



UNIVERSITATEA DIN CRAIOVA  
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI  
ELECTRONICĂ  
DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA  
INFORMAȚIEI



## **PROIECT DE DIPLOMĂ**

Popescu Alexandru-Iulian

### **COORDONATOR ȘTIINȚIFIC**

Șef lucrări dr. ing. Sboră Cătălin

IULIE 2022

CRAIOVA



UNIVERSITATEA DIN CRAIOVA  
FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI  
ELECTRONICĂ  
DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA  
INFORMAȚIEI



# **Platforma pentru intermedierea activităților de curierat folosind tehnologiile Cloud**

Popescu Alexandru-Iulian

**COORDONATOR ȘTIINȚIFIC**

Șef lucrări dr. ing. Sora Cătălin

IULIE 2022

CRAIOVA

„Căutăm adevărul, dar nu găsim decât incertitudine. Căutăm fericirea, dar nu găsim decât nefericire și mizerie. Suntem incapabili de a nu dori adevărul și fericirea; însă nu suntem capabili nici de certitudine, nici de fericire”. (**Blaise Pascal-Scieri alese**)

## DECLARAȚIE DE ORIGINALITATE

Subsemnatul Popescu Alexandru-Iulian, student la specializarea Calculatoare cu predare în limba română din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoștință de cele prezentate mai jos și că îmi asum, în acest context, originalitatea proiectului meu de licență:

- cu titlul Platforma pentru intermedierea activităților de curierat folosind tehnologiile Cloud,
- coordonată de Șef lucrări dr. ing. Sboră Cătălin ,
- prezentată în sesiunea Iulie 2022

La elaborarea proiectului de licență, se consideră plagiat una dintre următoarele acțiuni:

- reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele și referința precisă,
- redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
- prezentarea unor date experimentale obținute sau a unor aplicații realizate de alți autori fără menționarea corectă a acestor surse,
- însușirea totală sau parțială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situații neplăcute se recomandă:

- plasarea între ghilimele a citatelor directe și indicarea referinței într-o listă corespunzătoare la sfârșitul lucrării,
- indicarea în text a reformulării unei idei, opinii sau teorii și corespunzător în lista de referințe a sursei originale de la care s-a făcut preluarea,
- precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
- precizarea referințelor poate fi omisă dacă se folosesc informații sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data,

Semnătura candidatului,



UNIVERSITATEA DIN CRAIOVA  
Facultatea de Automatică, Calculatoare și Electronică  
Departamentul de Calculatoare și Tehnologia Informației

Aprobat la data de .....  
Șef de departament,  
Prof. dr. ing.  
Nicolae Enescu

## PROIECTUL DE DIPLOMĂ

Numele și prenumele studentului/-ei:	Popescu Alexandru-Iulian
Enunțul temei:	Platforma pentru intermedierea activităților de curierat folosind tehnologiile Cloud
Datele de pornire:	Modelul pe care a fost bazata dezvoltarea acestui proiect este reprezentat de procesul de curierat actual si de nevoia a mai multor opțiuni de transport rapid
Conținutul proiectului:	Primul capitol conține descrierea generala a aplicației Al doilea capitol prezinta arhitectura sistemului implementat, baza de date si tehnologiile Cloud folosite. Capitolul 3: Include toate funcționalitățile aplicației Capitolul 4: Cuprinde manualul utilizatorului Capitolul 5: Este reprezentat de concluzii Capitolul 6: Bibliografie
Material grafic obligatoriu:	Diagrame, scheme, capturi de ecran
Consultații:	Periodice
Conducătorul științific (titlul, nume și prenume, semnătura):	Șef lucrări dr. ing. Sboră Cătălin
Data eliberării temei:	
Termenul estimat de predare a proiectului:	
Data predării proiectului de către student și semnătura acestuia:	



UNIVERSITATEA DIN CRAIOVA  
Facultatea de Automatică, Calculatoare și Electronică

Departamentul de Calculatoare și Tehnologia Informației

## REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC

Numele și prenumele candidatului/-ei: Popescu Alexandru-Iulian  
Specializarea: Calculatoare cu predare în limba română  
Titlul proiectului: Platforma pentru intermedierea activităților de curierat folosind tehnologiile Cloud  
Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta):  
În facultate ☐  
În producție ☐  
În cercetare ☐  
Altă locație:

În urma analizei lucrării candidatului au fost constatate următoarele:

Nivelul documentării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Tipul proiectului		Cercetare <input type="checkbox"/>	Proiectare <input type="checkbox"/>	Realizare practică <input type="checkbox"/>	Altul [se detaliază]
Aparatul matematic utilizat		Simplu <input type="checkbox"/>	Mediu <input type="checkbox"/>	Complex <input type="checkbox"/>	Absent <input type="checkbox"/>
Utilitate		Contract de cercetare <input type="checkbox"/>	Cercetare internă <input type="checkbox"/>	Utilare <input type="checkbox"/>	Altul [se detaliază]
Redactarea lucrării		Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
Partea grafică, desene		Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
Realizarea practică	Contribuția autorului	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Mare <input type="checkbox"/>	Foarte mare <input type="checkbox"/>
	Complexitatea temei	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>
	Analiza cerințelor	Insuficient <input type="checkbox"/>	Satisfăcător <input type="checkbox"/>	Bine <input type="checkbox"/>	Foarte bine <input type="checkbox"/>
	Arhitectura	Simplă <input type="checkbox"/>	Medie <input type="checkbox"/>	Mare <input type="checkbox"/>	Complexă <input type="checkbox"/>
	Întocmirea specificațiilor funcționale	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>

	Implementarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Testarea	Insuficientă <input type="checkbox"/>	Satisfăcătoare <input type="checkbox"/>	Bună <input type="checkbox"/>	Foarte bună <input type="checkbox"/>
	Funcționarea	Da <input type="checkbox"/>	Parțială <input type="checkbox"/>	Nu <input type="checkbox"/>	
Rezultate experimentale		Experiment propriu <input type="checkbox"/>		Preluare din bibliografie <input type="checkbox"/>	
Bibliografie		Cărți	Reviste	Articole	Referințe web
Comentarii și observații					

În concluzie, se propune:

<b>ADMITEREA PROIECTULUI</b> <input type="checkbox"/>	<b>RESPINGEREA PROIECTULUI</b> <input type="checkbox"/>
--	--

Data,

Semnătura conducătorului științific,

## REZUMATUL PROIECTULUI

Platforma web poate fi folosită de către orice firmă de curierat, atât de către partea internă (gestionarea pachetelor, a șoferilor) cat și de către clienții acestei firme pentru crea cereri de ridicare a pachetelor. Aplicația poate fi utilizată pentru orice tip de comenzi care folosesc ansamblul general care este folosit de către aplicațiile ca Fancourier, Glovo sau Takeaway (șoferi, comenzi, Google maps, rute, dispecerat).

### **Etapele întregului proces de crearea și livrare de pachete**

Prima etapă a procesului este reprezentată de folosirea unui cont de Administrator pentru a crea conturi pentru șoferul angajat și pentru dispeceratul care va avea grijă de logistica din spate. După ce administratorul a creat conturile, dispeceratul poate adăuga mașinile care vor fi folosite pentru livrări în baza de date și poate vedea toate pachetele care au fost create de client. Acesta poate intra apoi pe pagina de rute pentru a crea o ruta care este făcută din mai multe comenzi și o poate asigna unui șofer împreună cu o mașină care este liberă. Șoferul poate începe ruta când este pregătit prin apăsarea butonului de start și o poate actualiza și finaliza când este gata.

**Termenii cheie:** curierat, administrare, CRUD, dispecerat, Cloud, maps



## MULȚUMIRI

Pentru realizarea acestei lucrări doresc să mulțumesc coordonatorului științific, Șef lucrări dr. ing. Sora Cătălin, pentru sfaturile și răbdarea de care a dat dovadă în călăuzirea mea pe tot drumul parcurs, cât și pentru documentația, suportul și feedback-ul acordat cu privire la dezvoltarea aplicației.

În aceeași măsură doresc să aduc mulțumiri întregii echipe profesionale cu care am interacționat în acești 4 ani, întrucât și-au adus aportul în structurarea cunoștințelor acumulate, în utilizarea acestora în practică, adică în întreaga mea formare profesională.

# CUPRINSUL

<b>1</b>	<b>INTRODUCERE .....</b>	<b>13</b>
1.1	SCOPUL.....	13
1.2	MOTIVAȚIA.....	13
1.3	DESCRIERE GENERALĂ .....	13
1.3.1	<i>Tehnologii și aplicații folosite.....</i>	<i>13</i>
1.3.2	<i>Tehnologii și framework-uri folosite .....</i>	<i>14</i>
1.4	FUNCȚIONALITĂȚILE APLICAȚIEI .....	17
1.4.1	<i>Funcționalitățile administratorului .....</i>	<i>17</i>
1.4.2	<i>Funcționalitățile șoferului .....</i>	<i>17</i>
1.4.3	<i>Funcționalitățile dispeceratului .....</i>	<i>17</i>
1.4.4	<i>Funcționalitățile clientului .....</i>	<i>17</i>
1.5	STRUCTURA LUCRĂRII.....	18
<b>2</b>	<b>ARHITECTURA SISTEMULUI.....</b>	<b>19</b>
2.1	AZURE CLOUD SERVICES.....	21
2.2	STRUCTURA BAZEI DE DATE.....	24
2.3	PROIECTAREA APLICAȚIEI.....	28
<b>3</b>	<b>CERINȚE PENTRU SISTEMUL IMPLEMENTAT .....</b>	<b>31</b>
3.1	FUNCȚIONALITĂȚILE ADMINISTRATORULUI.....	31
3.1.1	<i>Autentificare .....</i>	<i>31</i>
3.1.2	<i>Dashboard conturi .....</i>	<i>31</i>
3.1.3	<i>Adăugare cont .....</i>	<i>31</i>
3.1.4	<i>Editare cont.....</i>	<i>31</i>
3.1.5	<i>Logout.....</i>	<i>31</i>
3.2	FUNCȚIONALITĂȚILE DISPECERATULUI .....	31
3.2.1	<i>Autentificare .....</i>	<i>31</i>
3.2.2	<i>Dashboard vehicule.....</i>	<i>32</i>
3.2.3	<i>Dashboard comenzi .....</i>	<i>32</i>
3.2.4	<i>Dashboard șoferi.....</i>	<i>32</i>
3.2.5	<i>Dashboard rute .....</i>	<i>32</i>
3.3	FUNCȚIONALITĂȚILE ȘOFERULUI.....	32
3.3.1	<i>Operații pe ruta curenta .....</i>	<i>32</i>
3.4	FUNCȚIONALITĂȚILE CLIENTULUI FINAL.....	33
3.4.1	<i>Estimare cost transport.....</i>	<i>33</i>

3.4.2	<i>Editare profil</i> .....	33
3.4.3	<i>Adăugare locații</i> .....	33
3.4.4	<i>Editare locații</i> .....	33
3.4.5	<i>Ștergere locații</i> .....	33
3.4.6	<i>Creare comanda</i> .....	34
3.4.7	<i>Verificare stadiu comandă</i> .....	34
3.5	DETALII DE IMPLEMENTARE .....	35
3.5.1	<i>Rutarea folosind google maps</i> .....	35
3.5.2	<i>Utilizarea si configurarea serviciilor</i> .....	36
3.5.3	<i>Utilizarea partial views</i> .....	37
<b>4</b>	<b>MANUALUL UTILIZATORULUI .....</b>	<b>38</b>
4.1	PAGINA DE HOME (UTILIZATOR NEÎNREGISTRAT) .....	38
4.2	CREAREA UNUI CONT DE UTILIZATOR .....	40
4.3	PAGINA DE HOME (UTILIZATOR ÎNREGISTRAT) .....	41
4.4	PAGINA DE PROFIL .....	41
4.5	PAGINA DE COMENZI.....	43
4.6	AWB TRACKING .....	46
<b>5</b>	<b>CONCLUZII .....</b>	<b>47</b>
5.1	ÎMBUNĂTĂȚIRI SI DEZVOLTARE .....	47
<b>6</b>	<b>BIBLIOGRAFIE .....</b>	<b>48</b>
	<b>SITE-UL WEB AL PROIECTULUI.....</b>	<b>49</b>
<b>8</b>	<b>CD/DVD.....</b>	<b>50</b>

