minut Service Company SRL

Marti, 25 Iun 2019

Iasi - Roman - Piatra Neamt

1 locuri

[Conectează-te](#)  
sau  
[Continuă ca Narcis](#)

Adulti **1**

Prenume Nume Mobil Email

1

Tarif tur **Dus** 34

Prețul include TVA de 19%.

Total : 34,00<sup>lei</sup>  
Total de plată acum : 34,00<sup>lei</sup>  
Rest de plată la îmbarcare : 0,00<sup>lei</sup>

Observații

Va rugam sa prezentati biletul pdf pe telefon sau sa listati biletul on-line pentru a-l prezenta la imbarcare! Va rugam sa completati nr. de telefon !

**Termeni și condiții de călătorie**

Fya Trans / Iasi - Roman - Piatra Neamt  
Calatorii trebuie sa se prezinte la microbuz/ autocar cu cel puțin 20 de minute înainte de plecarea in cursa.  
La imbarcare prezentati acest e-Ticket soferului/ dispecerului pentru valabilitate. Va rugam sa pastrati biletul pe toata durata calatoriei pentru a fi prezentat la eventualele controale! Greutatea admisa a bagajelor este de maxim 20 kg /calator platitor de bilet de calatorie. Va rugam sa verificati daca datele si orele de calatorie sunt corecte si daca numele si prenumele corespund cu cele din cartea de identitate. Va dorim calatorie placuta! Echipa FYA TRANS AG IASI: 0754.20.80.80, AG ROMAN: 0749.02.88.89, AG PIATRA NEAMT: 0741.988.901, AG VASLUI: 0754.20.80.20, AG PASCANI: 0751.47.68.74, SECRETARIAT/ RECLAMATII: 0755.121.554

☐ Accept termenii și condițiile de călătorie și generali

[Cumpara bilet](#)

Figura 4. Completarea informațiilor personale<sup>6</sup>

REZMax<sup>7</sup> este sistemul folosit la gestiunea informațiilor publicate pe Autogări.ro și gestiunea rezervărilor și biletelor vândute prin intermediul Bileteria. Acesta poate fi folosit atât de companii de transport cât și de agenții de turism.

<sup>6</sup> Imagine preluată de pe: <https://bileteria.ro/wl.php?&agent=F9741800-5601-4EF8-A319-FF1217AA4073>

<sup>7</sup> <http://www.rezmax.ro/>

## 2.2 Teisa

**Teisa**<sup>8</sup> este o aplicație web, creată de agenția de turism Teisa Travel, ce oferă clienților posibilitatea de a face rezervări online la călătoria dorită. Pentru ca un utilizator să facă o rezervare acesta va trebui să completeze un formular ce se actualizează dinamic pe parcursul completării.

Utilizatorul trebuie să selecteze inițial locația de plecare, destinația, data și ora. După completarea acestor informații utilizatorul poate să selecteze numărul de bilete, în limita locurilor disponibile. Pentru fiecare bilet trebuie completat numele și prenumele persoanei și poate fi selectat tipul de bilet, aplicația oferind reduceri pentru studenți, bătrâni și copii sub 10 ani, și tipul de loc acesta influențând de asemenea prețul.

După aceasta, trebuie completate date de contact despre persoana care va plăti, aplicația oferind posibilitatea plății în numerar sau folosind cardul. Pentru plata cu cardul, aplicația oferă o reducere de 10% și confirmarea directă a locului. În cazul alegerii plății în numerar utilizatorul are obligația de a efectua plata în maximum 24 de ore, altfel rezervarea va fi anulată.

Selectați ruta ⓘ

Plecare din: Iasi Destinație: Otopeni

Selectare dată

Data plecare: 25-06-2019 Ora plecare: 08:00

☐ Vreau bilet dus-intors

Selectare persoane

Număr bilete: 1 Număr locuri disponibile: 2

Detalii călători

Diaconu Narcis Student Normal

Total Cost: 90 RON

PASUL URMĂTOR

Figura 5. Formular principal Teisa<sup>9</sup>

Introdu datele plătitului sau alege un călător:

Diaconu Narcis

Nume: Diaconu Prenume: Narcis Telefon: telefon

Localitate: Localitate Adresă: Adresă Email: Email

Metoda de plată: Card

Beneficiezi de 10% reducere pentru plata online!

TOTAL PLATĂ: 81.0 RON

După efectuarea rezervării aveți 20 de minute la dispoziție să efectuați plata online. După cele 20 de minute rezervarea va fi anulată.

☐ Sunt de acord cu termenii și condițiile de TRANSPORT.

☐ Sunt de acord cu termenii și condițiile LibraPay.

Inapoi Trimite

Figura 6. Confirmare rezervare Teisa<sup>10</sup>

<sup>8</sup> <https://www.teisa.ro/main/#/onlineBooking>

<sup>9</sup> Imagine preluată de pe: <https://www.teisa.ro/main/#/onlineBooking>

<sup>10</sup> Imagine preluată de pe: <https://www.teisa.ro/main/#/onlineBooking>

## Capitolul 3. Descrierea soluției

În acest capitol voi prezenta tehnologiile utilizate în dezvoltarea aplicației, arhitectura utilizată pentru aplicație și principalele funcționalități ce au fost implementate.

În Figura 7 este prezentată o diagramă de context a aplicației. Aici se pot observa principalele sisteme din aplicație: *Bus Reservation System*, *Optimal Routes Cloud Function* și Google Maps API<sup>11</sup>, serviciu oferit de Google.

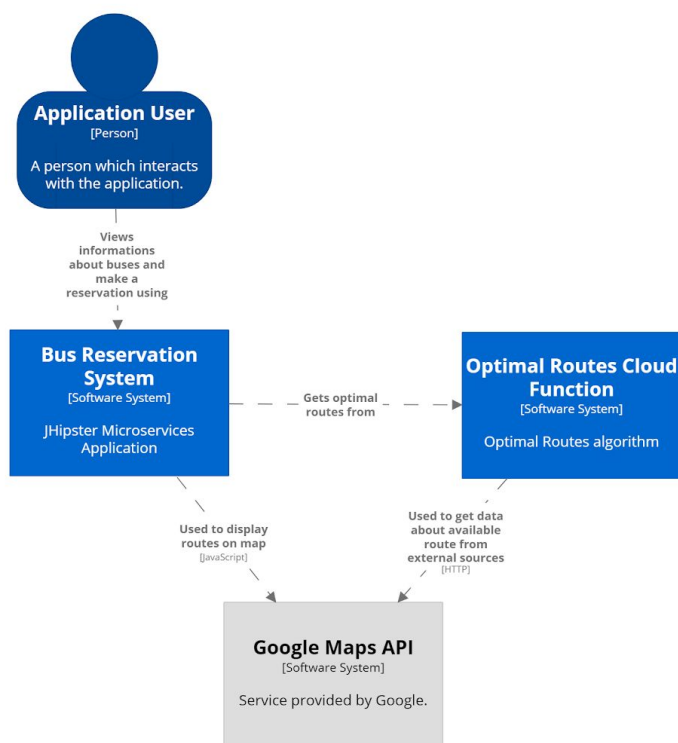


Figura 7. Diagrama de context a aplicației

### 3.1 Aplicația principală

Aplicația principală este reprezentată de componenta Bus Reservation System. Această aplicație folosește o arhitectură pe microservicii și a fost construită folosind generatorul JHipster<sup>12</sup>. Pentru o descriere a generatorului JHipster vizitați Anexa 1.

Am decis să folosesc JHipster pentru aplicația principală deoarece folosind acest generator de aplicații am reușit să creez scheletul de bază al unei aplicații ce folosește o

<sup>11</sup> <https://cloud.google.com/maps-platform/>

<sup>12</sup> <https://www.jhipster.tech/>

arhitectură pe microservicii, cu o interfață prietenoasă și ușor de utilizat, din doar câteva comenzi.

