|  |  |  |
| --- | --- | --- |
|  | 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. Sbora 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. Sbora 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-*Scrieri 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. Sbora 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 situaţii neplăcute se recomandă:

* plasarea între ghilimele a citatelor directe și indicarea referinței într-o listă corespunzătoare la sfărşitul lucrării,
* indicarea în text a reformulării unei idei, opinii sau teorii și corespunzător în lista de referinţe 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 referinţelor poate fi omisă dacă se folosesc informaţii 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. Sbora 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  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Tipul proiectului | | Cercetare  □ | Proiectare  □ | Realizare practică □ | Altul  [*se detaliază*] |
| Aparatul matematic utilizat | | Simplu  □ | Mediu  □ | Complex □ | Absent  □ |
| Utilitate | | Contract de cercetare □ | Cercetare internă □ | Utilare  □ | Altul  [*se detaliază*] |
| Redactarea lucrării | | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Partea grafică, desene | | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Realizarea practică | Contribuția autorului | Insuficientă  □ | Satisfăcătoare □ | Mare  □ | Foarte mare  □ |
| Complexitatea  temei | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Analiza cerințelor | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Arhitectura | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Întocmirea specificațiilor funcționale | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Implementarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Testarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Funcționarea | Da  □ | Parțială  □ | Nu  □ | |
| Rezultate experimentale | | Experiment propriu  □ | | Preluare din bibliografie  □ | |
| Bibliografie | | Cărți | Reviste | Articole | Referințe web |
| Comentarii  și  observații | |  | | | |

În concluzie, se propune:

|  |  |
| --- | --- |
| ADMITEREA PROIECTULUI  □ | RESPINGEREA PROIECTULUI  □ |

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. Sbora Cătălin, pentru sfaturile și răbdarea de care a dat dovadă în călăuzirea mea pe tot drumul parcurs,cât și pentru documentația,suportul şi feedback-ul acordat cu privire la dezvoltarea aplicației.

În aceeaşi măsură doresc să aduc mulțumiri întregii echipe profesorale cu care am

interacționat în aceşti 4 ani, întrucât şi-au adus aportul în structurarea cunoştințelor acumulate, în utilizarea acestora în practică,adică în întreaga mea formare profesională.

**CUPRINSUL**

[1 Introducere 13](#_Toc106486602)

[1.1 Scopul 13](#_Toc106486603)

[1.2 Motivația 13](#_Toc106486604)

[1.3 Descriere generală 13](#_Toc106486605)

[1.3.1 Tehnologii și aplicații folosite 13](#_Toc106486606)

[1.3.2 Tehnologii și framework-uri folosite 14](#_Toc106486607)

[1.4 Funcționalitățile aplicației 17](#_Toc106486608)

[1.4.1 Funcționalitățile administratorului 17](#_Toc106486609)

[1.4.2 Funcționalitățile șoferului 17](#_Toc106486610)

[1.4.3 Funcționalitățile dispeceratului 17](#_Toc106486611)

[1.4.4 Funcționalitățile clientului 17](#_Toc106486612)

[1.5 Structura lucrării 18](#_Toc106486613)

[2 Arhitectura sistemului 19](#_Toc106486614)

[2.1 Azure cloud services 21](#_Toc106486615)

[2.2 Structura bazei de date 24](#_Toc106486616)

[2.3 Proiectarea aplicației 28](#_Toc106486617)

[3 CerinȚe pentru sistemul implementat 31](#_Toc106486618)

[3.1 Funcționalitățile administratorului 31](#_Toc106486619)

[3.1.1 Autentificare 31](#_Toc106486620)

[3.1.2 Dashboard conturi 31](#_Toc106486621)

[3.1.3 Adăugare cont 31](#_Toc106486622)

[3.1.4 Editare cont 31](#_Toc106486623)

[3.1.5 Logout 31](#_Toc106486624)

[3.2 Funcționalitățile dispeceratului 31](#_Toc106486625)

[3.2.1 Autentificare 31](#_Toc106486626)

[3.2.2 Dashboard vehicule 32](#_Toc106486627)

[3.2.3 Dashboard comenzi 32](#_Toc106486628)

[3.2.4 Dashboard șoferi 32](#_Toc106486629)

[3.2.5 Dashboard rute 32](#_Toc106486630)

[3.3 Funcționalitățile șoferului 32](#_Toc106486631)