FIGURĂ 1 DIAGRAMA MVC.....	16
FIGURĂ 2 AZURE SERVICES .....	19
FIGURĂ 3 ARHITECTURA SISTEMULUI .....	20
FIGURĂ 4 AZURE SUBSCRIPTION .....	21
FIGURĂ 5 AZURE RESOURCE GROUP .....	21
FIGURĂ 6 AZURE APP SERVICE PLAN .....	22
FIGURĂ 7 AZURE SQL SERVER.....	22
FIGURĂ 8 AZURE DATABASE/QUERY EDITOR.....	22
FIGURĂ 9 AZURE DASHBOARD.....	23
FIGURĂ 10 AZURE APPLICATION INSIGHTS .....	23
FIGURĂ 11 DIAGRAMA DE CLASE (1) .....	25
FIGURĂ 12 DIAGRAMA DE CLASE (2) .....	26
FIGURĂ 13 DIAGRAMA DE CLASE (3) .....	26
FIGURĂ 14 MODELUL RELATIONAL AL BAZEI DE DATE .....	27
FIGURĂ 15 DIAGRAMA CAZURI DE UTILIZARE – CLIENT.....	28
FIGURĂ 16 DIAGRAMA CAZURI DE UTILIZARE – ADMINISTRATOR.....	29
FIGURĂ 17 DIAGRAMA CAZURI DE UTILIZARE – DISPECERAT .....	29
FIGURĂ 18 DIAGRAMA CAZURI DE UTILIZARE - ȘOFER .....	30
FIGURĂ 19 RUTARE GOOGLE MAPS 1.....	35
FIGURĂ 20 RUTARE GOOGLE MAPS 2.....	35
FIGURĂ 21 SERVICII .....	36
FIGURĂ 22 PARTIAL VIEW LOCATIONS.....	37
FIGURĂ 23 IMPLEMENTARE PARTIAL VIEW .....	37
FIGURĂ 24 DASHBOARD USER NELOGAT.....	38
FIGURĂ 25 COST ESTIMATE .....	39
FIGURĂ 26 COST ESTIMATE 2 .....	39
FIGURĂ 27 DETALII CALCULATOR ESTIMĂRI .....	40
FIGURĂ 28 REGISTER .....	40
FIGURĂ 29 DASBOARD UTILIZATOR INREGISTRAT.....	41
FIGURĂ 30 PAGINA DE PROFIL.....	41
FIGURĂ 31 PAGINA DE PROFIL 2 .....	42
FIGURĂ 32 ADD LOCATION .....	42
FIGURĂ 33 DASHBOARD ORDERS .....	43
FIGURĂ 34 NEW ORDER.....	44
FIGURĂ 35 ORDER MAPS.....	45
FIGURĂ 36 AWB TRACKING .....	46

# 1 INTRODUCERE

## 1.1 Scopul

Acest document are rolul de a prezenta aplicația web de curierat care poate fi folosită atât de utilizatorul final cât și de către firma de curierat pentru gestionarea pachetelor.

## 1.2 Motivația

Platforma de curierat a fost făcută cu scopul de a putea fi folosită și de către alte firme de start-up care vor să înceapă o firmă de curierat. Evoluția omenirii și a tehnologiei a făcut ca livrarea de orice tip să fie foarte căutată și folosită în orice domeniu datorită ușurinței de a obține orice este necesar din comoditatea locuinței, chiar și la distanțe mari.

## 1.3 Descriere generală

### 1.3.1 Tehnologii și aplicații folosite

Aplicația este o platformă web dezvoltată folosind framework-ul ASP .Net Core 3.1 cu modelul arhitectural MVC (Model-View-Controller) și cu Entity Framework Core 3.1.4 iar pentru autentificare și autorizare a fost integrat și folosit serviciul Identity versiunea 2.2.0. Aplicația a fost dezvoltată folosind IDE-ul Visual Studio 2022 și Microsoft SQL Server Management Studio 18 ca și mediu integrat care se ocupă cu managementul bazei de date. Partea de Cloud a fost făcută folosind Azure, portalul celor de Microsoft pentru management și Visual studio pentru publicare și modificare.



În plus, pentru a avea un istoric al dezvoltării și pentru siguranță, a fost folosit sistemul de control al versiunilor numit GIT, iar pentru găzduirea codului sursă a fost folosit serviciul Github.

Pe lângă acestea, jQuery a fost folosit pentru comunicarea cu server-ul folosind apeluri AJAX.

Pentru frontend, au fost folosite mai multe plugin-uri JavaScript care au scopul de a face aplicația mai intuitivă, deci de a crește gradul de utilizabilitate cât și câteva animații, fie reutilizate din diferite

pachete găsite pe internet, fie personificate pentru a avea o imagine de ansamblu a aplicației cât mai plăcută. Aceste plugin-uri sunt:

- DataTables, folosit pentru a structura tabelele într-o manieră mai ușor de înțeles
- Google Maps prin scripturi pentru a afișa în mod dinamic distanța și ruta pe hartă.

### 1.3.2 Tehnologii și framework-uri folosite

Aplicația este o platformă web care a fost implementată folosind framework-ul ASP .Net Core 3.1 cu serviciul de autentificare și autorizare Identity. De asemenea, Entity Framework Core este folosit ca și ORM (Object–Relational Mapper). ORM-ul ne permite să mapăm date din baza de date folosind paradigma programării orientate pe obiecte.

ASP .Net Core este un framework folosit pentru dezvoltarea de site-uri web independente de platformă (cross-platform) acesta folosind șablonul determinat de injectarea dependențelor (en. Dependency Injection sau DI). DI este o tehnică folosită pentru îndeplinirea principiului al 5-lea SOLID numit Dependency Inversion Principle între clase și dependențele lor. Așadar, repository-urile, dar și serviciile vor fi înregistrate în metoda ConfigureServices din clasa Startup.

În cazul serviciului Identity, acesta este un serviciu de autorizare și autentificare, fiind legat la o bază de date SQL Server pentru a stoca datele utilizatorilor cum ar fi parolele, email-urile, dar și alte date personale. Funcționalitățile de register, login, și editare a contului de utilizator au fost inițial create de serviciul Identity, acestea fiind apoi customizate pentru a se mula mai bine pe scopul proiectului. De asemenea, serviciul Identity oferă și funcționalitatea de autorizare astfel încât utilizatorii vor putea accesa numai paginile destinate rolului pe care îl au, deci clienții nu vor avea acces la funcționalitățile șoferilor, etc.

Dacă un utilizator care nu este autentificat încearcă să acceseze o pagină pentru care este nevoie să se autentifice, acesta va fi redirectat către pagina de autentificare, iar dacă utilizatorul este autentificat, dar încearcă să acceseze o pagină destinată altui tip de utilizator, va fi redirectat către o pagină de acces interzis („Access denied”).

De asemenea, Identity deține și opțiunea validării parolelor inserate. Aceste opțiuni fiind inserate în clasa de Startup.cs. Condițiile folosite sunt: parola trebuie să aibă minim 8 caractere, să conțină minim o cifră, o majusculă și un caracter alfanumeric.

Pe lângă tehnologiile descrise, au fost folosite și pachete NuGet, acestea fiind instalate prin intermediul managerului de pachete. Pachetele folosite sunt: GoogleMapsApi (folosit pentru

implementarea de funcționalității de hărți) și Razor runtime compilation pentru a putea face debug și hot reload fără a închide aplicația.

jQuery, care este o librărie de javascript ce poate procesa evenimente, manipula elemente html și a utiliza cereri de tip AJAX, a fost de asemenea folosit împreună cu anumite plugin-uri.

În plus, Bootstrap a fost folosit pentru view-uri pentru a face ca site-ul să fie responsive. Bootstrap este un toolkit open-source, care include un sistem grid responsive, diferite componente, dar și plugin-uri JavaScript.

Unul din plugin-urile folosite este DataTables, care a fost conceput pentru a modifica tabelele HTML cu scopul de a fi interactive. Prin utilizarea acestei librării, se vor stiliza tabelele folosind stilurile cuprinse de librărie. Acest plugin depinde doar de jQuery și necesită ca jQuery să fie integrat pentru a putea să fie folosit.

Plugin-ul DataTables oferă multiple funcționalități care fac aplicația mult mai ușor de utilizat și de înțeles cum ar fi:

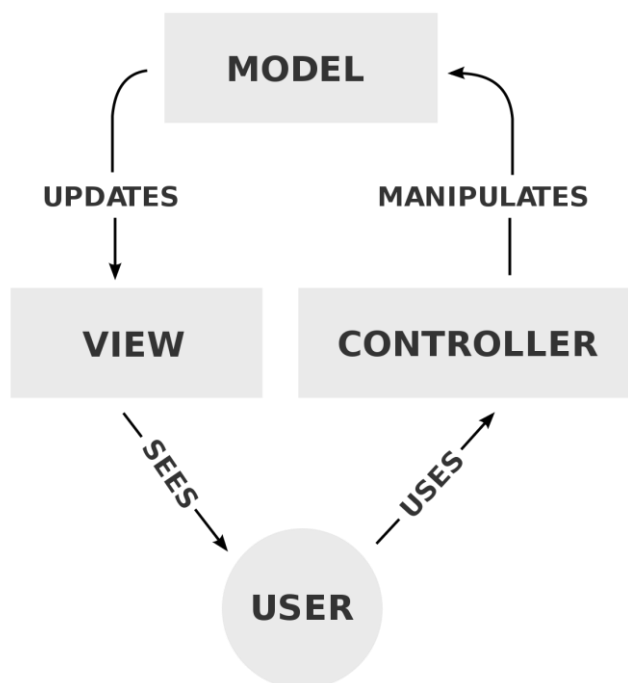
- posibilitatea de a sorta orice coloana aleasă de utilizator fie în ordine crescătoare sau descrescătoare
- împărțirea rezultatelor pe mai multe pagini
- posibilitatea de a alege câte rezultate sunt afișate în tabel
- filtrarea anumitor rezultate pe baza unui input dat de utilizator
- afișarea numărului total de rezultate

De asemenea, pe pagina de dashboard au fost create animații folosind fișiere de tip SVG care au fost personalizate și adaptate. Fișierele SVG au fost descărcate de pe un website care oferă ilustrații gratuite.

În ceea ce privește proiectarea aplicației și reducerea gradului de cuplare, aplicația a fost împărțită în trei straturi: stratul de prezentare, care include controllerele, view-urile, fișierele de stilizare (CSS), fișierele care conțin scripturile JavaScript, dar și componentele introduse de Identity în aplicație, stratul cu logica aplicației (Application Logic layer) și stratul folosit pentru comunicarea cu baza de date (Data Access layer). Pentru a crea straturile Application Logic și Data Access, au fost folosite câte două librării de clase (class libraries).

În plus, a fost folosit șablonul arhitectural MVC (Model-View-Controller), care separă logica de business de logica de prezentare. În timp ce modelele sunt folosite pentru a mapa tabelele din baza de date relațională, controllerele sunt folosite pentru a randa view-urile, pentru a face redirectările și pentru

a procesa cererile emise de utilizator. A treia componentă vizată de acest șablon este reprezentată de view-uri, acestea fiind reprezentate de componentele care sunt afișate în browser.



**Figură 1 Diagrama MVC**

În ceea ce privește baza de date, aceasta este una relațională. Tipul de bază de date relațională este cel mai răspândit tip de baze de date în care datele sunt memorate în tabele. „Pe lângă tabele, o bază de date relațională mai poate conține: indecși, proceduri stocate, declanșatori, utilizatori și grupuri de utilizatori, tipuri de date, mecanisme de securitate și de gestiune a tranzacțiilor etc.” (Wikipedia, 2022). În plus, pentru baza de date a fost folosit utilitarul Microsoft SQL Server Management Studio 18, care reprezintă un „mediu integrat folosit pentru managementul oricărei infrastructuri SQL [...] SSMS oferă funcții folosite pentru configurarea, monitorizarea și administrarea instanțelor de SQL Server și a bazelor de date”. [\[8\]](#)

Toata aceasta aplicație a fost publicata online folosind tehnologia Cloud, mai precis Microsoft Azure. Pentru acest lucru a fost nevoie de mai multe resurse, descrise în detaliu în [capitolul 1.5](#).