Pentru a putea folosi generatorul JHipster acesta poate fi instalat local folosind managerul de pachete NPM. Pentru ca acesta să poată fi instalat este necesară instalarea unei versiuni de Java (cel puțin 8) și instalarea unei versiuni de Node.js<sup>13</sup>. După instalarea acestora, JHipster poate fi instalat folosind comanda `npm install -g generator-jhipster`. Generatorul JHipster poate fi pornit folosind comanda `jhipster` la linia de comandă. În urma pornirii generatorului JHipster de la linia de comandă va apărea un meniu de unde putem selecta tipul de aplicație pe care dorim să îl generăm.

În Figura 8 este prezentată o diagramă unde se pot observa principalele componente ale aplicației: JHipster Registry, JHipster Gateway, Users Microservice, Stations Microservice, Routes Microservice, Buses Microservice și Tickets Microservice. Fiecare dintre aceste componente rulează separat și comunică cu restul sistemului prin HTTP.

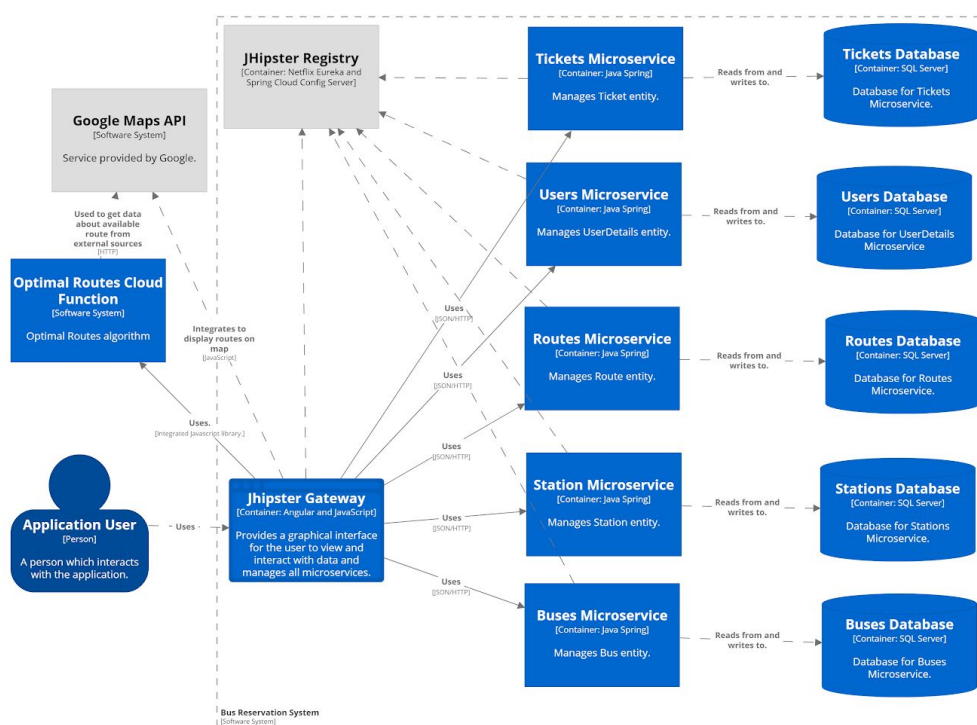


Figura 8. Diagrama de container a sistemului

<sup>13</sup> <https://nodejs.org/en/>



### 3.1.1 JHipster Registry

Registrul JHipster<sup>14</sup> este o aplicație esențială în această arhitectură. Această aplicație are rol de orchestrator, leagă toate celelalte componente și le permite să comunice între ele. Pe lângă rolul de orchestrator acesta oferă și câteva funcționalități suplimentare de monitorizare și de verificare a sănătății aplicațiilor. Aplicația este open-source și este oferită de echipa JHipster. Aceasta poate fi descărcată de pe GitHub<sup>15</sup> și poate fi rulată direct fără să fie nevoie de modificări.

### 3.1.2 Microservicii

Un microserviciu conține codul pentru back-end. Acesta expune un API pentru domeniul în care este implicat. O arhitectură pe microservicii poate conține mai multe astfel de aplicații, fiecare conținând câteva entități înrudite și logica necesară utilizării lor.

La pornirea unui microserviciu acesta se înregistrează în registru și obține configurația de la acesta. Pentru ca un serviciu să rămână conectat la registru, acesta va trimite un semnal periodic, la un interval regulat. Când microserviciul nu mai trimite acest semnal va fi eliminat din registru.

În continuare voi explica cum am generat microserviciile folosind JHipster.

Generarea unui microserviciu începe prin selectarea din meniul generatorului a opțiunii Microservice Application. În continuare trebuie să alegem din câteva opțiuni pe baza cărora microserviciul va fi configurat. Figura 9 prezintă un set complet de opțiuni.

Fișierele generate se află în folderul în care a fost executată comanda de pornire a generatorului.

```
1 Which *type* of application would you like to create? Microservice application
2 What is the base name of your application? tickets
3 As you are running in a microservice architecture, on which port would like your server to run? It should be unique to
  avoid port conflicts. 8081
4 What is your default Java package name? com.application.tickets
5 Which service discovery server do you want to use? JHipster Registry (uses Eureka, provides Spring Cloud Config support
  and monitoring dashboards)
6 Which *type* of authentication would you like to use? JWT authentication (stateless, with a token)
7 Which *type* of database would you like to use? SQL (H2, MySQL, MariaDB, PostgreSQL, Oracle, MSSQL)
8 Which *production* database would you like to use? Microsoft SQL Server
9 Which *development* database would you like to use? H2 with disk-based persistence
10 Do you want to use the Spring cache abstraction? Yes, with the Hazelcast implementation (distributed cache, for multip
  le nodes)
11 Do you want to use Hibernate 2nd level cache? Yes
12 Would you like to use Maven or Gradle for building the backend? Maven
13 Which other technologies would you like to use?
14 Would you like to enable internationalization support? No
15 Besides JUnit and Jest, which testing frameworks would you like to use?
16 Would you like to install other generators from the JHipster Marketplace? No
```

Figura 9. Generarea unui microserviciu folosind JHipster

<sup>14</sup> <https://www.jhipster.tech/jhipster-registry/>

<sup>15</sup> <https://github.com/jhipster/jhipster-registry>

După generarea unui microserviciu putem adăuga entități. O entitate poate fi generată folosind comanda *jhipster entity [nume\_entitate]* în folderul în care se află microserviciul. Pe parcursul generării vom putea adăuga câmpuri la entitate și relații cu alte entități. În urma generării vom obține o implementare a operațiilor CRUD pentru entitatea creată.

Fiecare microserviciu este o aplicație ce este împărțită pe mai multe nivele. În Figura 10 este prezentată o diagramă generală pe componente a unui microserviciu. Aici se pot observa principalele nivele din microserviciu: Controller, Service, Repository și Domain.

