**UNIVERSITATEA DIN**

**BUCUREȘTI**

**FACULTATEA DE**

**MATEMATICĂ ȘI**

**INFORMATICĂ**

**SPECIALIZAREA INFORMATICĂ**

**Lucrare de licență**

**Platformă web a unui comerciant de autovehicule**

**Absolvent**

**Vasile George-Cristian**

**Coordonator științific**

**Lect. dr. Ana Cristina Iova**

**București, iulie 2023**

**Rezumat**

Tema lucrării constă în dezvoltarea unei aplicații web a cărui scop fiind gestionarea unui lanț de magazine al unui comerciant de autovehicule rulate. Dezvoltarea aplicației va necesita atât proiectarea unei interfețe grafice intuitive, cât și a unui Web API care să gestioneze logica necesară funcționalităților și a stocării datelor. Aplicația va fi folosită atât de angajații magazinului cât și de potențialii clienți.

Funcționalitățile dedicate angajaților vor include adăugarea și modificarea locațiilor din lanț, gestionarea autovehiculelor prezente în locații și vizualizarea programărilor făcute de clienți prin intermediul platformei.

Clienții vor putea viziona autovehiculele scoase la vânzare. După crearea unui cont, aceștia vor putea face programări prin intermediul platformei. La crearea contului, clienții vor avea opțiunea de a include date personale pe baza cărora aplicația le va recomanda vehicule.

**Abstract**

The thesis’ subject consists of developing a web application with the purpose of managing a used car dealership chain. The application’s development will require designing both an intuitive graphical interface and a Web API that will manage the algorithms necessary for providing the application’s functionality and data storage. The application will be used by store employees and potential clients.

Functionalities dedicated to employees will include adding and modifying chain locations, management of the vehicles present in the locations and viewing appointments made by clients through the platform.

The clients will be able to view vehicles up for sale. After creating an account, they will be able to book appointments through the platform. On account creation, the clients will have the option to include some personal info based on which the application will recommend vehicles.

**Cuprins**

[Introducere 4](#_Toc135864328)

[1.1 Prezentarea generală a aplicației 4](#_Toc135864329)

[1.1.1 Descrierea backend-ului 4](#_Toc135864330)

[1.1.2 Descrierea frontend-ului 4](#_Toc135864331)

[1.2 Starea curentă a pieței auto 5](#_Toc135864332)

[1.3 Scopul și motivația alegerii temei 5](#_Toc135864333)

[1.4 Structura lucrării 5](#_Toc135864334)

[1.5 Termeni 5](#_Toc135864335)

[Limbaje și tehnologii 6](#_Toc135864336)

[2.1 Visual Studio Community 6](#_Toc135864337)

[2.1.1 Instalare 6](#_Toc135864338)

[2.1.2 Pachete necesare 7](#_Toc135864339)

[2.2 ASP.NET și C# 7](#_Toc135864340)

[2.3 Microsoft SQL Server și SSMS 8](#_Toc135864341)

[2.4 Node.JS și React 9](#_Toc135864342)

[2.4.1 Pachete auxiliare 9](#_Toc135864343)

[2.5 Visual Studio Code 10](#_Toc135864344)

[2.6 TypeScript și SCSS 10](#_Toc135864345)

[2.7 Alternative 10](#_Toc135864346)

[Descrierea implementării 11](#_Toc135864347)

[Concluzii 12](#_Toc135864348)

**Capitolul I**

# Introducere

## Prezentarea generală a aplicației

Asemenea multor alte aplicații web, aplicația este alcătuită din două componente interconectate. Prima componentă este un Web API – denumit în continuare backend, iar a doua componentă este o pagină web – denumită în continuare frontend. Frontend-ul comunică cu backend-ul prin intermediul unor apeluri de tip REST. Atât textul din interfața vizuală cât și totalitatea codului sunt scrise în limba engleză.