## **1.4 Funcționalitățile aplicației**

### **1.4.1 Funcționalitățile administratorului**

Primul administrator este creat folosind un seeding de date de către cod. Acest prim administrator poate face operații de tip CRUD (Create, Read, Update, Delete), pentru restul conturilor, în acest fel, primul administrator poate adăuga și alte conturi de admin, conturi de șofer sau de dispecerat.

### **1.4.2 Funcționalitățile șoferului**

Șoferul este utilizatorul care acces la ruta asignată acestuia de către dispecerat. După ce ruta este începută, șoferul își poate începe munca. Odată ce o comandă a fost dusă la bun sfârșit, sau au fost probleme cu ea, șoferul poate apăsa fie pe butonul de „Complete”, fie pe cel de „Cancel” pentru a actualiza statusul comenzii. Odată ce toate comenzile dintr-o ruta au fost aduse la un status de final, șoferul poate opri ruta.

### **1.4.3 Funcționalitățile dispeceratului**

Dispeceratul are acces la toate comenzile create de utilizatori. Aceștia pot crea rute, le poate lega la o mașină liberă și poate adăuga comenzi la această ruta.

### **1.4.4 Funcționalitățile clientului**

Clientul își poate edita profilul și locațiile de ridicare a coletelor din propria sa pagină de profil. De asemenea, acesta poate crea comenzi și le poate urmări folosind AWB-ul generat

## 1.5 Structura lucrării

Primul capitol numit „[INTRODUCERE](#)” conține detalii generale despre aplicație, mai exact, scopul documentului, motivația, funcționalitățile implementate, tehnologiile folosite.

Al doilea capitol numit „[Arhitectura sistemului](#)” prezintă arhitectura sistemului implementat, baza de date și tehnologiile Cloud folosite.

Capitolul 3: „[Cerințe pentru sistemul implementat](#)” include toate funcționalitățile aplicației.

Capitolul 4: „[Manualul utilizatorului](#)” include instrucțiuni de utilizare a sistemului.

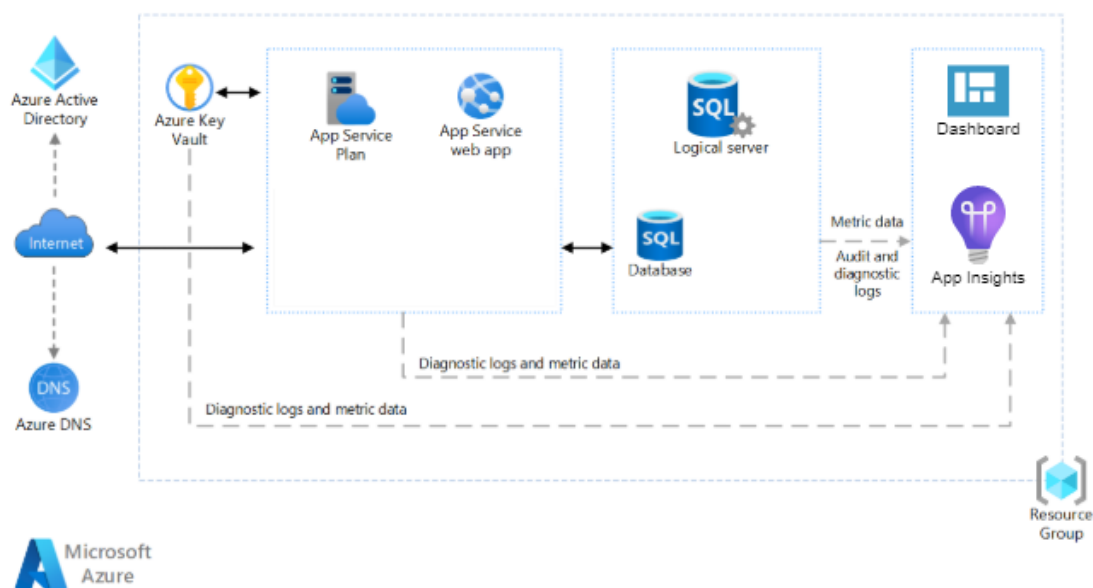
Capitolul 5: „[Concluzii](#)” este capitolul care cuprinde concluziile dobândite în urma implementării acestui proiect, detalii despre limitările sistemului.

Capitolul 6: „[Bibliografie](#)” include referințele din literatura de specialitate, dar și referințele web.

Ultimul capitol, „[CD/DVD](#)” include DVD-ul care conține codul sursă.

## 2 ARHITECTURA SISTEMULUI

Sistemul nu este unul monolitic, ci este unul format din mai multe straturi care comunică între ele cu scopul de a reduce gradul de cuplare al aplicației, fiecare strat putând fi reutilizat. Cele trei straturi principale folosite sunt: *stratul de prezentare (Presentation layer)*, *stratul care cuprinde logica aplicației (Application logic layer)* și *stratul Data access (Data Access layer)*.



Figură 2 Azure services

Toata aplicația este găzduită în cadrul Azure. Aceasta este înglobată într-un grup de resurse, unde se afla atât partea de secrets și aplicația în sine (app service), baza de date dar și partea de diagnoza, cum ar fi Dashboard sau Application Insights.

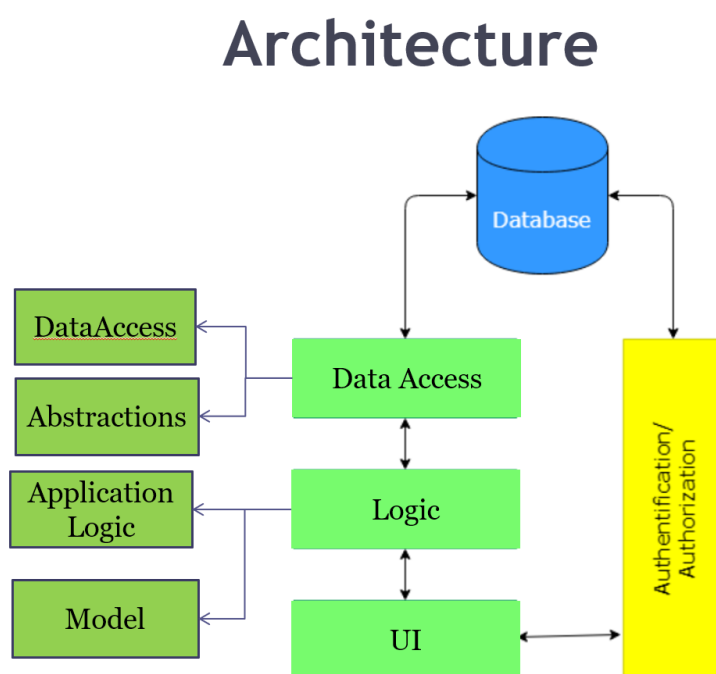
Stratul de prezentare este stratul folosit direct de utilizator și cuprinde: controllerele, view-urile, script-urile JavaScript, fișierele de stilizare CSS, imagini, documentele încărcate de administrator, fișierele de configurare și funcționalitățile generate folosind serviciul de autentificare și autorizare Identity.

Stratul care cuprinde logica aplicației este cel care include serviciile folosite în cadrul aplicației. Pentru fiecare serviciu a fost folosită o interfață pentru a putea reduce gradul de cuplare al aplicației deoarece ultimul principiu SOLID (Dependency Inversion Principle) care se referă la faptul că abstractizările ar trebui folosite în loc de clase concrete este astfel respectat. Acest strat cuprinde: interfețele care sunt implementate de repository-uri, interfețele implementate de servicii, serviciile

(clasele concrete) care conțin logica aplicației, modelele care sunt folosite la maparea tabelelor din baza de date, viewmodelele care sunt clase asemănătoare modelelor, dar care nu au un corespondent în baza de date. Viewmodelele sunt modele care conțin multiple proprietăți care nu sunt mapate în baza de date.

Stratul Data Access este stratul folosit pentru interacțiunea cu baza de date și reprezintă o abstractizare a sa. Acest strat cuprinde clasele concrete care reprezintă repository-urile și directorul care include migrările folosite pentru generarea bazei de date (folosind strategia CodeFirst).

De asemenea, a fost folosit Repository Pattern, care este un șablon de proiectare care constă în abstractizarea accesului datelor stocate în baza de date prin intermediul unor metode. Folosind acest șablon, va scădea gradul de cuplare al aplicației. Acest șablon constă în crearea unei clase de bază generice, în cadrul aplicației fiind numită EFBaseRepository, care va conține metode generice folosite pentru operații de tip CRUD (Create, Read, Update, Delete). În EFBaseRepository sunt definite metodele următoare: GetAll (tipul de return este IEnumerable), GetById (cu tip de return DataEntity), Add (folosită pentru inserarea de noi date/records în baza de date), Update (pentru a actualiza datele existente în baza de date), Remove (pentru a șterge date). Pe de altă parte, pentru fiecare tabel din baza de date a fost creată o clasă de repository care extinde EFBaseRepository. În plus, pentru a reduce gradul de cuplare al aplicației, prin respectarea principiilor 2 și 5 SOLID, fiecare repository va implementa o interfață, iar repository-urile, dar și serviciile vor fi înregistrate ca și servicii Scoped în clasa Startup pentru a putea fi folosite prin Dependency Injection. [\[6\]](#)

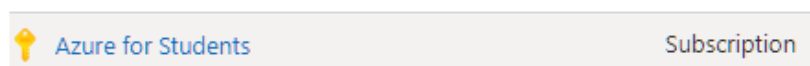


**Figură 3 Arhitectura sistemului**

## 2.1 Azure cloud services

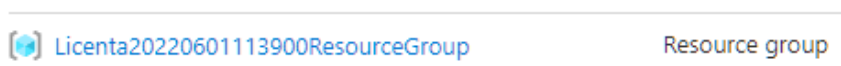
Platforma Azure cloud este un serviciu de cloud computing operat de Microsoft pentru gestionarea aplicațiilor prin centre de date gestionate de Microsoft. Oferă software ca serviciu (SaaS), platformă ca serviciu (PaaS) și infrastructură ca serviciu (IaaS) și acceptă multe limbaje de programare, instrumente și cadre diferite, inclusiv software și sisteme specifice Microsoft și terțe părți. [\[1\]](#)

- Cont, a fost folosit cel de „@robotics.ucv” pentru a avea beneficiile unui „Azure for students”. Acesta începe cu un credit Azure de 100\$, nu este nevoie de un card de credit iar majoritatea serviciilor sunt gratis.
- Subscription: Un abonament Azure este un container de bază care cuprinde un grup de resurse comerciale sau tehnice conexe. Grupul de resurse este utilizat și facturat împreună. Un abonament Azure acționează, de asemenea, ca o graniță administrativă, ceea ce înseamnă că permite administratorilor abonamentului să acceseze toate resursele din cadrul abonamentului și să delege accesul prin mecanisme de control al accesului bazate pe roluri.



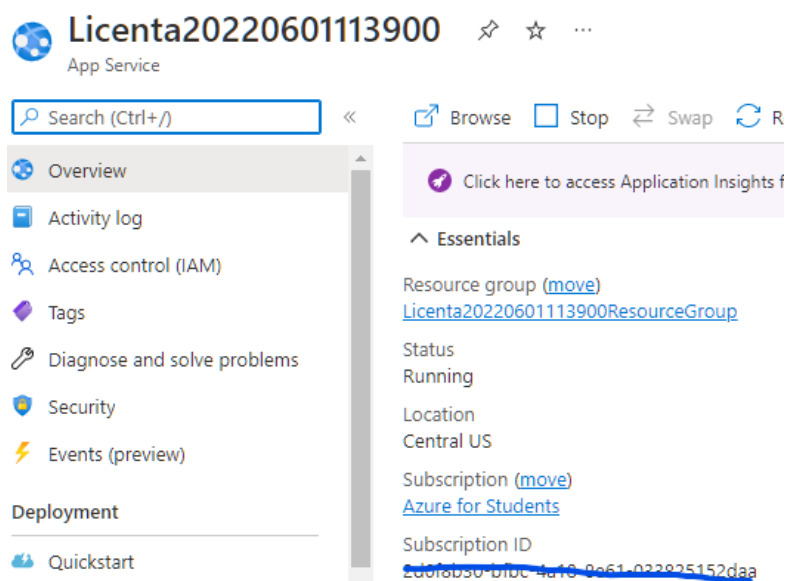
**Figură 4 Azure subscription**

- Resource group: include toate resursele folosite și descrise mai jos, un grup de resurse este asignat unui abonament (subscription). În general, se adaugă resursele care au același ciclu de viață la același grup de resurse, astfel încât să se poată implementa, actualiza și șterge cu ușurință ca un grup.