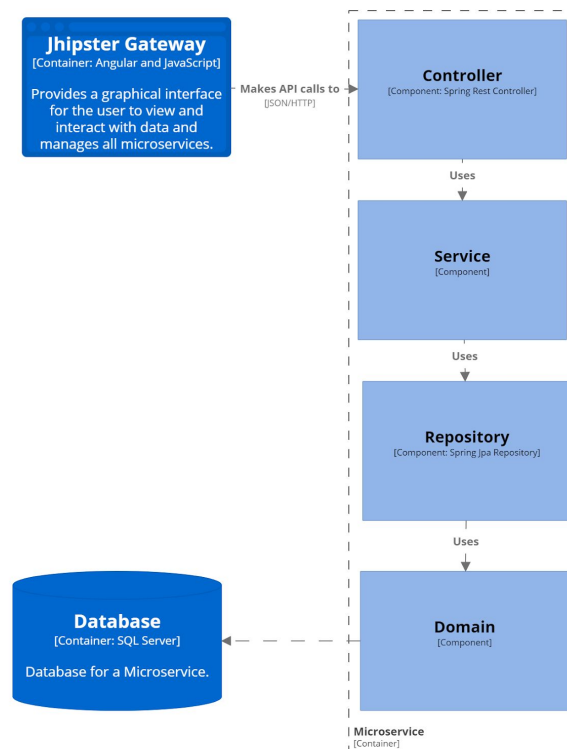


Figura 10. Diagrama generală pe componente a unui microserviciu

- Domain: aici se află clasele de tip entitate;
- Repository: aici se află clasele ce implementează operațiile de preluare și modificare a datelor în baza de date;
- Service: acest nivel permite comunicarea între Controller și Domain și oferă logica aplicației. Acesta este compus din clase de tip Service, Dto și Mapper. O clasă de tip Dto este o clasă ce este folosită pentru a transmite date de la un nivel la altul. O clasă de tip Mapper are rolul de a converti entități la Dto și invers. Clasa de tip service este clasa ce conține logica de bază și este folosită în următorul nivel.



- Controller: Acest nivel se ocupă de comunicarea cu exteriorul. Aici sunt gestionate cererile HTTP și este transmis răspunsul la apelant.

### 3.1.3 JHipster Gateway

Aplicația Gateway este un punct central către toate serviciile conectate. Aceasta va transmite automat toate cererile către microservicii. Dacă există mai multe instanțe ale aceluiași serviciu, aplicația va obține toate acele instanțe, va distribui cererile între instanțe folosind Netflix Ribbon<sup>16</sup> și va acționa ca un întrerupător folosind Netflix Hystrix<sup>17</sup> pentru a elimina sigur și rapid instanțele eșuate. Gateway-ul asigurat de JHipster integrează și Netflix Zuul<sup>18</sup> pentru rutare dinamică, monitorizare, elasticitate și securitate. În această aplicație se află și tot codul pentru interfață.

Generarea inițială a acestei aplicații a fost făcută utilizând JHipster.

Generarea unei aplicații Gateway este un proces asemănător generării unui microserviciu. Principalele diferențe sunt tipul aplicației selectat în meniul generatorului JHipster, acesta fiind acum Microservice Gateway și datorită faptului că această aplicație conține și codul pentru interfață va trebui să selectăm și tehnologiile pe care dorim să le utilizăm pentru acesta. În Figura 11 sunt evidențiate noile opțiuni de configurare ce nu erau disponibile la un microserviciu.

```
Which *type* of application would you like to create? Microservice gateway
What is the base name of your application? Gateway
As you are running in a microservice architecture, on which port would like your server to run? It should be unique to
avoid port conflicts. 8080
What is your default Java package name? com.application.gateway
Which service discovery server do you want to use? JHipster Registry (uses Eureka, provides Spring Cloud Config support
and monitoring dashboards)
Which *type* of authentication would you like to use? JWT authentication (stateless, with a token)
Which *type* of database would you like to use? SQL (H2, MySQL, MariaDB, PostgreSQL, Oracle, MSSQL)
Which *production* database would you like to use? Microsoft SQL Server
Which *development* database would you like to use? H2 with disk-based persistence
Do you want to use Hibernate 2nd level cache? Yes
Would you like to use Maven or Gradle for building the backend? Maven
Which other technologies would you like to use?
Which *Framework* would you like to use for the client? Angular
Would you like to enable *SASS* stylesheet preprocessor? Yes
Would you like to enable internationalization support? No
Besides JUnit and Jest, which testing frameworks would you like to use?
Would you like to install other generators from the JHipster Marketplace? No
```

Figura 11. Generarea unei aplicații Gateway folosind JHipster

Pentru a putea utiliza entitățile create în microservicii și în aplicația Gateway, JHipster oferă posibilitatea generării unui set de pagini pentru operațiile CRUD generate în microserviciu și a unei clase de tip service ce are ca rol comunicarea cu microserviciul în

<sup>16</sup> <https://github.com/Netflix/ribbon>

<sup>17</sup> <https://github.com/Netflix/Hystrix>

<sup>18</sup> <https://github.com/Netflix/zuul>

scopul citirii și modificării datelor. Generarea acestora poate fi făcută folosind aceeași comandă ca la microservicii, doar că de această dată nu mai este nevoie să adăugăm câmpuri și relațiile din nou, ci avem opțiunea de generare a entității dintr-un microserviciu existent.

După cum am menționat mai sus, în această componentă se află și interfața aplicației.

Pentru interfața aplicației am ales să folosesc Angular<sup>19</sup>, acesta fiind un framework de dezvoltare web, open-source, bazat pe Typescript<sup>20</sup> ce poate fi folosit pentru a crea aplicații web foarte interactive.

Am ales să utilizez acest framework pentru interfață, datorită următoarelor funcționalități pe care le oferă:

- injectarea dependențelor;
- șabloane bazate pe o versiune extinsă de HTML;
- rutare, folosind *@angular/routing*;
- cereri Ajax folosind *@angular/common/http*;
- construire de formulare folosind *@angular/forms*;
- structura și arhitectura create special pentru o scalabilitate excelentă a proiectului;

Pentru stilizare am folosit Bootstrap<sup>21</sup> deoarece acesta este unul dintre cele mai populare framework-uri ce poate fi utilizat pentru a obține o aplicație web responsive.

Aici este folosită și o funcționalitate ce face parte din Google Maps și anume Google Maps Javascript API<sup>22</sup>. Acest API este folosit pentru a afișa rutele obținute pe o hartă. Integrarea acestui API a fost făcută prin adăugarea următorului script în fișierul *index.html*.

```
<script  
src="http://maps.googleapis.com/maps/api/js?key=API\_KEY&libraries=geometry">  
</script>
```

---

<sup>19</sup> <https://angular.io/>

<sup>20</sup> <https://www.typescriptlang.org/>

<sup>21</sup> <https://getbootstrap.com/>

<sup>22</sup> <https://developers.google.com/maps/documentation/javascript/tutorial>

## 3.2 Rute optime

Pentru implementarea algoritmului de rute optime a fost utilizat limbajul de programare Python<sup>23</sup>, principalele avantaje ale acestuia fiind timpul de dezvoltare mai mic și structurile de date ușor de utilizat. Mai multe detalii se pot găsi în Anexa 3.

Scopul acestui algoritm este de a oferi o secvență de mijloace de transport în comun pentru ca o persoană să poată ajunge la destinația dorită în cel mai scurt timp posibil.

Problema găsirii unei rute optime între două locații folosind transportul în comun este poate fi considerată un caz special al problemei găsirii celui mai scurt drum. În acest caz, în urma reprezentării datelor într-un graf am obține un graf orientat și ponderat unde nodurile sunt stațiile prin care mijloacele de transport trec, muchiile sunt rutele disponibile și costul unei muchii este reprezentat de timpul necesar parcurgerii acesteia.

### 3.2.1 Algoritmul A\*

Pentru a rezolva această problemă am ales să folosesc algoritmul A\*. Acesta este unul dintre cei mai populari algoritmi pentru găsirea unui drum optim datorită performanței și acurateții oferite. A\* poate fi văzut ca o extensie a algoritmului lui Dijkstra<sup>24</sup>.