[3.3.1 Operații pe ruta curenta 32](#_Toc106486632)

[3.4 Funcționalitățile clientului final 33](#_Toc106486633)

[3.4.1 Estimare cost transport 33](#_Toc106486634)

[3.4.2 Editare profil 33](#_Toc106486635)

[3.4.3 Adăugare locații 33](#_Toc106486636)

[3.4.4 Editare locații 33](#_Toc106486637)

[3.4.5 Ștergere locații 33](#_Toc106486638)

[3.4.6 Creare comanda 34](#_Toc106486639)

[3.4.7 Verificare stadiu comandă 34](#_Toc106486640)

[3.5 Detalii de implementare 35](#_Toc106486641)

[3.5.1 Rutarea folosind google maps 35](#_Toc106486642)

[3.5.2 Utilizarea si configurarea serviciilor 36](#_Toc106486643)

[3.5.3 Utilizarea partial views 37](#_Toc106486644)

[4 Manualul utilizatorului 38](#_Toc106486645)

[4.1 Pagina de home (utilizator neînregistrat) 38](#_Toc106486646)

[4.2 Crearea unui cont de utilizator 40](#_Toc106486647)

[4.3 Pagina de home (utilizator înregistrat) 41](#_Toc106486648)

[4.4 Pagina de profil 41](#_Toc106486649)

[4.5 Pagina de comenzi 43](#_Toc106486650)

[4.6 AWB Tracking 46](#_Toc106486651)

[5 Concluzii 47](#_Toc106486652)

[5.1 Îmbunătățiri si dezvoltare 47](#_Toc106486653)

[6 Bibliografie 48](#_Toc106486654)

[Site-ul web al proiectului 49](#_Toc106486655)

[8 CD/DVD 50](#_Toc106486656)

[Figură 1 Diagrama MVC 16](#_Toc106486657)

[Figură 2 Azure services 19](#_Toc106486658)

[Figură 3 Arhitectura sistemului 20](#_Toc106486659)

[Figură 4 Azure subscription 21](#_Toc106486660)

[Figură 5 Azure resource group 21](#_Toc106486661)

[Figură 6 Azure app service plan 22](#_Toc106486662)

[Figură 7 Azure sql server 22](#_Toc106486663)

[Figură 8 Azure database/query editor 22](#_Toc106486664)

[Figură 9 Azure dashboard 23](#_Toc106486665)

[Figură 10 Azure application insights 23](#_Toc106486666)

[Figură 11 Diagrama de clase (1) 25](#_Toc106486667)

[Figură 12 Diagrama de clase (2) 26](#_Toc106486668)

[Figură 13 Diagrama de clase (3) 26](#_Toc106486669)

[Figură 14 Modelul relational al bazei de date 27](#_Toc106486670)

[Figură 15 Diagrama cazuri de utilizare – Client 28](#_Toc106486671)

[Figură 16 Diagrama cazuri de utilizare – Administrator 29](#_Toc106486672)

[Figură 17 Diagrama cazuri de utilizare – Dispecerat 29](#_Toc106486673)

[Figură 18 Diagrama cazuri de utilizare - Șofer 30](#_Toc106486674)

[Figură 19 Rutare google maps 1 35](#_Toc106486675)

[Figură 20 Rutare google maps 2 35](#_Toc106486676)

[Figură 21 Servicii 36](#_Toc106486677)

[Figură 22 Partial view locations 37](#_Toc106486678)

[Figură 23 Implementare partial view 37](#_Toc106486679)

[Figură 24 Dashboard user nelogat 38](#_Toc106486680)

[Figură 25 Cost Estimate 39](#_Toc106486681)

[Figură 26 Cost Estimate 2 39](#_Toc106486682)

[Figură 27 Detalii calculator estimări 40](#_Toc106486683)

[Figură 28 Register 40](#_Toc106486684)

[Figură 29 Dasboard utilizator inregistrat 41](#_Toc106486685)

[Figură 30 Pagina de profil 41](#_Toc106486686)

[Figură 31 Pagina de profil 2 42](#_Toc106486687)

[Figură 32 Add location 42](#_Toc106486688)

[Figură 33 Dashboard orders 43](#_Toc106486689)

[Figură 34 New order 44](#_Toc106486690)

[Figură 35 Order Maps 45](#_Toc106486691)

[Figură 36 AWB Tracking 46](#_Toc106486692)

# Introducere

## Scopul

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

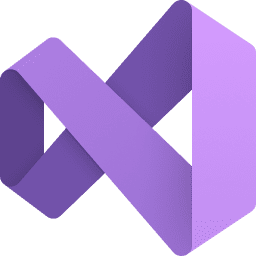
## Motivația

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

## Descriere generală

### 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 facută folosind Azure, portalul celor de Microsoft pentru management si Visual studio pentru publicare și modificare.