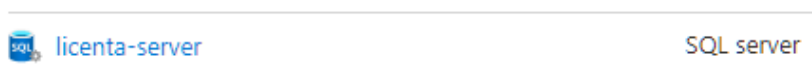
**Figură 5 Azure resource group**

- App service plan: definește un set de resurse care sunt asigurate către aplicația respectivă, care se plătesc, pentru ca aceasta să ruleze pe cloud.



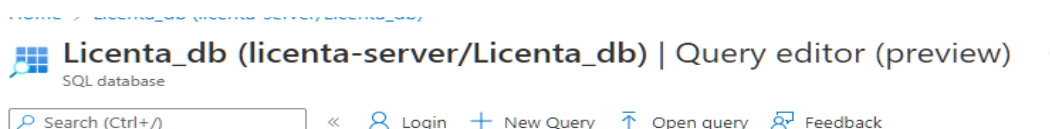
**Figură 6 Azure app service plan**

- Sql server: server-ul care susține bazele de date folosite de aplicație, acestea pot exista fie separat, fie folosind un elastic pool pentru a împărți un anumit set de resurse.



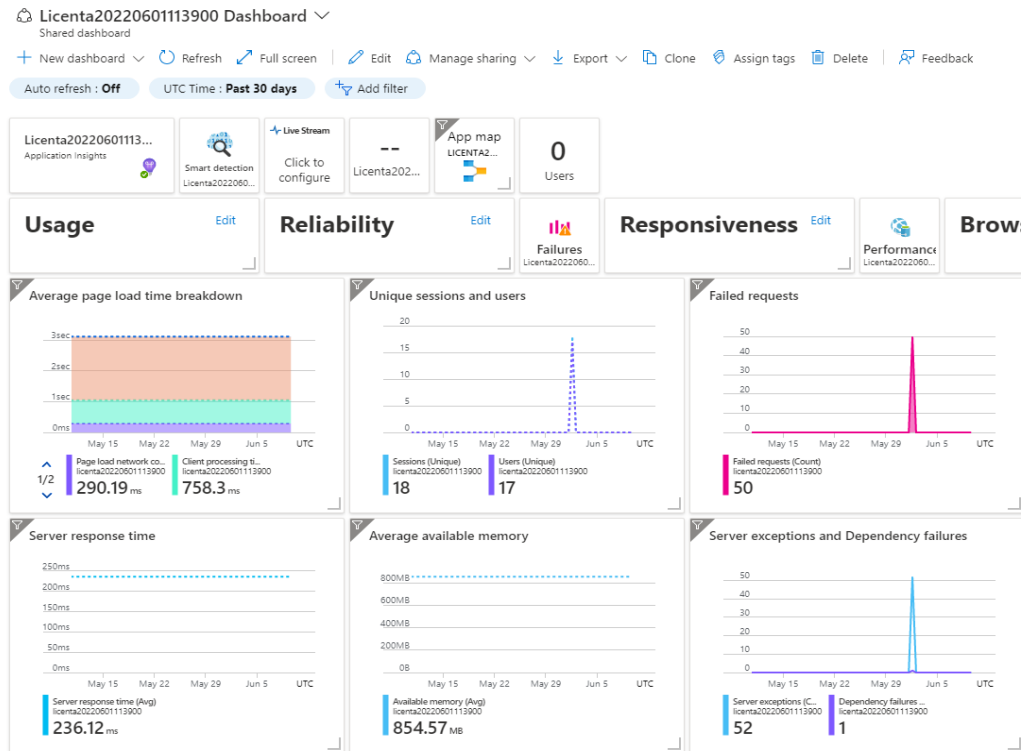
**Figură 7 Azure sql server**

- Sql database: baza de date folosita. Aceasta are deja un query editor implementat in Azure, fără a fi nevoie de SQL Management Studio.



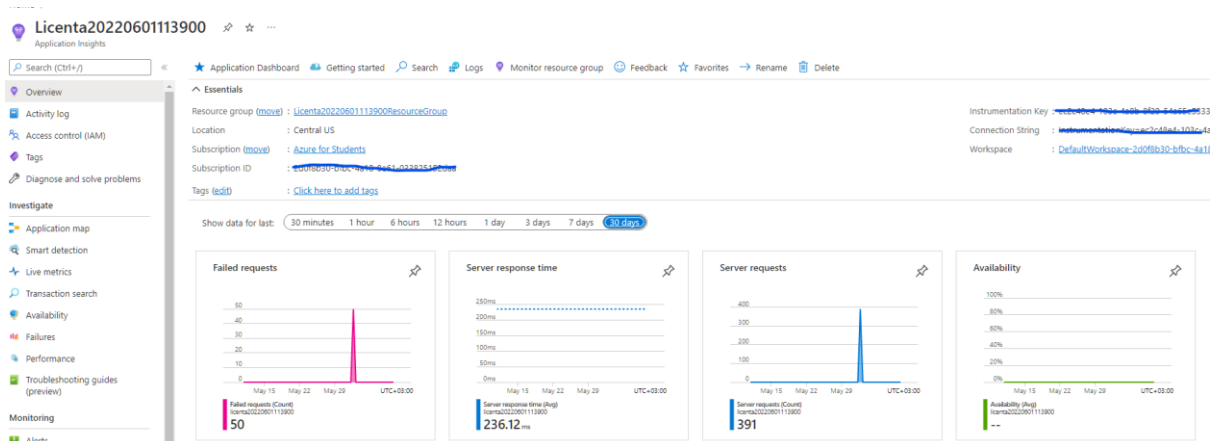
**Figură 8 Azure database/query editor**

- Dashboard: poate fi folosit de către dezvoltari/product owneri pentru a vedea diferite informații legate de utilizarea resurselor azure, de exemplu, numărul de requesturi către aplicație, numărul de mesaje transmise pe un service bus, numărul de timeouts pentru unele requesturi etc.



Figură 9 Azure dashboard

- Application Insights care este o caracteristică de monitorizare Azure ce oferă un management extensibil al performanței și monitorizării aplicației web live. Insights poate de exemplu să detecteze automat anomalii de performanță întâmpinate, ajuta la diagnosticarea problemelor găsite și poate vedea ce apeluri sunt făcute de către utilizatori



Figură 10 Azure application insights

## 2.2 Structura bazei de date

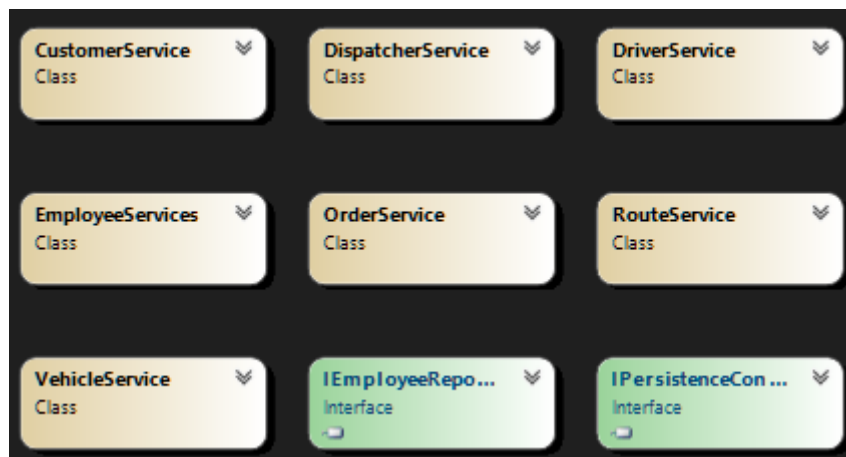
Baza de date este una relațională și a fost creată folosind strategia CodeFirst. Mai exact, înainte să fie creată baza de date, au fost create modelele, iar fiecare model reprezintă o entitate a tabelor din baza de date în aplicație. După ce modelele au fost create, au fost adăugate în clasa de context, iar cu ajutorul Entity Framework Core, a fost creată o migrare. Migrările sunt localizate în stratul de Data Access în directorul Migrations. Migrările reprezintă clase care conțin două metode: Up și Down. Metoda Up este folosită pentru a aplica schimbările pe baza de date, iar metoda Down este folosită pentru a reseta starea bazei de date, astfel încât după resetare, baza de date se va afla într-o stare stabilă, mai exact starea stabilă inițială. Migrările oferă avantajul că se poate face revert la starea precedentă dacă se constată că migrarea nu ar fi trebuit rulată. Un alt avantaj este că în cazul în care migrarea nu se poate executa cu succes, se aplică mecanismul de roll back, deci baza de date nu va fi alterată, ci se va afla în starea inițială. Migrările sunt construite pe baza migrărilor precedente și prin compararea modelelor folosite în aplicație cu starea curentă a bazei de date. Astfel se poate menține un istoric al schimbărilor și se poate reconstrui baza de date, această acțiune presupunând rularea migrărilor într-o manieră secvențială și cronologică.

Mai jos au fost adăugate diagramele de clase care sunt împărțite în funcție de principalele funcționalități. Deoarece fiecare model avea nevoie de un Id, sub forma de Guid, pentru a putea fi găsit în baza de date, toate modele moștenesc o clasa comună și anume DataEntity:





Figură 11 Diagrama de clase (1)

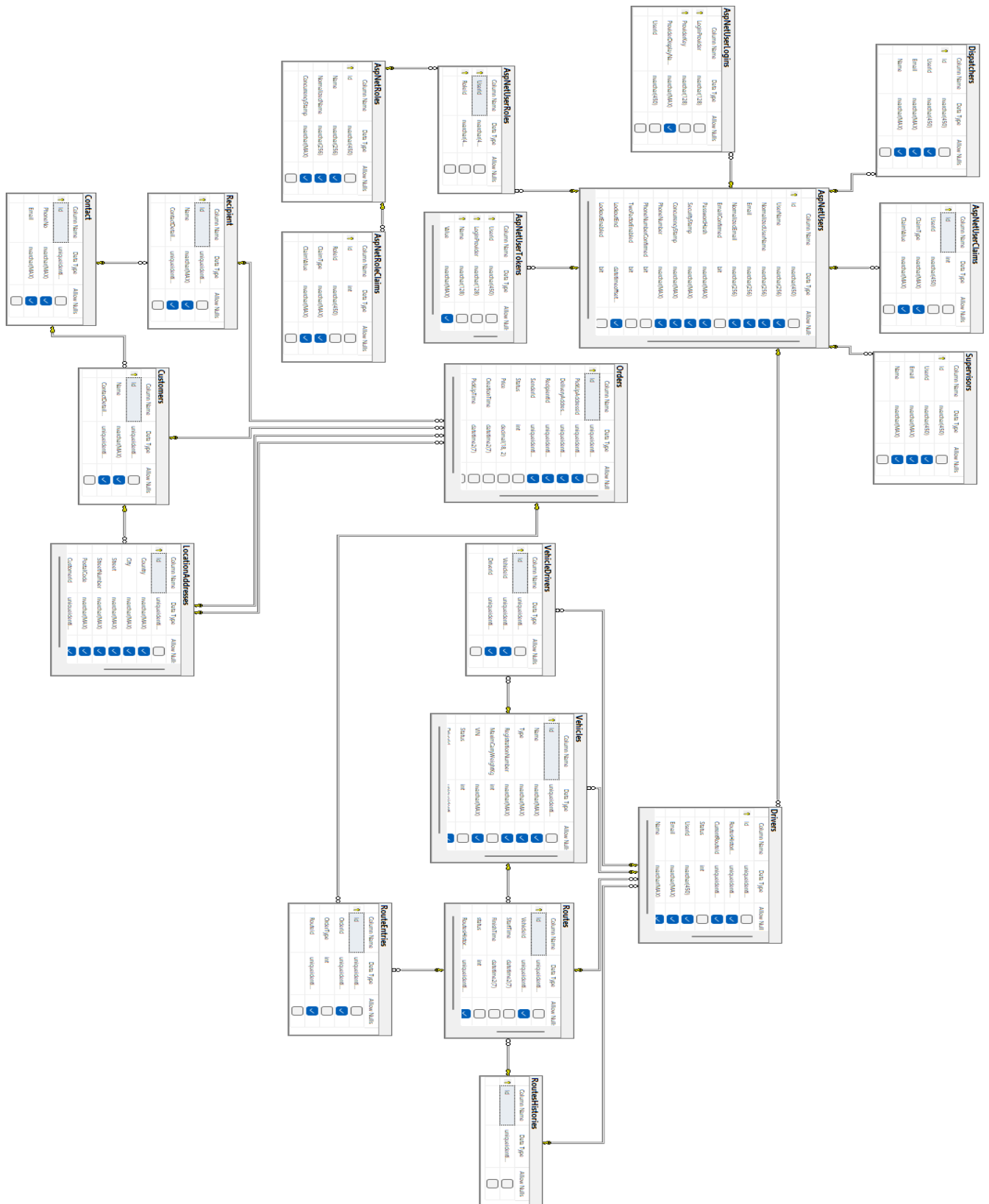


Figură 12 Diagrama de clase (2)