Ideea de bază a algoritmului este ca pornind de la un anumit nod să găsim cel mai optim drum către nodul dorit folosind o tehnică best-first. La fiecare pas algoritmul face alegerea de tip best-first pe baza unei funcții de cost numită F. Această funcție este reprezentată ca suma dintre costul deplasării din nodul inițial până în nodul curent, denumit costul G și o estimare a costului deplasării de la nodul curent către nodul final, costul H.

În implementarea de față, costul F este reprezentat de timpul necesar deplasării din punctul inițial până în punctul curent. Acesta este calculat în funcție de timpul de așteptare de la ora selectată de utilizator până la plecare, timpul călătoriei și timpul de așteptare la o anumită locație pe parcursul călătoriei. Pentru fiecare dintre acești timpi a fost folosită o pondere:

- timpul călătoriei are ponderea 1;
- timpul de așteptare la o anumită locație are ponderea 2;
- timpul de așteptare inițial are ponderea 1,5;

---

<sup>23</sup> <https://www.python.org/>

<sup>24</sup> <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

Aceste ponderi au fost alese pentru a minimiza timpul de așteptare ce nu este destinat călătoriei propriu-zise.

Algoritmul implementat face alegerea de tip best-first pe baza funcției descrise mai sus din o listă de stări(open\_set). O stare este reprezentată de o secvență de mijloace de transport în comun, locația în care am ajuns urmând această secvență și timpul la care am ajuns în acea locație(data și ora).

Varianta curentă a algoritmului nu se oprește atunci când am găsit cea mai optimă rută, ci continuă până la găsirea unui număr exact de rute sau la epuizarea mulțimii de stări. Am ales să fac acest lucru pentru a oferi utilizatorului nu doar cea mai optimă rută, ci o listă de rute din care acest poate să aleagă.

Principalele date ce sunt utilizate în algoritm sunt datele existente în aplicație și datele oferite de Google Directions API.<sup>25</sup>

### 3.2.2 Integrare

Datorită faptului că back-end-ul aplicației de bază este creat folosind Java, iar algoritmul de rute optime a fost creat utilizând Python, integrarea acestuia direct în aplicație ar fi fost un proces complicat. Pentru a ușura acest proces am decis să utilizez pentru integrare Google Cloud Function<sup>26</sup>. Principalele avantaje ale acestor funcții sunt lipsa administrării unui server și accesul către algoritm printr-o simplă cerere HTTP.

---

<sup>25</sup> <https://developers.google.com/maps/documentation/directions/start>

<sup>26</sup> <https://cloud.google.com/functions/>

## Capitolul 4. Prezentarea Aplicației

În acest capitol voi prezenta principalele funcționalități ce au fost implementate în aplicație și modul în care utilizatorul poate interacționa cu acestea.

Datorită faptului că această aplicație este o aplicație web, aceasta poate fi accesată utilizând un browser web.

### 4.1 Vizualizare informații rute optime

Una dintre principalele funcționalități ce au fost implementate este oferirea de rute optime. În urma accesării aplicației utilizatorul va vedea pagina principală. În pagina principală se află un formular pe baza căruia se vor genera rutele optime. Informațiile ce trebuie completate în acest formular sunt locația de plecare, destinația, data și ora la care utilizatorul dorește să plece. Pagina principală și formularul sunt prezentate în Figura 12.

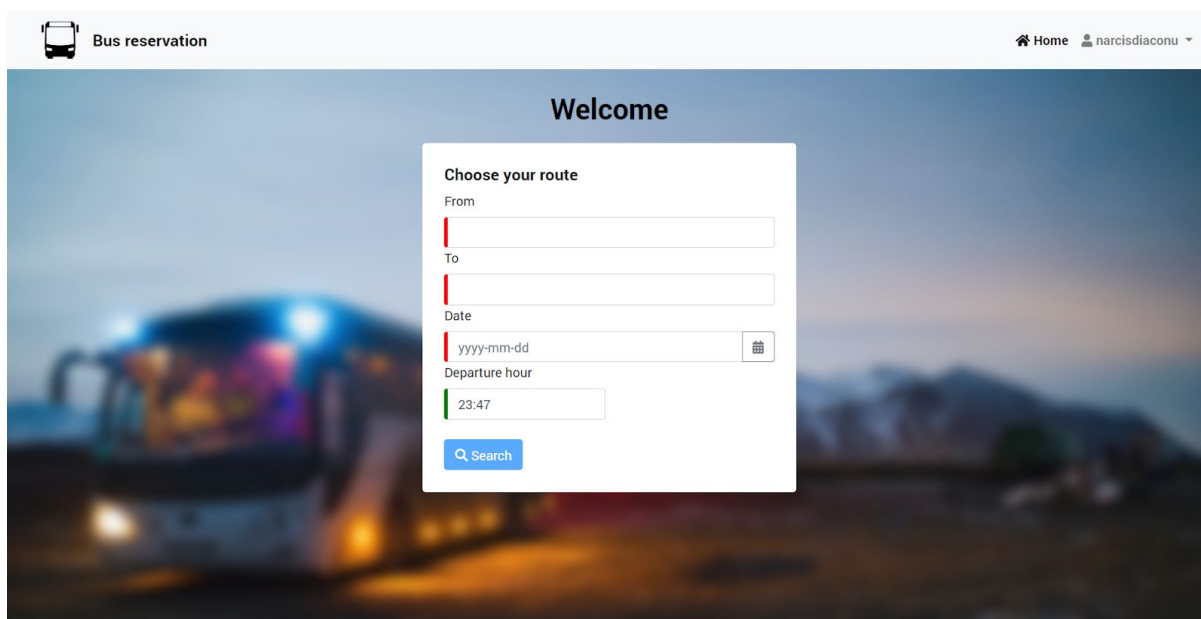


Figura 12. Pagina principală și formularul

După completarea informațiilor și apăsarea butonului *Search*, utilizatorul va fi redirecționat către următoarea pagină.

În această pagină va fi afișat rezultatul algoritmului pentru rute optime. Datorită faptului că durata de rulare a algoritmului poate ajunge la câteva secunde, am decis ca pe parcursul așteptării să afișez un mesaj și o animație pentru a informa utilizatorul.

După primirea rezultatului acesta va fi afișat în pagină. Principalele componente ce sunt afișate în pagină sunt:

- lista de rute găsite, în partea stângă;
- detalii legate de ruta selectată din listă, în centru;
- un rezumat al rutei selectate, în partea de jos, sub detalii;

Figura 13 prezintă această pagină după primirea rezultatului.