### Descrierea backend-ului

Backend-ul a fost realizat în framework-ul ASP.NET, fiind scris în limbajul C#. Acesta interacționează cu o bază de date SQL Server Express. Baza de date nu este creată folosind comenzi SQL, ci este generată printr-o abordare „code first” prin intermediul Entity Framework. Codul este structurat în conformitate cu „Repository Pattern” si „Specification Pattern”. Fiind un Web API, acesta își expune metodele REST prin intermediul unor controllere.

### Descrierea frontend-ului

Frontend-ul este realizat cu ajutorul librăriei React și este scris în limbajul Typescript. Acesta își expune funcționalitățile în funcție de starea utilizatorului (autentificat/ neautentificat) și rolul acestuia. Comunicarea cu backend-ul se face prin apelarea metodelor REST menționate în secțiunea 1.1.1.

Secțiunea de autentificare – „auth” – este împărțită în 4 pagini: o pagină de autentificare, una de înregistrare, una pentru confirmarea prin email a unui cont nou și una pentru resetarea parolei unui cont existent.

Secțiunea principală – „main” – pune la dispoziție o pagină în care poate fi vizionată și filtrată lista de autovehicule și o pagină secundară care pune la dispoziție detalii amănunțite despre un vehicul selectat din listă. Funcționalitățile vizibile doar administratorilor sunt expuse pe pagina cu lista de autovehicule prin intermediul unor formulare de tip „Dialog”.

## 1.2 Starea curentă a pieței auto

De la apariția autovehiculelor până în trecutul apropiat piața a rămas in mare parte neschimbată, fiind dominată de motoarele ce ard combustibili fosili. În prezent, anumiți factori sociopolitici au demarat începuturile „electrificării” autovehiculelor, împingând dezvoltarea de tehnologii noi în domeniul auto. Această schimbare rapidă a pieței de la stabilitate la volatilitate a cauzat multe platforme de acest tip să nu poată ține pasul cu noile tehnologii.

## Scopul și motivația alegerii temei

Domeniul dezvoltării de aplicații web este unul care mă atrage deoarece necesită atât o aprofundare a metodelor de stocare și manipulare a datelor în backend, cât și proiectarea unei interfețe vizuale atractive în frontend, buna implementare a ambelor componente fiind crucială în realizarea unei platforme de succes.

Am ales această temă deoarece sunt nemulțumit de experiențele mele anterioare cu platforme de acest tip. Câteva probleme pe care îmi doresc să le rezolv sunt lipsa algoritmilor de recomandare satisfăcători și inabilitatea de a filtra conținutul pe baza unor atribute mai specifice. În plus, lucrarea își propune abordarea problemelor de scalabilitate cauzate de schimbările menționate anterior, în subcapitolul 1.2.

## Structura lucrării

NEFINAL, va fi modificat ulterior

Lucrarea este structurată în patru capitole, conținutul fiecăruia fiind prezentat în continuare.

Primul capitol conține o descriere succintă a aplicației, starea curentă a pieței pe care se bazează funcționalitatea, scopul lucrării și motivația alegerii temei.

În al doilea capitol sunt descrise limbajele, tehnologiile și programele utilizate în implementarea aplicației, cât și motivația alegerii acestora. De asemenea, sunt expuse alternative viabile.

Al treilea capitol prezintă tehnicile de implementare și structura codului.

În al patrulea capitol sunt descrise concluziile.

## Termeni

Acest subcapitol poate fi folosit pentru a defini termenii precum callback, request, debugging, token, framework, care nu au neapărat un echivalent în limba română.

**Capitolul II**

# Limbaje și tehnologii

## Visual Studio Community

Visual Studio este un IDE – „Integrated Development Environment”, fiind și singurul care integrează dezvoltarea aplicațiilor pe bază de .NET. Acesta conține multe unelte care facilitează dezvoltarea aplicațiilor, printre care se numără un debugger foarte detaliat ce permite chiar vizualizarea progresului la nivel de fir de execuție. Aplicațiile web pot primi un număr ridicat de cereri simultan așa că este vitală utilizarea firelor de execuție. Începând cu versiunea 2022, Visual Studio integrează și „Github Copilot”, o unealtă bazată pe inteligența artificială ce poate genera cod automat sau pe baza unei descrieri. Versiunea „Community” incorporează majoritatea funcționalităților și este gratuită pentru studenți.