Figură 13 Diagrama de clase (3)

În figurile următoare se pot observa legăturile dintre tabelele bazei de date:

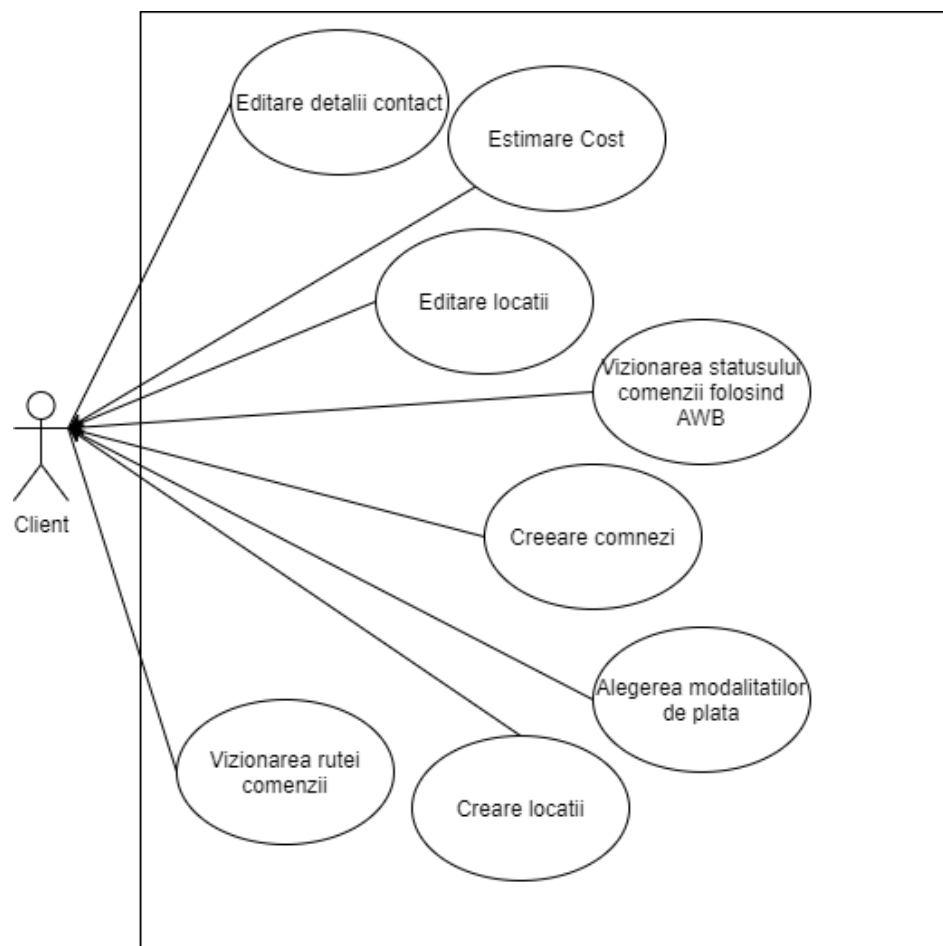


**Figură 14 Modelul relational al bazei de date**

## 2.3 Proiectarea aplicației

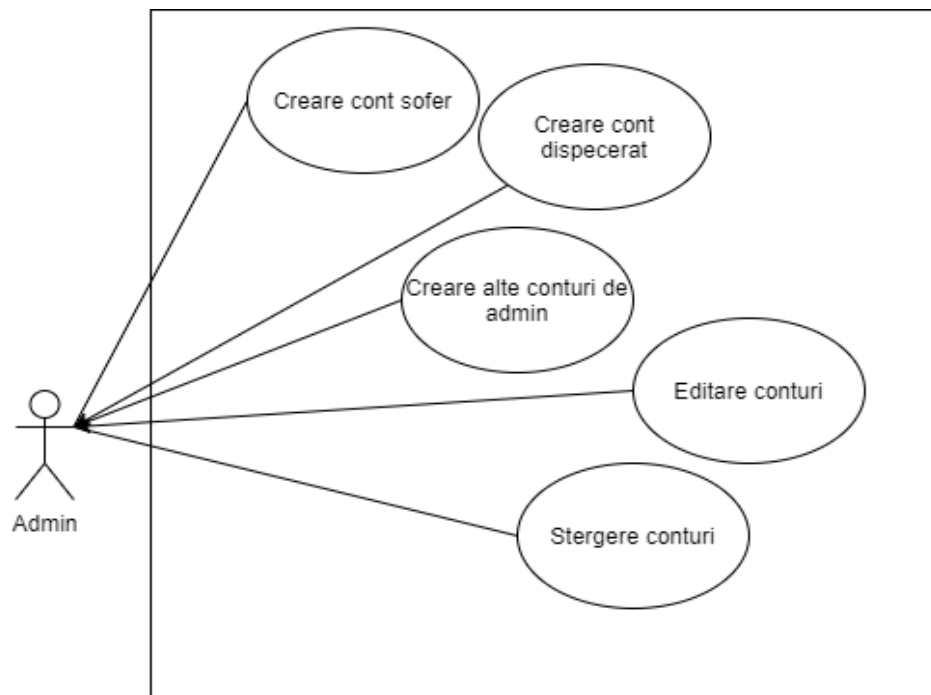
Aplicația de curierat a fost concepută cu scopul de a fi folosită de toate persoanele care ar putea exista în cerul acestei aplicației. Cu acest scop, aplicația este creată cu mai multe roluri în minte. În plus, principiul trei SOLID numit Principiul Substituției Liskov, a fost respectat pentru că orice clasă de bază poate fi înlocuită de o clasă derivată a sa fără să fie nevoie de modificarea codului. Pentru a respecta al patrulea principiu, Interface Segregation Principle, a fost creat câte un repository și câte un serviciu pentru fiecare entitate implicată în sistem, conținând doar metodele care sunt strict necesare pentru a folosi acel tip de entitate. Mai jos sunt prezentate use-case-urile acestor utilizatori, acestea fiind explicate în detaliu în cadrul cerințelor:

- Client



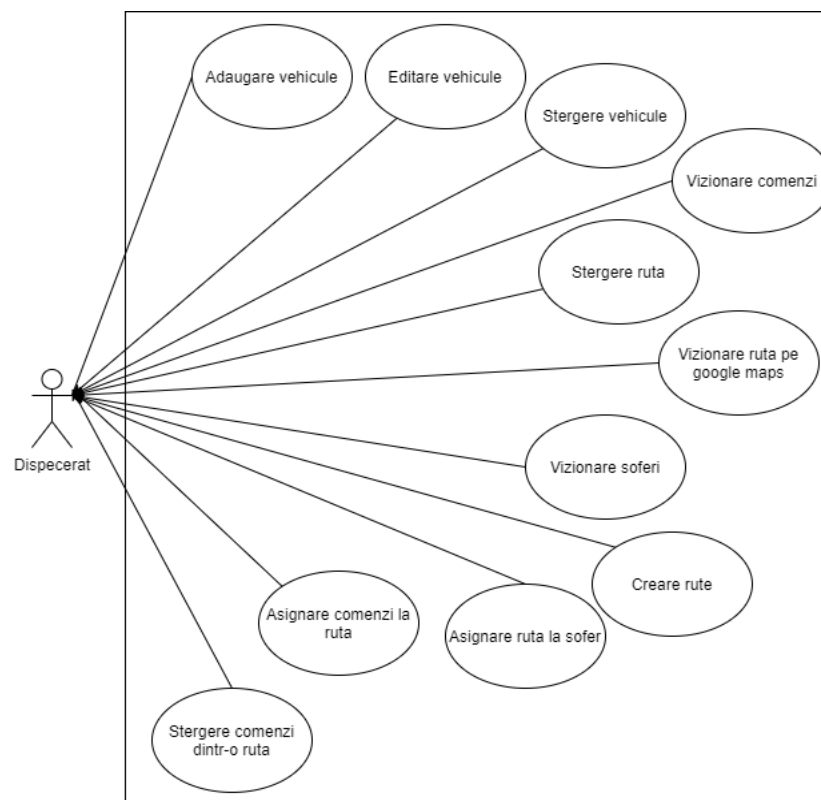
Figură 15 Diagrama cazuri de utilizare – Client

- Admin



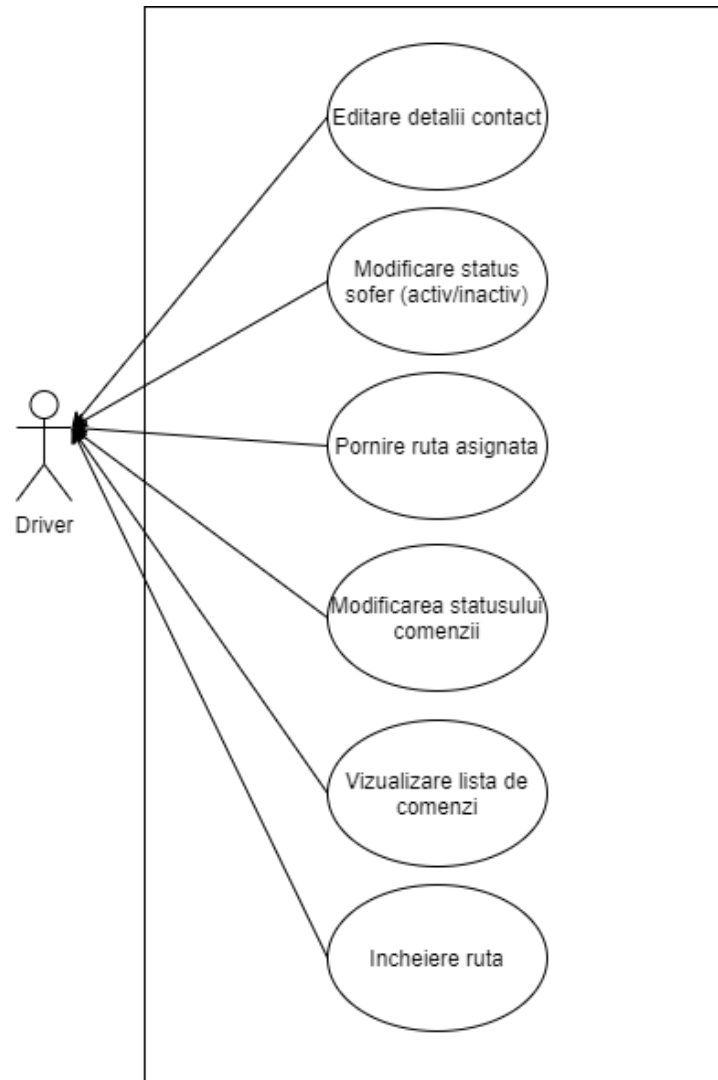
**Figură 16 Diagrama cazuri de utilizare – Administrator**

- Dispecerat



**Figură 17 Diagrama cazuri de utilizare – Dispecerat**

- Șofer



**Figură 18 Diagrama cazuri de utilizare - Șofer**

## **3 CERINȚE PENTRU SISTEMUL IMPLEMENTAT**

### **3.1 Funcționalitățile administratorului**

#### **3.1.1 Autentificare**

Contul de administrator este destinat managerilor din firmă. Acesta nu se poate crea ca un cont normal de client, ci este creat automat la start-up. Administratorul se poate autentifica folosind ca și user: admin@gmail.com și parola Admin1!. După ce s-a autentificat, administratorul va fi redirecționat către pagina de dashboard.

