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

PROIECT DE DIPLOMĂ

Mircea-Denis Brașoveanu

COORDONATOR ȘTIINȚIFIC

[*Titlul științific, prenumele și numele coordonatorului*]

Iulie 2025

CRAIOVA

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

[*TITLUL PROIECTULUI DE DIPLOMĂ*]

Mircea-Denis Brașoveanu

COORDONATOR ȘTIINȚIFIC

[*Titlul științific, prenumele și numele coordonatorului*]

Iulie 2025

CRAIOVA

*„The important thing is not to stop questioning. Curiosity has its own reason for existence.”*

Albert Einstein

**DECLARAȚIE DE ORIGINALITATE**

Subsemnatul Mircea-Denis Brașoveanu student la specializarea Calculatoare (în limba engleză) din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoştinţă de cele prezentate mai jos şi că îmi asum, în acest context, originalitatea proiectului meu de licenţă:

* cu titlul [*TITLUL LUCRĂRII*],
* coordonată de [*TITLUL ȘTIINȚIFIC, PRENUMELE ȘI NUMELE COORDONATORULUI*],
* prezentată în sesiunea Iulie 2025.

La elaborarea proiectului de licenţă, se consideră plagiat una dintre următoarele acţiuni:

* 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 şi referinţa 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 obţinute sau a unor aplicaţii realizate de alţi autori fără menţionarea corectă a acestor surse,
* însuşirea totală sau parţială 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 şi indicarea referinţei într-o listă corespunzătoare la sfărşitul lucrării,
* indicarea în text a reformulării unei idei, opinii sau teorii şi 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,

30.06.2025

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de [Calculatoare și Tehnologia Informației / Automatică și Electronică / Mecatronică și Robotică] | Aprobat la data de  …………………  Şef de departament,  Prof. dr. ing.  Nicolae Enescu/  Comin IONETE/  Dorian COJOCARU |

**PROIECTUL DE DIPLOMĂ**

|  |  |
| --- | --- |
| Numele și prenumele studentului/-ei: | Brașoveanu Mircea-Denis |
| Enunțul temei: | [*Titlul lucrării / descrierea pe scurt a temei*] |
| Datele de pornire: | [*Descrierea datelor inițiale de la care s-a început activitatea de cercetare/dezvoltare a tezei*] |
| Conținutul proiectului: | [*Descrierea succintă a conținutului fiecărui capitol al lucrării*] |
| Material grafic obligatoriu: |  |
| Consultații: | [*Periodice/zilnice/săptămânale/lunare*] |
| Conducătorul științific  (titlul, nume și prenume, semnătura): | Șef lucrări/Conferențiar/Profesor dr. ing. ................. |
| Data eliberării temei: | 15.10.2022 |
| Termenul estimat de predare a proiectului: | 01.06.2023 |
| Data predării proiectului de către student și semnătura acestuia: |  |

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de [Calculatoare și Tehnologia Informației / Automatică și Electronică / Mecatronică și Robotică] |  |

**REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC**

|  |  |
| --- | --- |
| Numele și prenumele candidatului/-ei: |  |
| Specializarea: | [*Denumirea oficială a specializării absolvite de candidat*] |
| Titlul proiectului: | [*Titlul lucrării*] |
| 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: [*se detaliază*] |

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,

**PROJECT SUMMARY**

This project focuses on the development of a mobile delivery application built with React Native, offering separate interfaces for both regular users and administrators. A key feature of the app is a personalized recommendation system, based on a custom algorithm I created. Its purpose is to suggest relevant options to users in a smart and adaptive way, depending on their preferences and behavior within the app.

The idea for this project came from my passion for programming, algorithms, and learning new technologies, as well as my growing interest in how digital tools support the business world. I wanted to build something practical and up-to-date, using technologies that are widely adopted in the industry today.

Throughout the process, I dealt with several challenges, such as ensuring smooth cross-platform performance, managing data efficiently, and integrating the recommendation system without compromising user experience. These were addressed through a mix of research, trial and error, and gradual improvements.

My personal contribution includes the design and implementation of the recommendation algorithm, building both the user and admin interfaces, and choosing the right tools and libraries to make the app scalable and efficient. Besides the technical experience, this project taught me a lot about mobile development, structuring real-world apps, and the balance between functionality and usability.

**Keywords**: React Native, mobile development, smart recommendations, delivery application, algorithm design, user interface, admin panel, cross-platform, business application, modern technologies.**ACKNOWLEDGEMNTS**

I would like to thank my project coordinator, [COORDONATOR], for his support and guidance during this final year. His advice helped me stay on track and complete this project.

I am also grateful to all my teachers from the past four years. Thanks to them, I have learned many important things that I will use in my future career.

Last but not least, I want to thank my family for their constant support and encouragement. They have always been there for me, and I appreciate everything they have done.