### Instalare

Versiunea „Community” 2022 poate fi obținută de pe pagina oficială Microsoft, la adresa <https://visualstudio.microsoft.com/vs/community/>.[1] Pentru a putea dezvolta o aplicație pe baza ASP.NET trebuie instalată și componenta „ASP.NET and web development” din secțiunea „Web & Cloud”, ilustrată în figura 2.1.

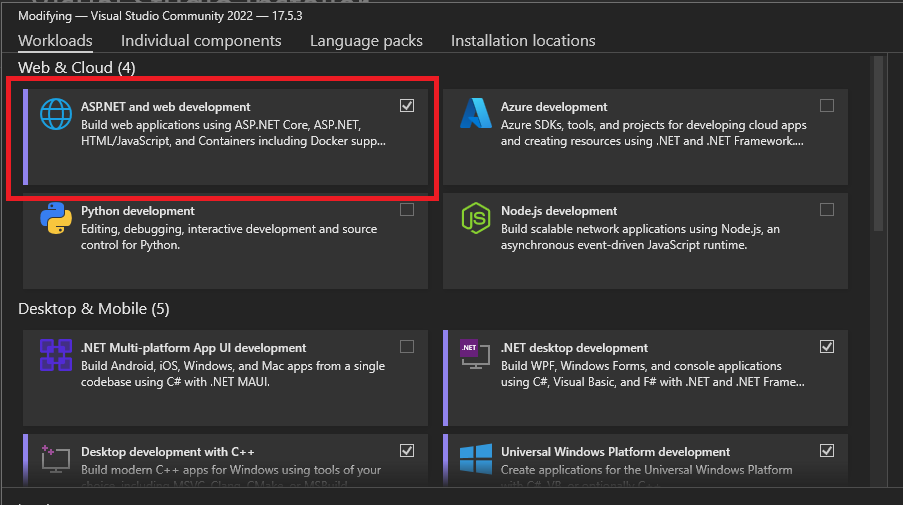


Figura 2.2 – Alegerea unui profil de distribuire

### Pachete necesare

Un alt punct forte al Visual Studio este abilitatea de a adăuga pachete care să extindă funcționalitatea de bază din .NET. În interfața Visual Studio, navigând în panglică pe calea „Tools -> NuGet Package Manager -> Manage NuGet Packages for Solution…” pot fi gestionate pachetele NuGet instalate. Nu este necesar să instalăm aceste pachete manual, fiind descărcate automat la prima rulare a proiectului. În cadrul acestui proiect sunt folosite șase pachete NuGet.

Microsoft.AspNetCore.Authentication.JwtBearer permite folosirea de tokeni JWT pentru autentificare.

Microsoft.AspNetCore.Identity.EntityFrameworkCore este folosit pentru a adăuga funcționalitatea ce se gestionează conturile de utilizator la EntityFramework, generând automat tabelele necesare în baza de date în funcție de configurația făcută.

Microsoft.AspNetCore.Mvc.NewtonsoftJson adaugă abilitatea de a trimite și primi informații în formatul JSON prin intermediul endpoint-urilor din controllere.

Microsoft.EntityFrameworkCore.SqlServer conține logica necesară pentru a face legătura între EntityFramework și o bază de date de tipul Microsoft SQL Server.

Microsoft.EntityFrameworkCore.Tools adaugă comenzi utilitare, cum ar fi „Scaffold-DbContext” ce generează automat clase în C# pe baza tabelelor din baza de date sau comenzile inverse „Add-Migration” și „Update-Database”, prima generând automat o „Migrație” ce presupune operații de adăugare și modificare a tabelelor din baza de date în funcție de entitățile descrise prin clase de C# iar a doua rulând aceste comenzi pentru a actualiza baza de date.

Swashbuckle.AspNetCore permite documentarea endpoint-urilor și testarea acestora dintr-o interfață ce se deschide în interiorul unui navigator web.