 Icon

Description automatically generated

În plus, pentru a avea un istoric al dezvoltării si 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 cat 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 in mod dinamic distanța și ruta pe hartă.

### 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 si ORM (Object–Relational Mapper). ORM-ul ne permitem 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 serviciul 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 si 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 harți) 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 customizate 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 cate 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]](#_Bibliografie)

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 in detaliu in [capitolul 1.5](#_Azure_cloud_services).

## Funcționalitățile aplicației

### 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, in acest fel, primul administrator poate adaugă si alte conturi de admin, conturi de șofer sau de dispecerat.

### Funcționalitățile șoferului

Șoferul este utilizatorul care acces la ruta asignata acestuia de către dispecerat. După ce ruta este începuta. șoferul își poate începe munca. Odată ce o comanda a fost dusa la bun sfârșit, sau au fost probleme cu ea, șoferul poate apasă fie pe butonul de „Complete”, fie pe cel de „Cancel” pentru a actualiza statusul comenzii. O data ce toate comenzile dintr-o ruta au fost adusa la un status de final, șoferul poate opri ruta.

### Funcționalitățile dispeceratului

Dispeceratul are acces la toate comenzile create de utilizatori. Aceștia pot crea rute, le poate lega la o mașina libera si poate adaugă comenzi la aceasta ruta.

### Funcționalitățile clientului

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

## Structura lucrării

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

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

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

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

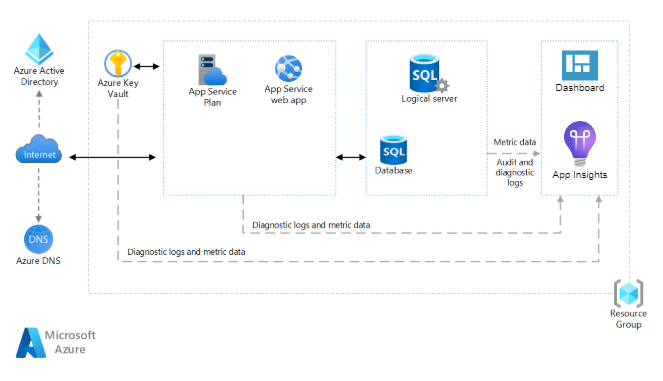
Capitolul 5: [„Concluzii”](#_Concluzii) este capitolul care cuprinde concluziile dobândite în urma implementării acestui proiect, detalii despre limitările sistemului.