#### **3.1.2 Dashboard conturi**

Apăsând pe butonul de „Accounts”, administratorul va fi întâmpinat de un tabel populat cu toți utilizatorii, pe care acesta poate face operații CRUD.

#### **3.1.3 Adăugare cont**

Administratorul este singurul rol care poate adăuga alte conturi folosite în interiorul firmei (alt admin, dispecerat, șofer).

#### **3.1.4 Editare cont**

Administratorul poate să modifice conturile deja existente, în caz că este nevoie de un schimb de username, parolă sau rol.

#### **3.1.5 Logout**

Utilizatorii autentificați au posibilitatea de a se deconecta prin folosirea butonului de logout prezent în bara de navigație.

### **3.2 Funcționalitățile dispeceratului**

#### **3.2.1 Autentificare**

După crearea contului de utilizator, dispecerul se poate autentifica folosind email-ul și parola setate la pasul de înscriere. Dacă autentificarea nu a avut succes (date invalide), utilizatorul va primi mesaje de eroare. În cazul în care autentificarea a avut loc cu succes, dispecerul va fi redirecționat către pagina de dashboard.

### **3.2.2 Dashboard vehicule**

Dispecerul are acces la un tabel care va include toate vehiculele adăugate în baza de date. Pe lângă detaliile generale despre un vehicul ( nume, tip, VIN), dispecerul poate vedea și statusul vehiculului ( Free daca este liber, Busy daca este deja atașat unei rute).

### **3.2.3 Dashboard comenzi**

În tabelul de comenzi, dispecerul poate vedea adresa de ridicare, adresa de livrare, recipientul, user-ul care a făcut comanda (în cazul în care este necesar să îl contacteze pentru informații adiționale), statusul comenzii (Created, Assigned, PickedUp, Delivering, Delivered, Canceled), prețul și un buton pentru deschiderea unui pop-up pentru Google maps unde dispecerul poate vedea cea mai rapida rută creată de Google maps.

### **3.2.4 Dashboard șoferi**

În tabelul de șofer, dispecerul poate vedea toți șoferii înregistrați (id-ul lor, email-ul și numele), ruta curentă cu câte comenzi sunt asignate pe acea rută, statusul comenzii și daca ruta a fost asignată sau nu unui șofer.

### **3.2.5 Dashboard rute**

Această pagină este folosită pentru diferite acțiuni ce se pot efectua asupra unei rute. Dispecerul poate crea o rută nouă, fiind obligat să îi adauge un vehicul care are statusul de „Free”. Ruta este creată fără nicio comandă, acestea trebuind să fie adăugate de pe butonul de „Add Order”. Se poate vedea, de asemenea, ruta pe Google maps, aceasta unind rutele de la fiecare comandă și încercând să facă cel mai scurt drum. Dacă un pachet este adăugat din greșeală, user-ul poate elimina doar un pachet sau toată ruta. De asemenea, se poate schimba vehiculul cu un altul liber dacă este necesar.

## **3.3 Funcționalitățile șoferului**

### **3.3.1 Operații pe ruta curentă**

În această pagină, șoferul poate vedea ce ruta a fost asignată acestuia de către unul dintre dispeceri. După ce șoferul are o rută, acesta poate vedea pachetele care sunt incluse în acea rută și de asemenea poate vedea, folosind Google Maps, cel mai scurt drum pentru a putea ridica și lăsa toate comenzile incluse în ruta respectiva. În acest moment, pachetele au tag-ul de „Assigned”. Când șoferul este gata de plecare, acesta poate apăsa pe butonul de „Start Route”, tag-ul pachetelor devenind „Delivering”.

Odată ce ruta este pornită, fiecare pachet are două opțiuni: Delivered (pachetul a ajuns cu succes la destinație) și Cancel ( au fost diferite probleme cu pachetul). Îndată ce toate pachetele din ruta



respectivă ajung la una din cele două stări de mai sus, ruta se poate încheia apăsând pe butonul „End Route”, șoferul devenind iar liber și pregătit de o nouă rută.

### **3.4 Funcționalitățile clientului final**

#### **3.4.1 Estimare cost transport**

Clientul are acces la un calculator de costuri estimativ pentru transport. Acesta trebuia să introducă valoarea comenzii, ce tip de transport este nevoie să fie chemat pentru comanda respectivă în funcție de dimensiuni și kilometraj (bicicleta, mașina, camion) și un număr de kilometri.

#### **3.4.2 Editare profil**

Clientul poate folosi pagina de profil pentru a edita detalii despre acesta cum ar fi numele, numărul de telefon, email-ul.

#### **3.4.3 Adăugare locații**

Din pagina de editare a profilului, utilizatorul își poate adăuga și locațiile de ridicare a coletelor. O locație nouă are nevoie obligatoriu de următoarele date (tară, oraș, strada, nr.stradă,cod poștal) și,opțional,,de un „Tag”, acesta fiind folosit în loc de toata adresa pentru a fi mai ușor de găsit (de ex. Home).

#### **3.4.4 Editare locații**

Utilizatorul poate edita toate datele unei locații deja create anterior.

#### **3.4.5 Ștergere locații**

Utilizatorul își poate șterge propriile locații dacă nu mai sunt necesare.

### 3.4.6 Creare comanda

Pe pagina de dashboard a comenzilor, clientul poate apăsa pe butonul de „New Order” pentru a deschide modalul cu informațiile necesare pentru a crea o comandă nouă:

- PickUp Address: este un dropdown cu toate locațiile care le are clientul respectiv (adresa sau, dacă există, tag-ul)
- Country
- City
- Street
- Street Number
- Postal Code
- Recipient Name
- Recipient Email
- Recipient Phone Number
- Price

### 3.4.7 Verificare stadiu comandă

Stadiul unei comenzi poate fi văzut din două locuri: Dacă clientul este deja logat pe site, acesta poate intra pe dashboard-ul de comenzi și poate analiza statusul comenzii în tabel. În schimb, dacă acesta nu este logat și doar vrea să verifice statusul unei comenzi, acesta o poate face din dashboard-ul site-ului principal, fără a se loga, prin folosirea awb-ului generat după crearea comenzii. Clientul poate introduce AWB-ul în inputul specificat pentru a afla statusul, în timp real, al comenzii.

## 3.5 Detalii de implementare

### 3.5.1 Rutarea folosind google maps

Pentru a putea folosi opțiunea de rute implementata de google, este nevoie sa cream anumite puncte de interes pe harta (waypoints), care sunt punctele de ridicare si de livrare ale comenzilor.

```
}  
var ordersNumber = addresses.length;  
var wayPoints = [];  
var startAddress = addresses[0];  
var finishAddress = addresses[ordersNumber - 1];  
addresses.pop();  
addresses.shift();  
for (var i = 0; i < addresses.length; i++) {  
    var address = {  
        location: addresses[i].StreetNumber + " " + addresses[i].Street + "," + addresses[i].City,  
        stopover: true  
    }  
    wayPoints.push(address);  
}  
}
```

Figură 19 Rutare google maps 1

Înainte sa inițializăm harta, iterăm prin toate adresele care sunt găsite asupra rutei respective si le adăugăm sub forma de un string concatenat la un vector de puncte folosind funcția „push”.

În funcția de inițializare, avem nevoie de serviciul de direcție de la google dar si de serviciul de afișare a direcției. După ce centram harta, putem adauga o ruta noua, cu originea ca adresa de start, destinația ca adresa de livrare si lista de waypoints creata mai sus, pentru a putea alege cel mai scurt drum automat. Alegem de asemenea ca mod de transport „Driving” pentru ca toți șoferii o sa conducă, acest mod verificând si traficul, sensurile unice etc.

```
function initMap() {  
    var directionsService = new google.maps.DirectionsService;  
    var directionsDisplay = new google.maps.DirectionsRenderer;  
    var map = new google.maps.Map(document.getElementById('map'), {  
        zoom: 15,  
        center: { lat: 41.85, lng: -87.65 }  
    });  
    directionsDisplay.setMap(map);  
    directionsService.route({  
        origin: startAddress.Street + " " + startAddress.StreetNumber + "," + startAddress.City,  
        destination: finishAddress.Street + " " + finishAddress.StreetNumber + "," + finishAddress.City,  
        waypoints: wayPoints,  
        travelMode: 'DRIVING'  
    }, function (response, status) {  
        if (status === 'OK') {  
            directionsDisplay.setDirections(response);  
        } else {  
            window.alert('Directions request failed due to ' + status);  
        }  
    });  
}
```

Figură 20 Rutare google maps 2

### 3.5.2 Utilizarea si configurarea serviciilor

ASP.NET Core utilizează injectia de dependențe ca o caracteristică fundamentală pentru a gestiona dependențele în întregul cadru. Pentru ca cadrul de injectie a dependențelor să știe cum să rezolve dependențele, aceste dependențe sau "servicii" trebuie să fie configurate mai întâi. Acesta este motivul pentru care există metoda `ConfigureServices`: Aici, se pot adăuga serviciile pe care dorim să le activăm, chiar si cele personalizate, create de noi.

```
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
        options/*.UseLazyLoadingProxies()*/.UseSqlServer(
            Configuration.GetConnectionString("LicentaContextConnection")));

    services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
        .AddRoles<IdentityRole>()
        .AddEntityFrameworkStores<ApplicationDbContext>();
    services.AddControllersWithViews().AddNewtonsoftJson();

    services.AddMvc();
    services.AddScoped<IPersistenceContext, EFPersistenceContext>();
    services.AddScoped<ICustomerRepository, EFCustomerRepository>();
    services.AddScoped<IOrderRepository, EFOOrderRepository>();
    services.AddScoped<CustomerService>();
    services.AddScoped<OrderService>();
    services.AddScoped<EmployeeServices>();
    services.AddScoped<DriverService>();
    services.AddScoped<VehicleService>();
    services.AddScoped<DispatcherService>();
    services.AddScoped<RouteService>();

    services.AddRazorPages().AddRazorRuntimeCompilation();
}
```

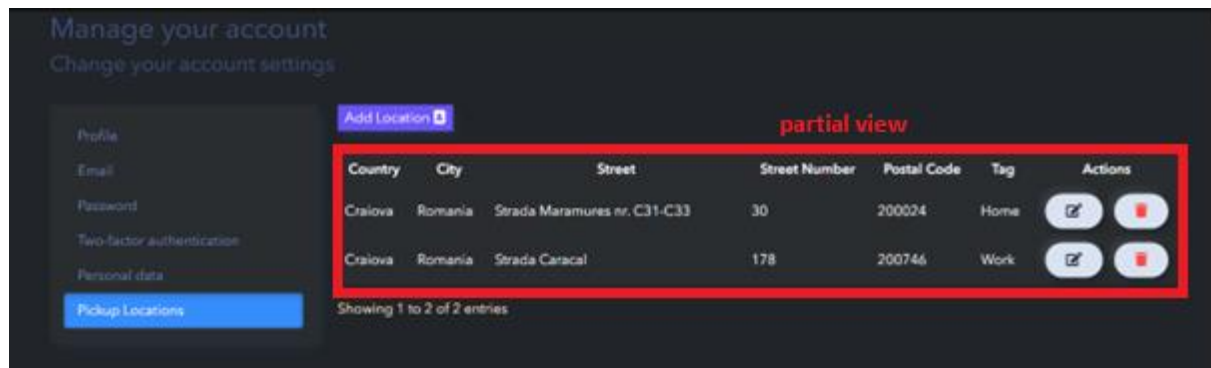
Figură 21 Servicii

Pe lângă serviciile create de noi (cele configurate folosind funcția “`AddScoped`”, avem si alte servicii “helper”, de exemplu:

- `AddDbContext` – leagă baza de date de către contextual din aplicație. Aceasta baza de date este create folosind implementarea “Code first” si este create folosind migrări.
- `AddDefaultIdentity` – serviciu folosit pentru Identity. Adaugă la aceeași baza de date de mai sus tabelele si configurarea necesara pentru a folosi useri, roluri etc.
- `AddRazorPages` – aceasta este partea de front-end, împreuna cu funcția “`AddRazorRuntimeCompilation()`”, putem face modificări in toate fișierele .cshtml fără sa resetam aplicația, acestea vor fi validate după un hard refresh al paginii de browser (ctrl + F5)

### 3.5.3 Utilizarea partial views

Aplicatia de curierat se bazeaza foarte mult pe afisarea anumitor tabele cu diverse informatii, de la locatii, la comenzi. Aceste tabele pot avea diferite actiuni care se pot face pe ele, de exemplu o actualizare a anumitor date din acesta. Pentru a nu reincarca intreaba pagina, aplicatia foloseste Partial Views. Astfel, daca un set de date este actualizat in tabelul din pagina respectiva, doar acea parte din pagina cu noile date va fi actualizata, acest lucru reducand foarte mult timpul de incarcare al paginilor.