## ASP.NET și C#

ASP.NET este un web framework dezvoltat de către Microsoft ce are ca scop facilitarea creări aplicațiilor web. Fiind parte din ecosistemul .NET, o aplicație dezvoltată cu ajutorul acestuia poate rula atât pe Windows, cât și pe MAC OS și Linux. Visual Studio conține două șabloane pentru crearea unui proiect de tip Web API, ilustrate în figura 2.3 aflată la sfârșitul acestui subcapitol.

Pentru a putea rula un Web API bazat pe ASP.NET în versiunea de producție există 2 opțiuni, distribuirea în variantă „Framework-dependent” și distribuirea în variantă “Self-contained”, precum se poate vedea în figura 2.2. Varianta „Framework-dependent” necesită ca versiunea țintă de .NET să fie instalată pe mașina care încearcă să îl ruleze. Opțiunea „Self-Contained” va cauza ca toate librăriile necesare rulării aplicației să fie incluse în directorul final.

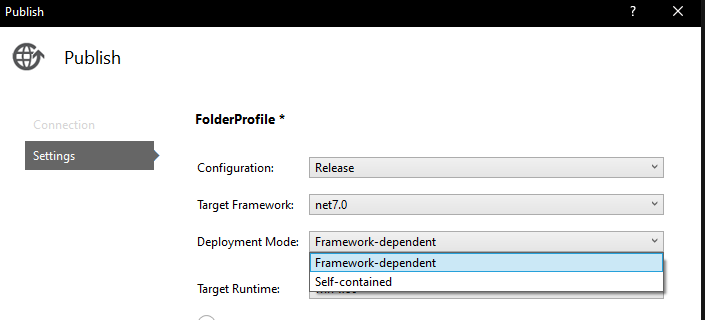
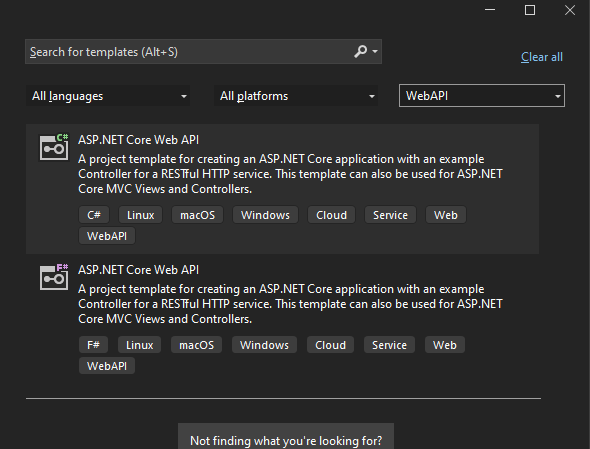
C# este un limbaj de programare orientat pe obiecte cu sintaxa similară limbajului Java, fiind unul dintre cele mai populare limbaje la momentul actual. Am ales acest limbaj în favoarea lui F# deoarece Microsoft pune la dispoziție mai multe resurse de învățare în documentația oficială.

Figura 2.3 – Șabloanele în limbajele C# și F#

Figura 2.2 – Alegerea unui profil de distribuire

## Microsoft SQL Server și SSMS

Entity Framework este compatibil cu o multitudine de baze de date printre care se numără SQL Server, Azure Cosmos DB și Maria DB.[2] Am ales Microsoft SQL Server deoarece, fiind dezvoltat tot de Microsoft, are o compatibilitate ridicată cu ecosistemul .NET. Versiunea „Express” este gratuită și consumă mai puține resurse deoarece nu include toate funcționalitățile care nu sunt necesare dezvoltării unei baze de date „code-first”. Ultima versiune stabilă poate fi descărcată de pe pagina oficială Microsoft la adresa <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>. [3]

SSMS (SQL Server Management Studio) este un program dezvoltat de către Microsoft care incorporează multe unelte care facilitează proiectarea și menținerea unei baze de date relaționale de tip Microsoft SQL Server. În cadrul acestui proiect am folosit doar o mică parte din funcționalitățile existente, fiind totuși foarte util în verificarea manuală a relațiilor generate de către Entity Framework prin intermediul generării automate a unei diagrame entitate-relație. Ultima versiune poate fi descărcată de pe pagina oficială Microsoft la adresa <https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms> din secțiunea „Download SSMS”. [4]