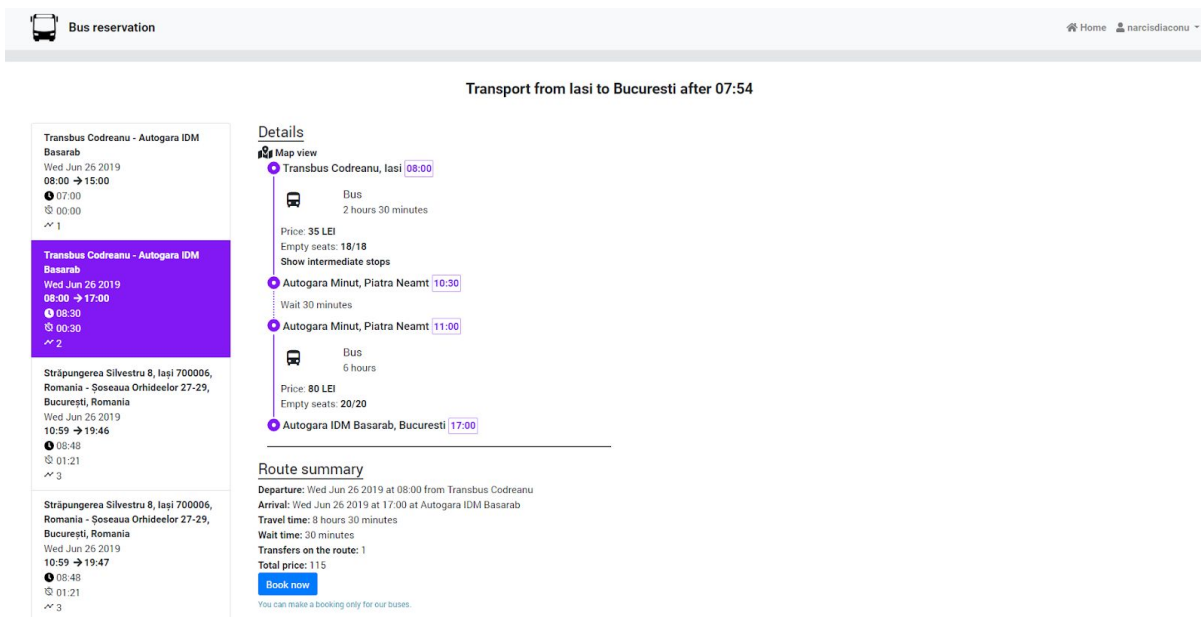


Figura 13. Pagina ce afișează rutele optime

Lista de rute din partea stângă conține rezultatele algoritmului și pentru fiecare rută disponibilă sunt afișate următoarele informații:

- locația exactă de plecare și destinația;
- data plecării;
- ora plecării și ora la care se va ajunge la destinație;
- timpul total necesar deplasării;
- timpul de așteptare la o anumită locație pe parcursul călătoriei;
- numărul de transferuri ce au loc pe acea rută;

Această listă este ordonată după ora la care se ajunge la destinație.

În centru sunt afișate informații legate de fiecare pas ce trebuie urmat pe ruta găsită. Pentru fiecare pas este afișat mijlocul de transport ce este utilizat, durata călătoriei, prețul dacă acesta este disponibil, iar pentru autobuzele ce au fost adăugate în aplicație se pot vedea opririle intermediare, numărul de locuri disponibile și numărul de locuri totale. Figura 13

conține un exemplu de rută ce a fost obținută prin combinarea a două autobuze disponibile în aplicație.

Pentru mijloacele de transport ce sunt oferite de alte companii de transport este afișat un url către site-ul companiei, numărul de opriri pe parcursul călătoriei și un număr de telefon de contact. Figura 14 prezintă un exemplu de astfel de rezultat.

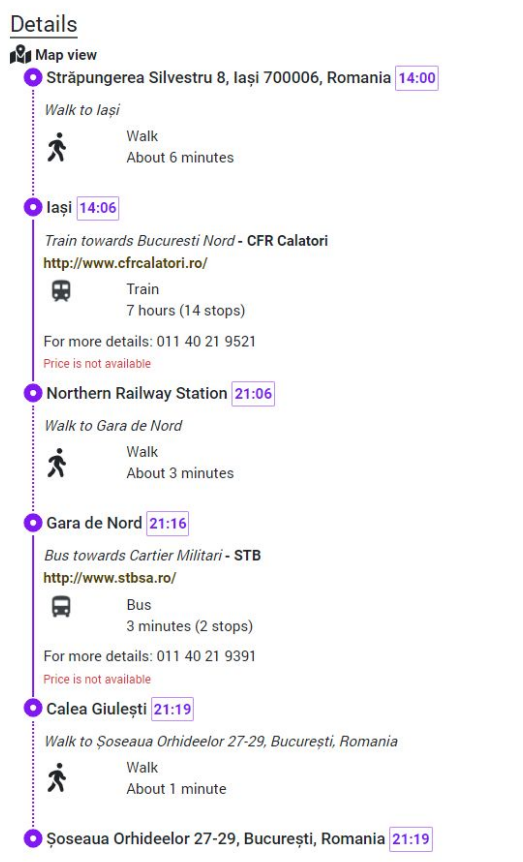


Figura 14. Exemplu rezultat ce folosește mijloace de transport ce nu sunt adăugate în aplicație

Ruta selectată poate fi vizualizată și pe hartă prin apăsarea butonului *Map view* situat sub titlul *Details*. În urma apăsării butonului, detaliile text vor fi înlocuite de o hartă oferită de Google și pe această hartă va fi afișată inițial ruta completă. În Figura 15 se poate observa o rută afișată pe hartă.



## Details

### Text view

Select a step to focus on map

Entire route

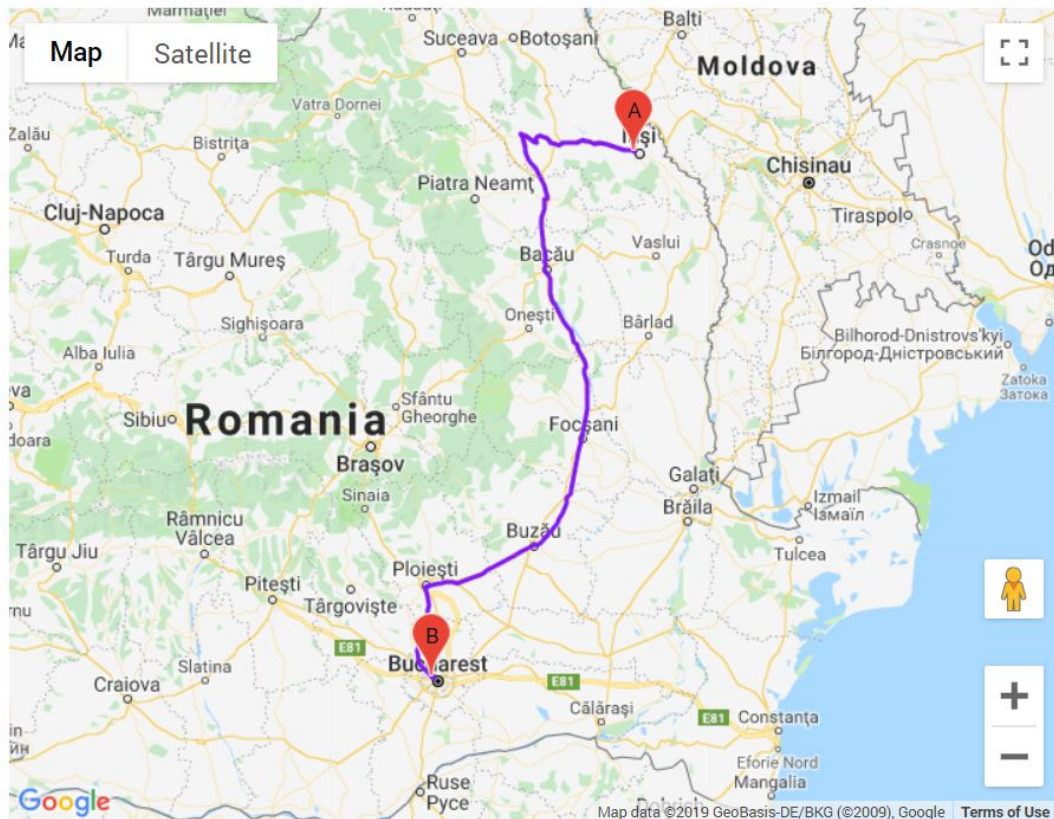


Figura 15. Vizualizare rută pe hartă

Deoarece unele rute sunt compune din mai mulți pași am oferit posibilitatea vizualizării fiecărui pas, individual, pe hartă. Pasul afișat poate fi selectat din lista disponibilă deasupra hărții. În Figura 16 este afișat un pas ce face parte din ruta afișată în Figura 15.

În unele cazuri, pentru a te deplasa de la o stația la alta cel mai optim mod este deplasarea pe jos. Pentru acest caz, deasupra hărții este afișat fiecare pas ce trebuie urmat pentru a ajunge la destinație. Un exemplu este afișat în Figura 16.