**CONTENT**

[1 INTRODUCTION 1](#_Toc200299336)

[1.1 Scope 1](#_Toc200299337)

[1.2 Motivation 1](#_Toc200299338)

[2 THEORETICAL CONTEXT AND TECHNOLOGIES 2](#_Toc200299339)

[2.1 React Native 2](#_Toc200299340)

[2.2 Expo 4](#_Toc200299341)

[2.3 .NET 5](#_Toc200299342)

[2.4 Entity Framework 6](#_Toc200299343)

[2.5 SQL Server 8](#_Toc200299344)

[2.6 Visual Studio 9](#_Toc200299345)

[2.7 Visual Studio Code 10](#_Toc200299346)

[2.8 GitHub 11](#_Toc200299347)

[3 Objectives and benchmarks 12](#_Toc200299348)

[3.1 Problem Description 12](#_Toc200299349)

[3.2 Objectives 12](#_Toc200299350)

[3.3 Constraints 13](#_Toc200299351)

[3.4 Identifying User Problems 13](#_Toc200299352)

[3.5 Functional Requirements 13](#_Toc200299353)

[3.6 Non-Functional Requirements 13](#_Toc200299354)

[4 Software Architecture 14](#_Toc200299355)

[4.1 General Architecture 14](#_Toc200299356)

[4.2 Database Structure 14](#_Toc200299357)

[4.2.1 Stored Procedures 16](#_Toc200299358)

[4.2.2 SQL Server Agent Jobs 16](#_Toc200299359)

[4.3 Backend Structure 17](#_Toc200299360)

[4.4 Frontend Structure 17](#_Toc200299361)

[5 Software Implementation 17](#_Toc200299362)

[React Native Maps 17](#_Toc200299363)

[6 Conclusions and future improvements 18](#_Toc200299364)

[7 Bibliography 18](#_Toc200299365)

[A. Source Code 20](#_Toc200299366)

[B. Site-ul web al proiectului 21](#_Toc200299367)

[C. CD / DVD 22](#_Toc200299368)

[Index 23](#_Toc200299369)

**FIGURES LIST**

[Figure 1: React Native 2](#_Toc200299370)

[Figure 2: Cross-platform apps with one codebase [2] 2](#_Toc200299371)

[Figure 3: React Native is being used in thousands of apps [2] 3](#_Toc200299372)

[Figure 4: Expo 4](#_Toc200299373)

[Figure 5: .NET 5](#_Toc200299374)

[Figure 6: Entity Framework 6](#_Toc200299375)

[Figure 7: SQL Server 8](#_Toc200299376)

[Figure 8: Visual Studio 9](#_Toc200299377)

[Figure 9: Visual Studio Code 10](#_Toc200299378)

[Figure 10: GitHub 11](#_Toc200299379)

[Figure 11: Database ERD (entity-relationship diagram) 15](#_Toc200299380)

[Figure 12: Stored Procedures Structure 16](#_Toc200299381)

[Figure 13: SQL Server Agent Job Configuration 16](#_Toc200299382)

**TABLES LIST**

[Tabelul 1. Nume de utilizatori și valorile rezumat ale parolelor acestora 5](#_Toc309893145)

# INTRODUCTION

## Scope

The scope of this thesis is designing and creating a mobile delivery application with React Native. There are three primary sections to the app: administrator view, delivery user view and customer view. The app's unique feature is its SMART suggestion system, which was developed using a custom-made algorithm to provide recommendations for users based on the system orders.

This project covers every step of creating the application, from organizing the structure, selecting the appropriate technologies, creating the user interfaces, and putting the features into use and testing them. The goal is to design an user-friendly software that follows current mobile development standards.

The application was designed to offer a modality for smaller stores to sell their products using their own dedicated app.

The study also looks at how delivery apps' user experience might be enhanced by clever algorithms. The finished item is a working prototype that can be expanded upon.

## Motivation

I have always been interested in algorithms, programming, and learning new technologies, which is why I picked this topic. I liked working on real-world projects during my education; therefore, I wanted my final project to be practical and applicable.

At the same time, I’ve become more curious about how technology is used in the business environment, especially in areas like delivery, where smart systems can improve services and make processes more efficient. I wanted to make something that combined this business viewpoint with my technical expertise.

With this project, I was able to use cutting-edge tools like React Native and create a clever recommendation system from the ground up using my own reasoning. It aided in my development and gave me a better understanding of how software can solve real-life problems.

# THEORETICAL CONTEXT AND TECHNOLOGIES

## React Native

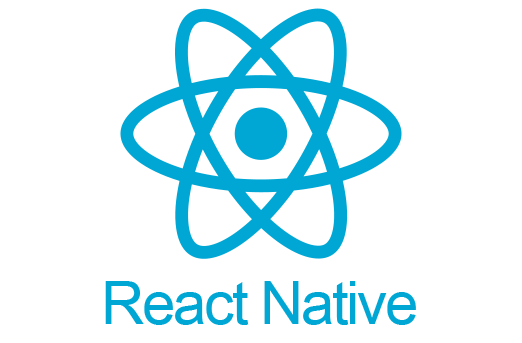


Figure : React Native

React and React Native are two different JavaScript libraries used to build user interfaces. React is mainly used for building web applications, while React Native is used for creating mobile apps. Even though React Native is based on React, they are designed for different platforms and have different purposes.