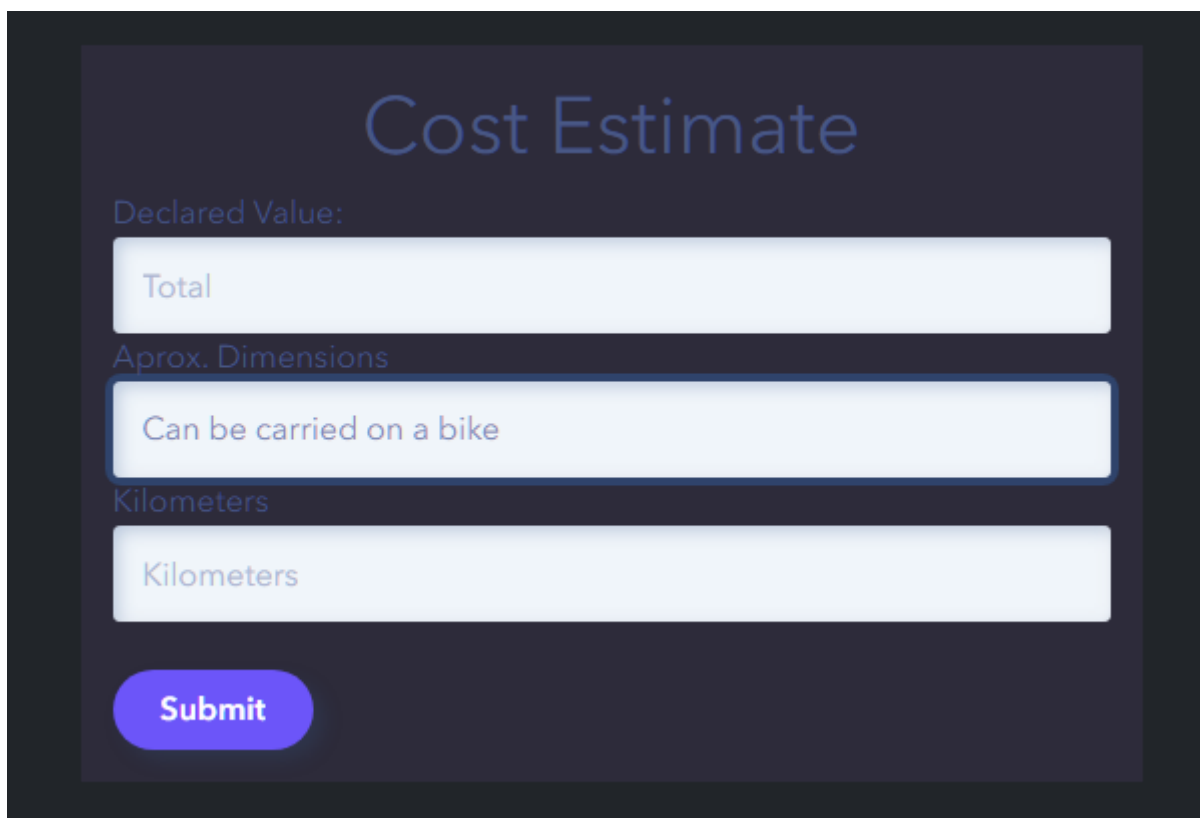
Figură 22 Partial view locations

```
0 references
public IActionResult AddressTable()
{
    var user = User.Identity.GetUserId();
    var addressView = new AddressViewModel()
    {
        Addresses = customerService.GetCustomerAddresses(user)
    };

    return PartialView("_AddressTablePartial", addressView);
}
```

Figură 23 Implementare partial view

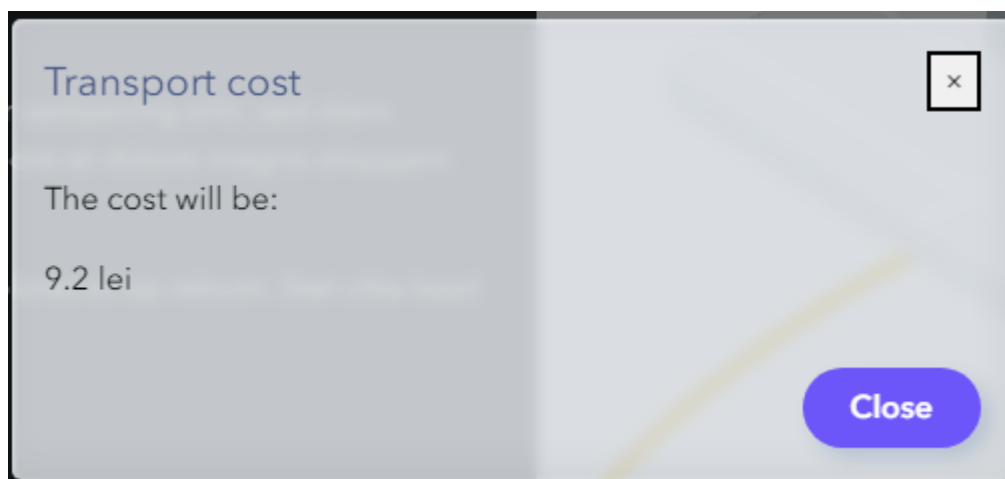




A dark-themed form titled "Cost Estimate". It contains three input fields: "Declared Value:" with a placeholder "Total", "Aprox. Dimensions" with a placeholder "Can be carried on a bike", and "Kilometers" with a placeholder "Kilometers". A blue "Submit" button is at the bottom.

**Figură 25 Cost Estimate**

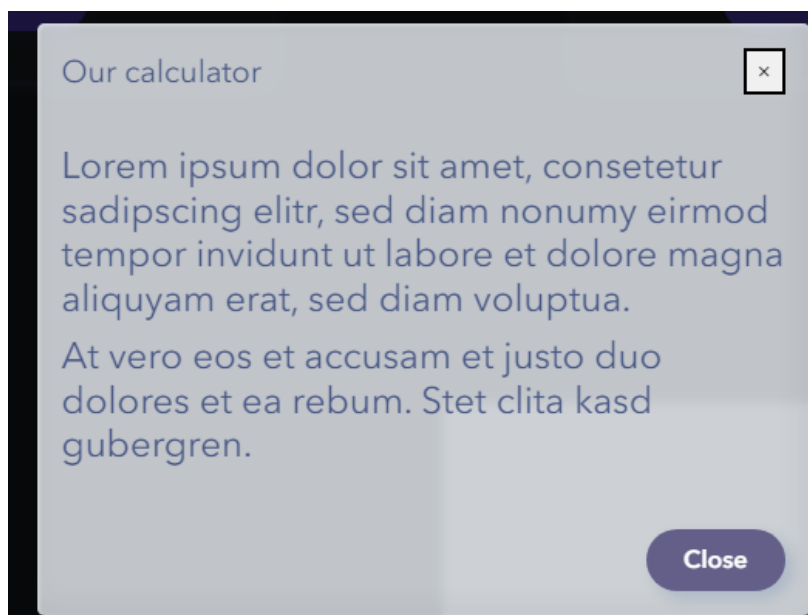
Pentru calculatorul de cost, orice client își poate calcula cu aproximație prețul pe care trebuie să îl plătească curierului, în funcție de 3 clasificări: valoarea declarată a pachetului, dimensiunile aproximative și numărul de kilometri.



A light-themed modal window titled "Transport cost" with a close button (X) in the top right. It displays "The cost will be:" followed by "9.2 lei". A blue "Close" button is at the bottom right.

**Figură 26 Cost Estimate 2**

De asemenea, calculatorul are și un buton de „Read More”, unde deținătorul final al aplicației poate scrie detalii despre cum este calculată rata de transport, etc.



Figură 27 Detalii calculator estimări

## 4.2 Crearea unui cont de utilizator

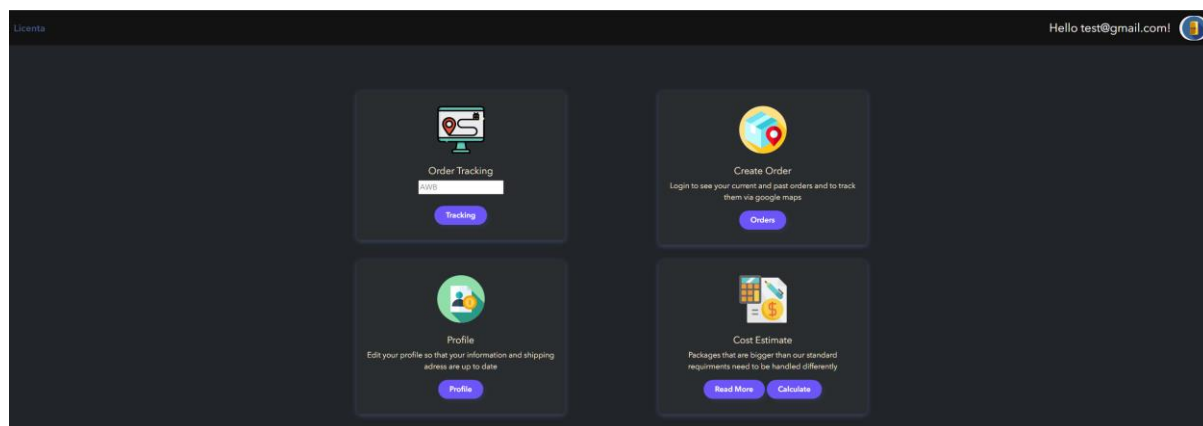
Pentru a crea un cont de utilizator, se va accesa pagina de înregistrare (apăsând pe butonul de „Register” din card-ul de autentificare. După apăsare, utilizatorul va fi redirecționat către pagina de register

Figură 28 Register



### 4.3 Pagina de home (utilizator înregistrat)

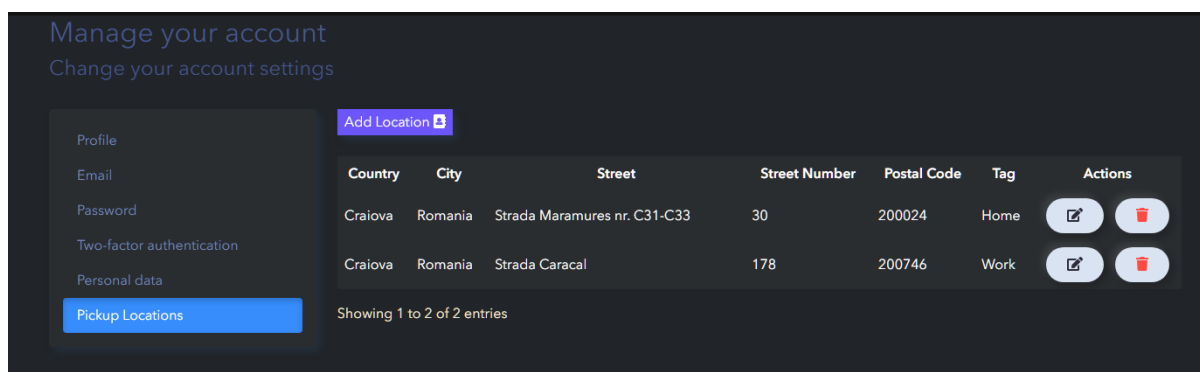
Pagina de start este „îmbunătățită” când un utilizator este înregistrat. Pe lângă card-urile de autentificare și cost estimate se mai adaugă un card pentru comenzi și unul pentru editarea profilului.