## Details

### Text view

Select a step to focus on map

Walk: Walk to Gara de Nord

Instructions:

Head **southwest** on **Piața Gării de Nord**

Turn **right**

Turn **left**

Turn **left**

Take the stairs

Turn **right**

Walking directions are in beta. Use caution – This route may be missing sidewalks or pedestrian paths.

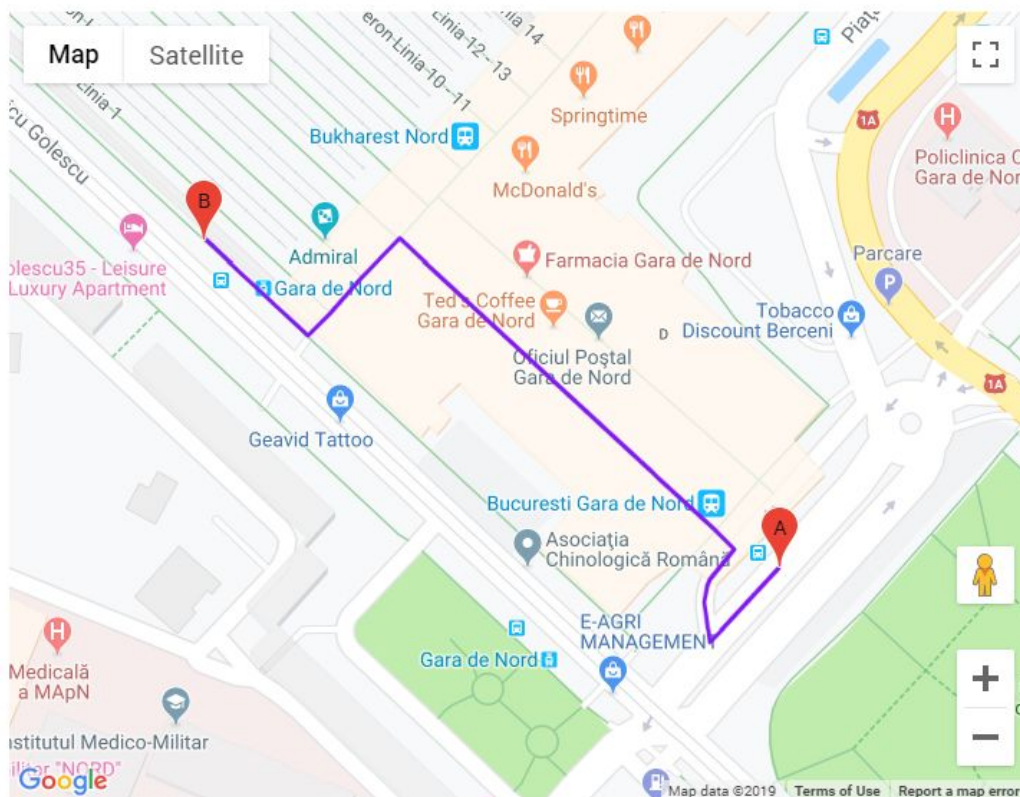


Figura 16. Afișarea unui singur pas și a instrucțiunilor de mers

Pentru ruta selectată este afișat și un rezumat, sub detaliile rutei. Rolul acestui rezumat este de a oferi utilizatorului o privire de ansamblu asupra informațiilor ce sunt afișate în detalii. În Figura 17 este afișat un rezumat al unei rute.

## Route summary

**Departure:** Wed Jun 26 2019 at 08:00 from Transbus Codreanu

**Arrival:** Wed Jun 26 2019 at 17:00 at Autogara IDM Basarab

**Travel time:** 8 hours 30 minutes

**Wait time:** 30 minutes

**Transfers on the route:** 1

**Total price:** 115

Figura 17. Exemplu de rezumat al unei rute

Acest rezumat este așezat sub detalierea rutei, acest lucru putând fi văzut în Figura 13.

Principalele informații ce sunt afișate aici sunt:


- data, ora și locația plecării;
- data, ora și locația în care se ajunge;
- timpul total al călătoriei;
- timpul total de așteptare între anumiți pași;
- numărul de transferuri ce trebuie făcute pe parcursul rutei;
- prețul total(dacă prețul pentru un anumit pas nu este disponibil, atunci va fi afișat un mesaj de informare cu privire la acest lucru);

Sub rezumat se află butonul *Book now*. Acest buton trimite utilizatorul către următoarea funcționalitate și anume rezervarea unui loc. Butonul poate fi utilizat doar dacă ruta selectată conține cel puțin un autobuz disponibil în aplicație.

## 4.2 Rezervarea unui loc

A doua funcționalitate principală este rezervarea unui loc la călătoria dorită. Această funcționalitate poate fi accesată din pagina în care sunt afișate rutele optime prin apăsarea butonului *Book now*. După apăsarea butonului, din ruta curentă vor fi extrase autobuzele la care se poate face rezervare și utilizatorul este redirecționat către următoarea pagină.

În această pagină sunt afișate autobuzele ce au fost extrase din pagina precedentă și conține un formular în care utilizatorul trebuie să adauge câteva informații personale și de contact. În Figura 18 este prezentat un exemplu pentru această pagină.

 Bus reservation

Home narcisdiaconu

Booking details

Buses

From	Iasi Transbus Codreanu	08:00	Wed Jun 26 2019
To	Piatra Neamt Autogara Minut	10:30	Wed Jun 26 2019
Price per ticket: 35 Remaining seats: 18			

From	Piatra Neamt Autogara Minut	11:00	Wed Jun 26 2019
To	Bucuresti Autogara IDM Basarab	17:00	Wed Jun 26 2019
Price per ticket: 80 Remaining seats: 20			

Person details

First Name	Last Name	Email	Phone number	Address
Narcis	Diaconu	narcisdiaconu998@gmail	0751795744	Iasi

Number of tickets:

-

2

+

Total price:

230

Your information will be saved for future bookings.  
You can view and update your data on [your profile page](#).

Book

Previous page

Figura 18. Exemplu pagină rezervări

Pentru fiecare autobuz ce a fost extras din pagina anterioară sunt afișate următoarele informații:

- data, ora și locația de plecare;
- data, ora și locația unde ajunge autobuzul;
- prețul per loc;
- numărul de locuri libere rămase;

Sub acest informații se află formularul pe care utilizatorul trebuie să îl completeze pentru a putea face rezervarea. Aceste informații sunt necesare pentru a putea identifica persoana ce a făcut rezervarea. Informațiile sunt salvate pentru viitoare rezervări și pot fi modificate oricând de către utilizator din pagina de profil.

Utilizatorul poate selecta numărul de locuri pe care dorește să îl rezerve, în limita numărului de locuri disponibile, iar prețul va fi calculat automat.

Confirmarea rezervării se face prin apăsarea butonului *Book*, iar dacă rezervarea a fost creată cu succes va fi afișat un mesaj de confirmare. În Figura 19 este prezentat mesajul de confirmare. După acesta utilizatorul se poate întoarce la pagina principală sau se poate duce la pagina de rezervări.