## Node.JS și React

Node.JS este un mediu de execuție open-source pentru limbajul JavaScript.[5] Acesta este versatil și poate fi folosit pentru a construi atât frontend cât și backend. NPM (Node Package Manager) vine la pachet cu instalarea Node.JS și poate fi folosit pentru a instala extensii – numite pachete – în proiect. Adresa la care poate fi găsit installer-ul este <https://nodejs.org/en>. [5]

React este o librărie open-source pentru dezvoltarea de frontend, bazat pe componente plasate într-un arbore vizual. Poate fi folosită atât pentru aplicații web, cât și pentru aplicații mobile sau chiar aplicații desktop (cu Electron). O aplicație React nu are nevoie de Node.JS pentru a rula, ci pentru a fi creată și build-ată. Instalarea se face prin intermediul NPM, cu rularea comenzii „npm install create-react-app” într-un terminal powershell. Pentru a crea o nouă aplicație putem rula comanda “npx create-react-app <numele aplicației>”.

### Pachete auxiliare

Pe lângă conținutul librăriei React, proiectul mai folosește și unele pachete auxiliare. Înainte de a rula aplicația trebuie să instalăm pachetele necesare rulând comanda “npm update” folosind un terminal powershell în directorul aplicației.

Browser-image-compression ajută la comprimarea imaginilor și este folosit în cadrul proiectului pentru a reduce timpul de așteptare cauzat de încărcarea imaginilor prin comprimarea lor înainte de încărcarea către backend.

Pachetele @react-redux și @reduxjs/toolkit sunt o îmbunătățire a mecanismului de stare din React native. React suportă definirea unor variabile de stare care forțează o redesenare a arborelui vizual atunci când valoarea acesteia se schimbă, începând cu componenta în care se află variabila. Implicit, aceste variabile de stare pot fi declarate doar la nivel de componentă iar dacă este necesară comunicarea lor către altă componentă aceasta se face prin intermediul funcțiilor callback sau proprietăților componentelor. Redux toolkit extinde această funcționalitate, permițând definirea unor variabile de stare la nivel global.

Jwt-Decode este folosit pentru extragerea unor informații din tokenul de autentificare de tip JWT. Mai precis, în interiorul aplicației, este folosit pentru a verifica dacă sesiunea utilizatorului curent a expirat fără a mai fi nevoie de trimiterea unui request către backend.

@mui/material conține o multitudine de componente vizuale printre care se numără selectoare, câmpuri de text și casete de selecție. Material UI este destul de comun în aplicațiile moderne, fiind suportat atât de React cât și de Angular și chiar Android.

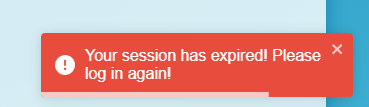
Folosind react-toastify experiența utilizatorilor poate fi îmbunătățită prin afișarea unor notificări de tip toast. O astfel de notificare este exemplificată mai jos, în figura 2.4.

Figura 2.4 – O alertă de tip toast

## Visual Studio Code

## TypeScript și SCSS

## Alternative

**Capitolul III**

# Descrierea implementării

**Capitolul IV**

# Concluzii

**Bibliografie**

[[1]](#_Instalare) <https://visualstudio.microsoft.com/vs/community/> (accesat la 23.05.2023)

[[2]](#_Microsoft_SQL_Server) <https://learn.microsoft.com/en-us/ef/> (accesat la 24.05.2023)

[[3]](#_Microsoft_SQL_Server) <https://www.microsoft.com/en-us/sql-server/sql-server-downloads> (accesat la 24.05.2023)

[[4]](#_Microsoft_SQL_Server) <https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms> (accesat la 24.05.2023)

[5] <https://nodejs.org/en> (accesat la 24.05.2023)