Figură 29 Dashboard utilizator inregistrat

### 4.4 Pagina de profil

Apăsând pe butonul de profil, utilizatorul este redirecționat la pagina de editare a profilului, unde poate modifica numele, adresa de email și, în special, locațiile.



Figură 30 Pagina de profil

Manage your account

Change your account settings

- Profile
- Email
- Password
- Two-factor authentication
- Personal data
- Pickup Locations

Profile

Full Name

Phone Number

Save

**Figură 31 Pagina de profil 2**

Manage your account

Change your account settings

- Profile
- Email
- Password
- Two-factor authentication
- Personal data
- Pickup Locations

Add Location

Country

City

Street

Street number

PostalCode

Tag

Close Add Location

Country	City	Street	Street Number	Postal Code	Tag	Actions
Craiova	Romania			200024	Home	
Craiova	Romania		8	200746	Work	

Showing 1 to 2 of 2

**Figură 32 Add location**

Pentru a adăuga o locație, este nevoie de următoarele informații obligatorii: Tara, Oraș, Strada, Numărul străzii, codul poștal. Opțional, utilizatorul își poate alege și un „Tag” pentru locația adăugată, fie la creare, fie la editare. Acest tag poate fi ceva familiar pentru a fi mai ușor de găsit locația în liste (de ex. Home). Dacă un tag este introdus, acesta va apărea în locul adresei când este aleasă pentru o comandă.

## 4.5 Pagina de comenzi

Pagina de comenzi include un tabel, făcut cu ajutorul plugin-ului „DataTables”, care include detalii despre comenzi cum ar fi: adresa de ridicare, adresa de destinație, recipientul și informațiile despre el (intr-un dropdown), statusul comenzii, awb, prețul și opțiunea de a deschide google maps pentru a vedea ruta pe care va merge șoferul pentru a efectua comanda.

Pickup Address	Delivery Address	Recipient	Status	AWB	Price	Action
Home	Craiova, Strada Pandurilor	test	Created	J3dt0Yfj	222.00	
Work	Craiova, Strada Carpenului	test2	Created	u8FAnHeu	1200.00	

Showing 1 to 2 of 2 entries

Country: Romania  
City: Craiova  
Street: Strada Carpenului  
Street number: 30  
Postal Code: 200110

Previous 1 Next

Figură 33 Dashboard orders

După apăsarea butonului de „New Order” userul trebuie sa introducă anumite date pentru o nouă comandă:

- Tara
- Oraș
- Strada
- Numărul străzii
- Cod poștal
- Informații despre recipient
- Preț

New order

×

Pickup Address:

Home

Country

City

Street

StreetNumber

PostalCode

Recipient Name

Recipient Email

Recipient Phone Number

Price

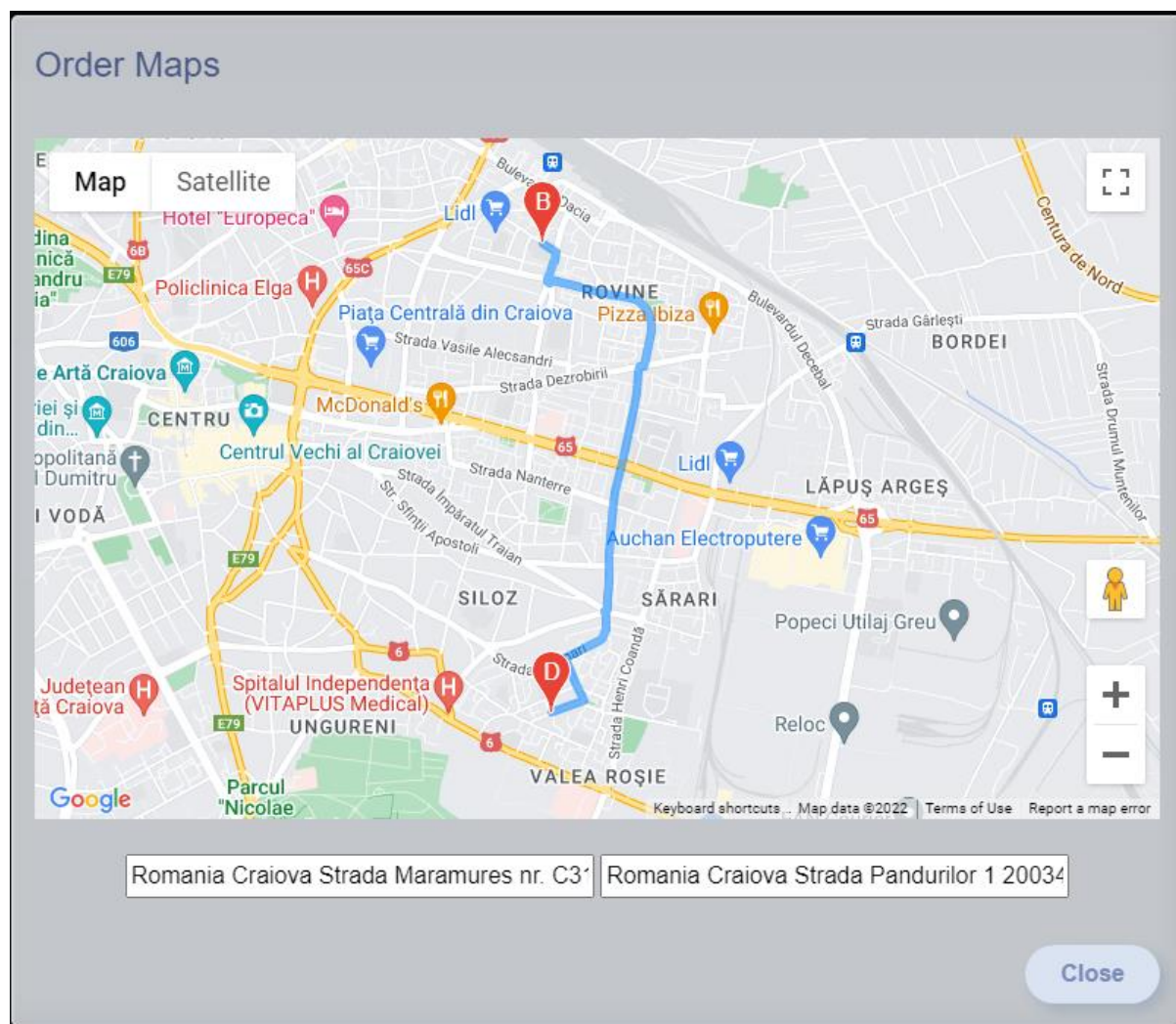
0

Close

Save changes

**Figură 34 New order**

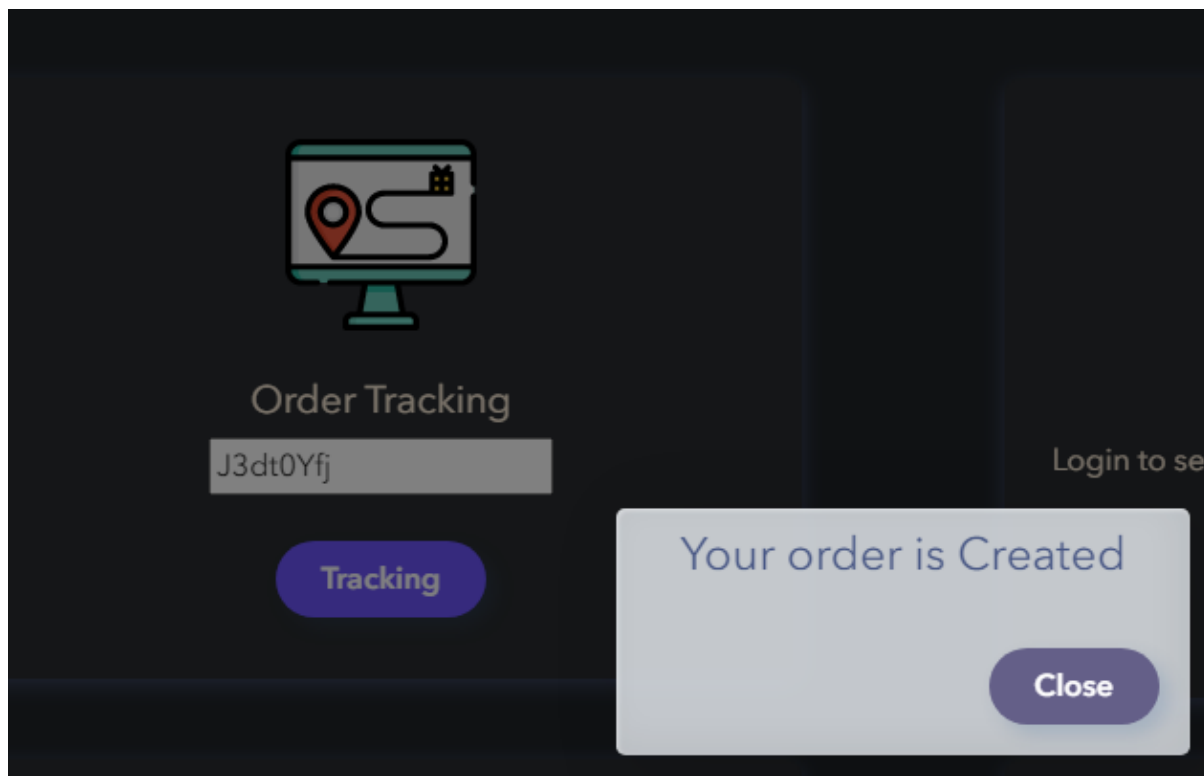
Adresele de ridicare pot fi alese dintr-un meniu de dropdown, unde meniul arata fie câteva detalii despre adresa (cod poștal, oraș) sau tag-ul (de ex. Home). După ce se apasă pe butonul de „Save Changes”, o comandă este salvată. Aceasta se poate vedea și pe google maps apăsând SVG-ul din coloana de „Action”:



Figură 35 Order Maps

## 4.6 AWB Tracking

Opțiunea de AWB Tracking este valabilă atât pentru un user înregistrat cât și pentru un user fără un cont. În cardul corespunzător, utilizatorul poate introduce awb-ul comenzii, pe care l-a văzut când a creat comanda pentru a afla statusul comenzii.



Figură 36 AWB Tracking

## **5 CONCLUZII**

În concluzie, această aplicație web a fost dezvoltată cu scopul de a adăuga în repertoriul aplicațiilor de curierat o aplicație dinamică, care poate fi utilizată de mai multe tipuri de firme de transport și/sau curierat. Folosind aceasta aplicație, angajaților le va fi mult mai ușor să gestioneze toate flow-urile de care sunt responsabili, de la atribuirea de rute, până la livrarea coletelor.

### **5.1 Îmbunătățiri și dezvoltare**

Pentru direcțiile de dezvoltare, se poate introduce o aplicație mobilă, care poate fi folosită de către șoferi pentru a avea un tracking la comenzi mult mai ușor. Aceasta ar putea lega rutele direct la aplicația Google Maps sau la Waze, pentru a vedea în timp real atât locația șoferului, cât și locația de ridicare/livrare (la fel ca un șofer Uber).

## 6 BIBLIOGRAFIE

Azure documentation (2022), accesibil la: <https://docs.microsoft.com/en-us/azure/?product=popular>

Bootstrap (2022), accesibil la: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>

C. Sboră (2021), *platforme de laborator PAW*

Google Maps documentation (2022), accesibil la: <https://developers.google.com/maps/documentation>

jQuery (2022), accesibil la: <https://jquery.com/>

R. C. Martin, M. Martin (2006), *Agile Principles, Patterns, and Practices in C#*, 9780132055109

S. McConnell (2004), *Code Complete, Second Edition*, 9789350041246

Wikipedia (2022), *accesibil la*: <https://ro.wikipedia.org/>

W3Schools (2022), accesibil la: <https://www.w3schools.com/>



## **SITE-UL WEB AL PROIECTULUI**

<https://licenta20220601113900.azurewebsites.net/>

## **8 CD/DVD**