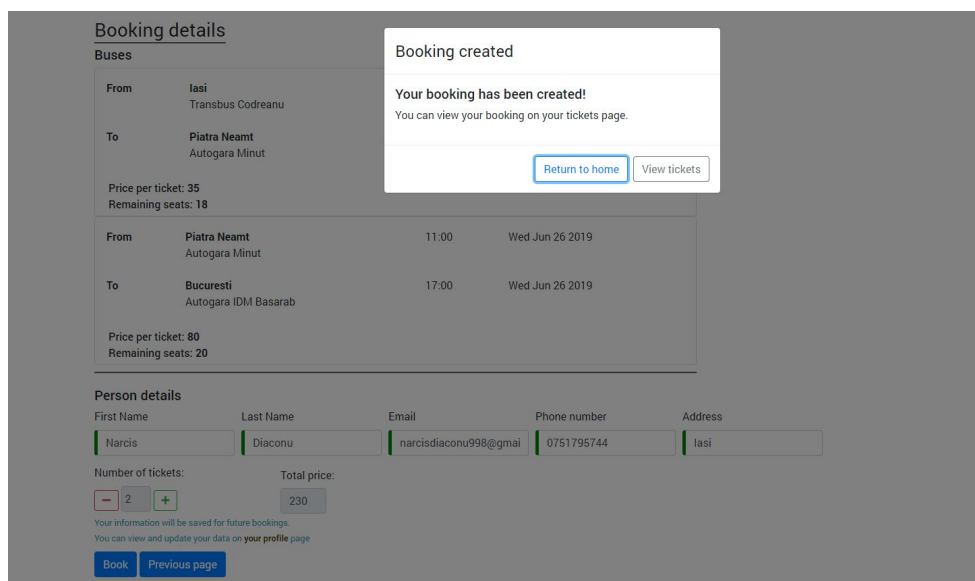


Figura 19. Mesajul de confirmare a rezervării

### 4.3 Vizualizare rezervări făcute

Utilizatorul poate vizualiza în orice moment rezervările făcute prin accesarea paginii *Tickets* din submeniul ce poate fi accesat din bara de navigare prin apăsarea pe numele utilizatorului.

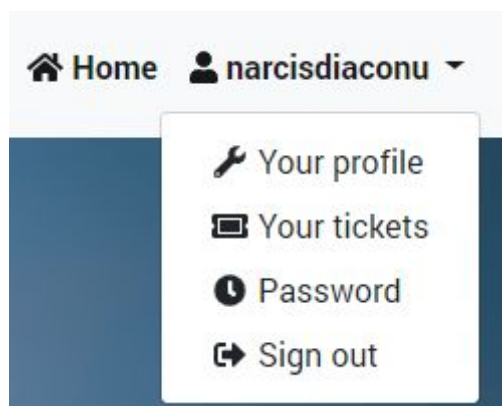





Figura 20. Submeniul utilizatorului

În această pagină este afișată o listă cu toate rezervările făcute de utilizator. În Figura 21 este prezentată pagina de rezervări a unui utilizator.



Bus reservation

 Home

 narcisdiaconu

Your tickets

From	To	Date	Hour	Places	Price	Paid	
Transbus Codreanu, Iasi	Autogara IDM Basarab, Bucuresti	Jun 24, 2019	08:00	1	80	No	<div>X Delete ticket</div>
Transbus Codreanu, Iasi	Autogara Minut, Piatra Neamt	Jun 26, 2019	08:00	2	70	No	
Autogara Minut, Piatra Neamt	Autogara IDM Basarab, Bucuresti	Jun 26, 2019	11:00	2	160	No	


@ 2019 Bus reservation by using optimal routes

Figura 21. Pagina de rezervări a unui utilizator

După cum se poate observa în imagine, pentru unele rezervări este afișat un buton de ștergere. Utilizatorul poate șterge o rezervare doar dacă acea rezervare a fost făcută pentru o zi anterioară zilei curente.

## 4.4 Vizualizarea tuturor rezervărilor la un autobuz

Aplicația oferă și posibilitatea vizualizării a tuturor rezervărilor pentru un anumit autobuz, la o anumită dată. Acest lucru poate fi făcut doar prin intermediul unui cont de administrator. În urma autentificării în aplicație cu un cont ce are rol de administrator, în bara de navigare va apărea un nou meniu, *Administration*. Pentru a putea vedea rezervările, trebuie accesat *Tickets* din acel meniu.



Bus reservation

[Home](#)

[Administration](#)

[Application](#)

[administrator](#)

Tickets

Choose route

Piatra Neamt - Bucuresti

Choose bus

11:00

Choose date

2019-06-26

Person	Email	Phone	Places	Price	Paid	Start Station	End Station
Narcis Diaconu	narcisdiaconu998@gmail.com	0751795744	2	160	No	Autogara Minut	Autogara IDM Basarab
Ion Popescu	popescu_ion@gmail.com	0748958292	1	80	No	Autogara Minut	Autogara IDM Basarab
Codrut Amariei	codrut@gmail.com	0728494928	1	80	No	Autogara Minut	Autogara IDM Basarab

@ 2019 Bus reservation by using optimal routes

Figura 22. Pagina de rezervări pentru un autobuz

Pentru a vedea rezervările pentru un autobuz, trebuie selectată inițial ruta, apoi este selectat autobuzule, pe baza orei de plecare, iar apoi data. După selectarea acestora, vor fi afișate rezervările ce respectă filtrarea. Figura 22 prezintă un exemplu.

Pe lângă vizualizarea rezervărilor, un cont de administrator poate configura și rutele și autobuzele din aplicație. Pentru fiecare entitate există pagini ce oferă posibilitatea creării și modificării a datelor existente. Aceste pagini pot fi accesate din meniul *Administration*.

# Concluzii

## Concluzii generale

După cum am menționat în Introducere, scopul acestui proiect a fost de a crea o aplicație care să simplifice rezervările pentru autobuze și care să rezolve problema lipsei unui mijloc de transport într-un orar favorabil între anumite locații.

Am reușit, cu ajutorul tehnologiilor utilizate și a algoritmului implementat, să creez o aplicație ce permite unei persoane să vadă cele mai optime trasee pentru a putea ajunge la destinația dorită în funcție de data și ora dorită. În plus, persoana poate să facă și o rezervare pentru cursele disponibile, pentru a avea un loc la călătoria dorită.

## Direcții viitoare

Această aplicație poate fi dezvoltată atât pe partea de rute optime cât și pe partea sistemului de rezervări.

Algoritmul de rute optime poate fi îmbunătățit pentru a oferi rezultate în funcție de preferințele utilizatorului, cum ar fi:

- posibilitatea de a căuta rute și în funcție de ora la care persoana dorește să ajungă la destinația dorită;
- posibilitatea de a oferi cele mai optime rute doar în funcție de durata totală a călătoriei sau de timpul de așteptare;
- posibilitatea de filtrare a mijloacelor de transport, în funcție de tip, pe care algoritmul să le includă în rezultatul final;
- posibilitatea de optimizare și în funcție de prețul total al călătoriei, ținându-se cont și de vârsta persoanei(copil, elev, student, bătrân);
- posibilitatea de oferire de rute optime, pornind de la locația, data și ora curentă a utilizatorului până la destinația dorită;

Pe lângă îmbunătățirile ce pot fi aduse la algoritm, se pot aduce și îmbunătățiri la sistemul de rezervări, pentru a asigura un nivel cât mai mare de încredere pentru ambele părți.

Pentru sistemul de rezervări poate fi adăugată posibilitatea de predicție a numărului de locuri ce va fi ocupat, în funcție de numărul de rezervări ce au fost făcute anterior, pe o

anumită perioadă. Astfel o companie se poate pregăti pentru un număr mai mare de rezervări în anumite perioade și, la nevoie, poate crește numărul de locuri disponibile, sau, în timp, poate renunța la autobuzele ce nu sunt frecventate.