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

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

# 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

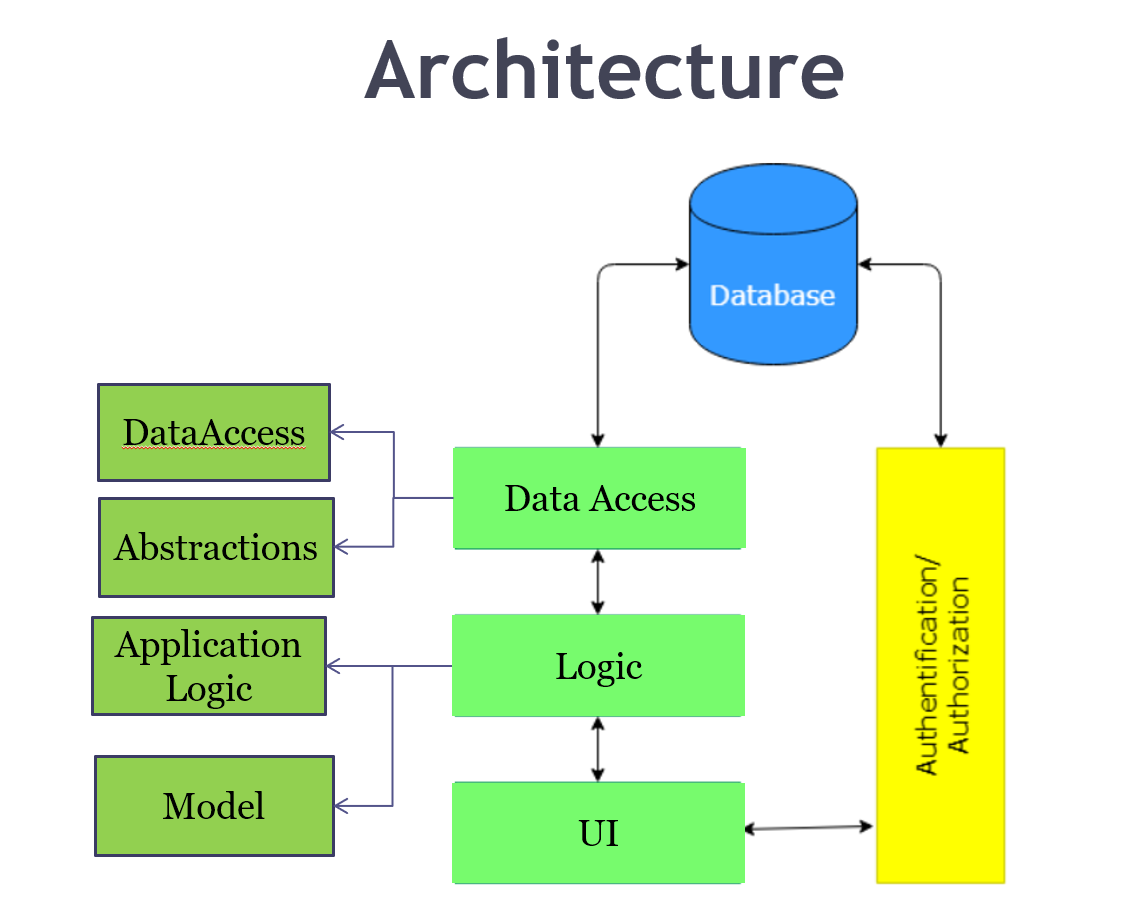
Toata aplicația este găzduită în cadrul Azure. Aceasta este înglobată într-un grup de resurse, unde se afla atât partea de secrets și aplicația în sine (app service), baza de date dar și partea de diagnoza, cum ar fi Dashboard sau Application Insights.

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 consistă 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 pricipiilor 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]](#_Bibliografie)



Figură 3 Arhitectura sistemului

## 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]](#_Bibliografie)

* Cont, a fost folosit cel de „@robotics.ucv” pentru a avea beneficiile unui „Azure for students”. Acesta incepe 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 si 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: defineste un set de resurse care sunt asignate către aplicația respectiva, care se plătesc, pentru ca aceasta sa ruleze pe cloud.

Graphical user interface, text, application

Description automatically generated

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.

A picture containing text

Description automatically generated

Figură 8 Azure database/query editor

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

Graphical user interface, application

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

Figură 10 Azure application insights

## Structura bazei de date

Baza de date este una relațională și a fost creată folosind strategia CodeFirst. Mai exact, înainte sa fie creată baza de date, au fost create modelele, iar fiecare model reprezintă o entitate a tabelelor 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.

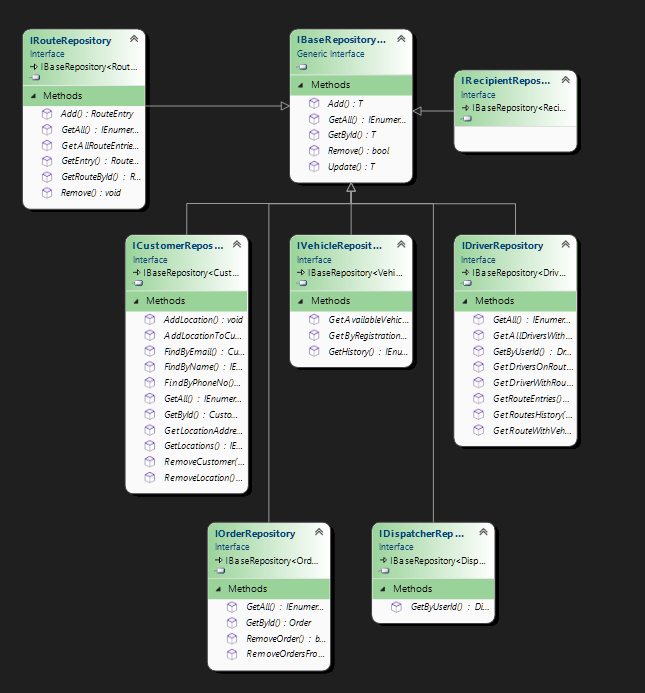
Mai jos au fost adăugate diagramele de clase care sunt împărțite în funcție de principalele funcționalități. Deoarece fiecare model avea nevoie de un Id, sub forma de Guid, pentru a putea fi găsit in baza de date, toate modele moștenesc o clasa comuna și anume DataEntity:



Figură 11 Diagrama de clase (1)

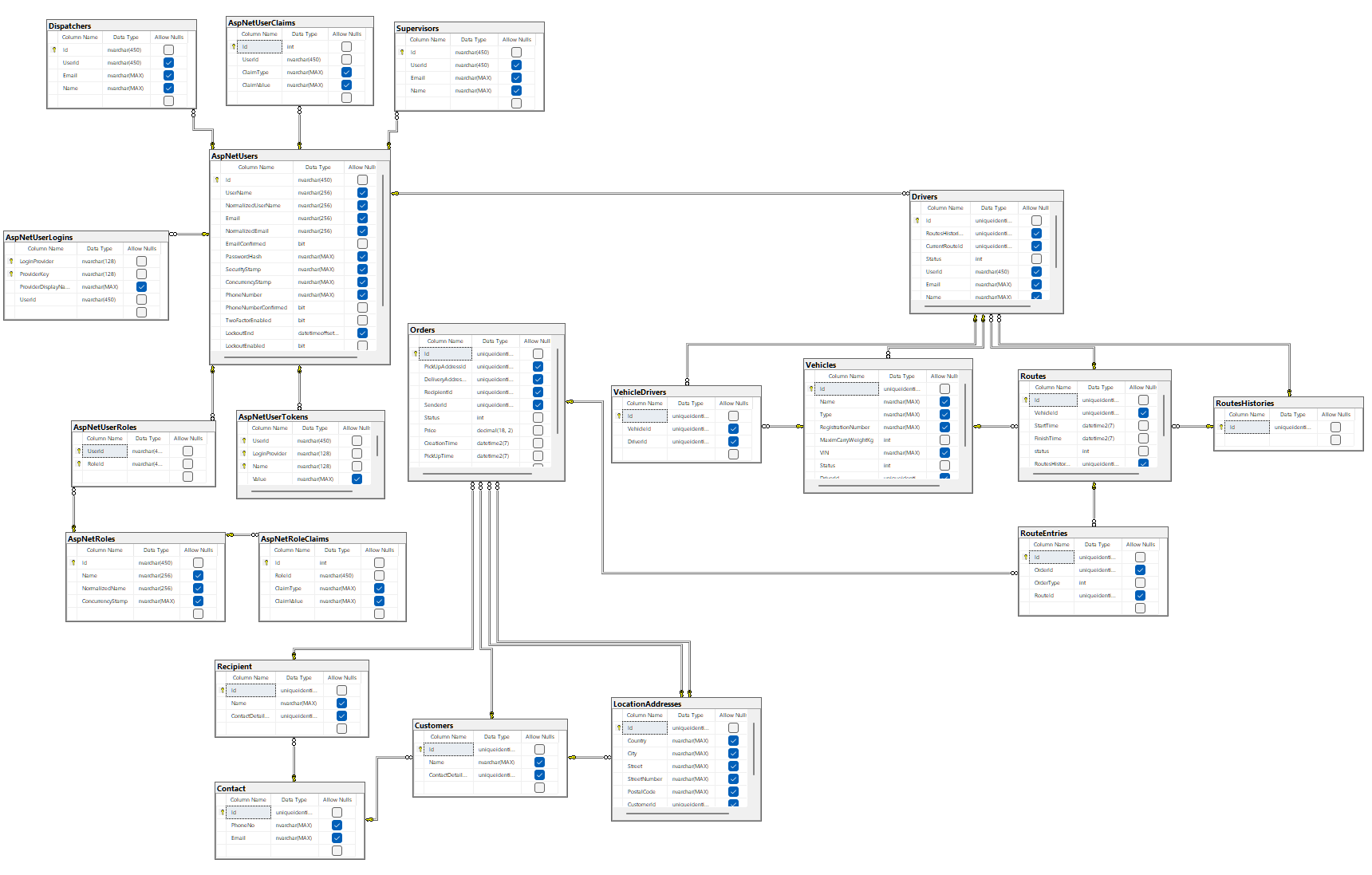


Figură 12 Diagrama de clase (2)