Pe lângă cele menționate, pentru sistemul de rezervări ar putea fi adăugate următoarele funcționalități:

- integrarea cu un sistem de plată online, pentru a oferi clientului posibilitatea de a plăti direct din aplicație biletul;
- posibilitatea de a alege locul exact din autobuz în momentul în care utilizatorul dorește o rezervare;
- un sistem de reduceri;
- un sistem de rapoarte ce poate fi folosit pentru a identifica cele mai căutate curse;
- posibilitatea de exportare a rezervărilor pentru un autobuz la o anumită dată în vederea printării listei;

# Bibliografie

1. Sendil Kumar N, Deepu K Sasidharan: *Full Stack Development with JHipster*(2018)
2. Bileteria <https://www.bileteria.ro/>
3. Autogari.ro <https://www.autogari.ro/>
4. REZMax <http://www.rezmax.ro/>
5. Teisa <https://www.teisa.ro/main/#/onlineBooking>
6. Public Transport [https://en.wikipedia.org/wiki/Public\\_transport](https://en.wikipedia.org/wiki/Public_transport)
7. <https://blog.turnit.com/differences-between-bus-ticket-reservation-system-and-bus-ticketing-system>
8. JHipster <https://www.jhipster.tech/>
9. Introducing JHipster <https://spring.io/blog/2015/02/10/introducing-jhipster>
10. JHipster Microservice Architecture <https://www.baeldung.com/jhipster-microservices>
11. Microservices with JHipster <https://www.jhipster.tech/microservices-architecture/>
12. The JHipster Registry <https://www.jhipster.tech/jhipster-registry/>
13. JHipster Gateway <https://www.jhipster.tech/api-gateway/>
14. Angular vs React <https://programmingwithmosh.com/react/react-vs-angular/>
15. Angular  
<https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d>
16. Bootstrap <https://getbootstrap.com/docs/4.3/getting-started/introduction/>
17. A\* algorithm <https://www.geeksforgeeks.org/a-search-algorithm/>
18. A\* search <https://brilliant.org/wiki/a-star-search/>
19. Python language advantages  
<https://www.geeksforgeeks.org/python-language-advantages-applications/>
20. Python language advantages and disadvantages  
<https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121>
21. Spring Framework [https://www.tutorialspoint.com/spring/spring\\_overview.htm](https://www.tutorialspoint.com/spring/spring_overview.htm)
22. Java Spring [https://en.wikipedia.org/wiki/Spring\\_Framework](https://en.wikipedia.org/wiki/Spring_Framework)
23. Spring Boot <https://spring.io/projects/spring-boot>
24. Spring Data <https://spring.io/projects/spring-data>



# Anexe

## Anexa 1 - JHipster

JHipster<sup>27</sup> este un generator de aplicații, open-source, folosit pentru a dezvolta rapid aplicații web moderne și microservicii folosind Angular sau React<sup>28</sup> pentru front end și Java Spring pentru back end. JHipster a devenit foarte popular pe GitHub într-o perioadă foarte scurtă de timp și a fost prezentat în reviste online și în conferințe în întreaga lume.

JHipster se concentrează pe generarea de aplicații de înaltă calitate cu un back end Java folosind o multitudine de tehnologii Spring(*Spring Boot*, *Spring Security*, *Spring Data*, *Spring MVC* și altele) un front end Angular sau React și un număr mare de instrumente de dezvoltare preconfigurate precum *Yeoman*, *Maven*, *Gradle*, *Gulp.js*, *Bower*.

JHipster poate fi folosit pentru a genera atât aplicații cu arhitectură monolitică cât și aplicații cu arhitectură pe microservicii. O arhitectură monolitică folosește o singură aplicație care conține atât codul pentru front end cât și codul pentru back end. O aplicație cu arhitectură pe microservicii împarte front-end-ul și back-end-ul astfel încât aplicația să fie mai ușor scalabilă.

În Figura 23 se pot observa toate tehnologiile ce pot fi utilizate într-o aplicație cu arhitectură pe microservicii folosind JHipster. Componentele cu verde sunt componentele specifice aplicației iar cele cu mov oferă infrastructura de bază.

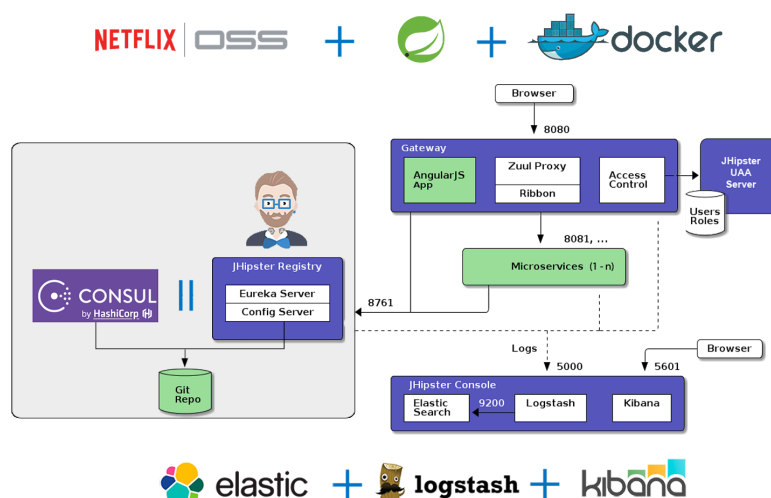


Figura 23. Arhitectura pe microservicii JHipster detaliată<sup>29</sup>

<sup>27</sup> <https://www.jhipster.tech/>

<sup>28</sup> <https://reactjs.org/>

<sup>29</sup> <https://www.jhipster.tech/microservices-architecture/>

## Anexa 2 - Java Spring

Spring<sup>30</sup> este o platformă open-source ce poate fi folosită pentru a simplifica scrierea aplicațiilor în Java. Acesta poate fi folosit pentru orice aplicație Java și a devenit popular în comunitatea Java ca o completare sau chiar o înlocuire a modelului Enterprise JavaBeans.

Platforma Spring oferă flexibilitate în configurarea unei clase de tipul JavaBean folosind fișiere XML, adnotări sau JavaConfig. Din cauza faptului că numărul de funcționalități a crescut și platforma a devenit mai complexă, configurarea unei aplicații Spring a devenit un proces costisitor.

Spring Boot este o extensie a framework-ului Spring ce simplifică configurarea unei aplicații. Acesta dispune de un sistem inteligent de configurare automată ce configurează aplicația pe baza dependențelor jar adăugate. Spring Boot dispune și de un server incorporat ce permite rularea directă a aplicației.

Spring Security este un framework utilizat pentru securizarea aplicațiilor bazate pe Spring. Acesta se axează pe autentificarea și autorizarea aplicațiilor Java și poate fi foarte ușor extins pentru a îndeplini cerințe personalizate.

Spring Data este o parte din Spring ce are ca scop reducerea cantității de cod necesară implementării nivelului ce se ocupă de accesul la date pentru diverse tipuri de baze de date.

---

<sup>30</sup> <https://spring.io/>

## Anexa 3 - Python

Python<sup>31</sup> este un limbaj de programare cu scop general, interpretat, de nivel înalt ce suportă mai multe tipuri de paradigme de programare inclusiv programare imperativă, programare procedurală, programare orientată obiect și programare funcțională.

În python, timpul de dezvoltare este mai mic în comparație cu alte limbaje de programare deoarece python este un limbaj interpretat, ceea ce exclude necesitatea de a compila codul înainte de executare și abstractizează multe detalii sofisticate din codul de programare. În plus, codul scris în python tinde să fie mai mic decât același cod scris în alte limbaje de programare.

Deși python oferă un timp mai scurt de dezvoltare, acesta este mai încet în timpul de execuție față de alte limbaje. Acest dezavantaj este datorat faptului că python este un limbaj interpretat, ceea ce necesită ca înainte de execuție codul să fie analizat, compilat și interpretat, și faptului că python este tipizat dinamic ceea ce necesită ca tipul variabilelor să fie determinat de fiecare dată când variabila este folosită pe parcursul rulării programului.

---

<sup>31</sup> <https://www.python.org/>