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

## Proiectarea aplicației

Aplicația de curierat a fost concepută cu scopul de a fi folosită de toate persoanele care ar putea exista în cercul 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

Diagram

Description automatically generated

Figură 15 Diagrama cazuri de utilizare – Client

* Admin

Diagram

Description automatically generated

Figură 16 Diagrama cazuri de utilizare – Administrator

* Dispecerat

Diagram

Description automatically generated

Figură 17 Diagrama cazuri de utilizare – Dispecerat

* Șofer

Diagram

Description automatically generated

Figură 18 Diagrama cazuri de utilizare - Șofer

# CerinȚe pentru sistemul implementat

## Funcționalitățile administratorului

### 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.

### 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.

### Adăugare cont

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

### Editare cont

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

### Logout

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

## Funcționalitățile dispeceratului

### 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 redirectat către pagina de dashboard.

### 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).

### 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.

### 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.

### Dashboard rute

Această pagină este folosită pentru diferite acțiuni ce se pot efectua asupra unei rute. Dispecerul poate crea o rută noua, fiind obligat sa îi adauge un vehicul care are statusul de „Free”. Ruta este creată fără nicio comandă, acestea trebuind sa 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. Daca 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.

## Funcționalitățile șoferului

### Operații pe ruta curenta

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. In 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 devening „Delivering”.

Odată ce ruta este pornită, fiecare pachet are doua 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.

## Funcționalitățile clientului final

### 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.

### Editare profil

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

### Adăugare locații

Din pagina de editare a profilului, utilizatorul își poate adaugă si 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).

### Editare locații

Utilizatorul poate edita toate datele unei locații deja create anterior.

### Ștergere locații

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

### 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

### Verificare stadiu comandă

Stadiul unei comenzi poate fi văzut din două locuri: Daca 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 sa 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.

## Detalii de implementare

### 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.

Text

Description automatically generated

Figură 19 Rutare google maps 1

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

In 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.

Text

Description automatically generated

Figură 20 Rutare google maps 2

### Utilizarea si configurarea serviciilor

ASP.NET Core utilizează injecția de dependențe ca o caracteristică fundamentală pentru a gestiona dependențele în întregul cadru. Pentru ca cadrul de injecție 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 activam, chiar si cele personalizate, create de noi.

Text

Description automatically generated

Figură Servicii

Pe lingă 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)

### 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.

A screenshot of a computer

Description automatically generated with medium confidence

Figură Partial view locations

Text

Description automatically generated

Figură Implementare partial view

# Manualul utilizatorului

## Pagina de home (utilizator neînregistrat)

Pagina de home poate fi accesata atât de un utilizator autentificat cât și de unul care nu s-a autentificat. Pe aceasta pagină utilizatorul poate vedea câteva detalii despre aplicație, o parte de „About Us”, o parte de contact și un estimator de costuri de transport.

Graphical user interface, application

Description automatically generated

Figură 24 Dashboard user nelogat

Graphical user interface, text, application, chat or text message

Description automatically generated

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.

Text

Description automatically generated with medium confidence

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.

Text, timeline

Description automatically generated

Figură 27 Detalii calculator estimări

## 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

Graphical user interface, text, application, chat or text message

Description automatically generated

Figură 28 Register

## Pagina de home (utilizator înregistrat)

Pagina de start este „îmbunătățita” 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.

Graphical user interface

Description automatically generated

Figură 29 Dasboard utilizator inregistrat

## 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.

A screenshot of a computer

Description automatically generated with medium confidence

Figură 30 Pagina de profil

Graphical user interface, text, application, chat or text message

Description automatically generated

Figură 31 Pagina de profil 2

A screenshot of a computer

Description automatically generated with medium confidence

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). Daca un tag este introdus, acesta va apărea în locul adresei când este aleasă pentru o comandă.

## 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.

A screenshot of a computer screen

Description automatically generated with medium confidence

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

A picture containing graphical user interface

Description automatically generated

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”:

Map

Description automatically generated

Figură 35 Order Maps

## AWB Tracking

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

Graphical user interface, application

Description automatically generated

Figură 36 AWB Tracking

# 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 sa gestioneze toate flow-urile de care sunt responsabili, de la atribuirea de rute, până la livrarea coletelor.

## Îmbunătățiri si 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).

# 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. Sbora (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/>

# CD/DVD