React Native is a popular mobile app framework based on JavaScript. It allows developers to create mobile apps for both iOS and Android using a single codebase. This means users can build apps for different platforms without writing separate code for each one.

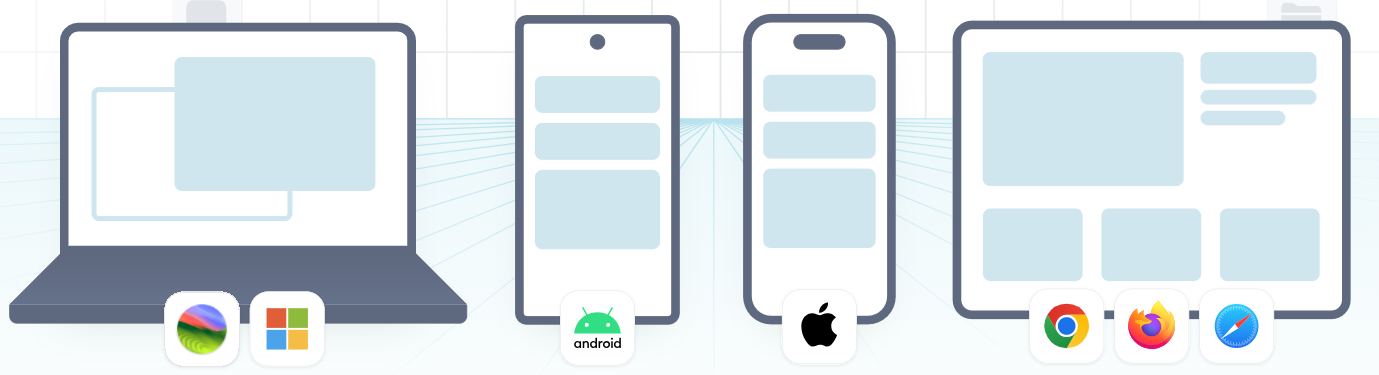


Figure : Cross-platform apps with one codebase [2]

“React Native was released as an open-source project by Facebook in 2015. Since then, it has become one of the most popular tools for mobile app development. It is used in many well-known apps, such as Instagram, Facebook, and Skype.” [1]

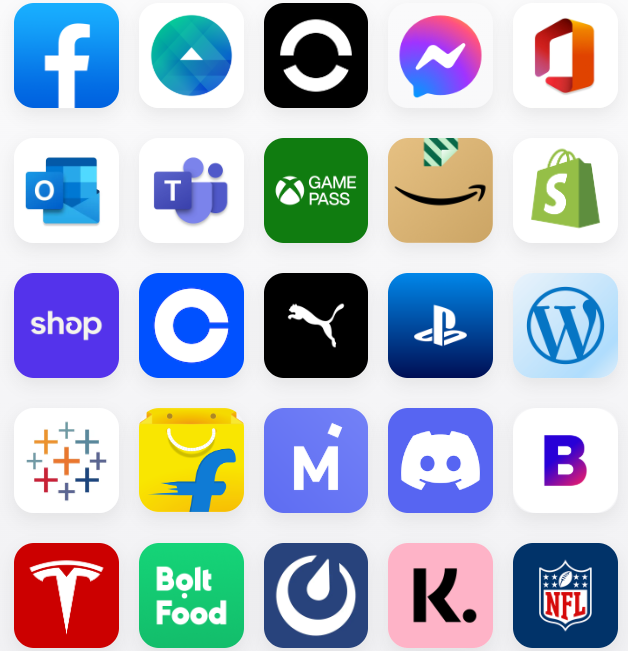


Figure : React Native is being used in thousands of apps [2]

“React Native can be used on its own or within a framework that provides a set of tools and APIs for building production-ready applications. Using a framework such as Expo offers features such as file-based routing, access to universal libraries, and support for creating plugins that interact with native code, without the need to directly handle native files.” [3]

## Expo

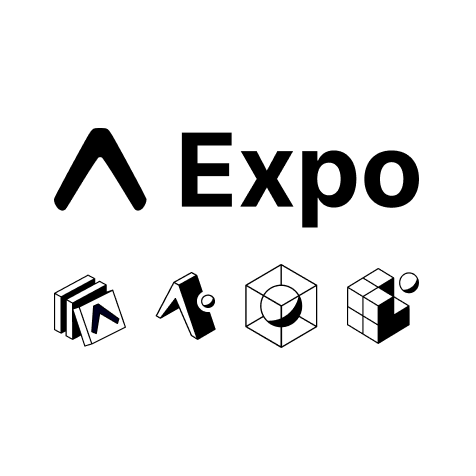


Figure : Expo

“Expo is a framework and platform built around React Native that helps with the development, building, and deployment of apps. It allows developers to create iOS, Android, and web applications using the same JavaScript or TypeScript codebase, with tools that support fast development and testing.” [4]

Expo simplifies the development process with pre-configured tools, libraries, and services.

Expo is a good choose because it assures:

1. Fast Setup: Expo simplifies the initial setup process by removing the need to configure native development environments, allowing developers to focus on core app features.
2. Cross-Platform Support: Applications can be developed using a single codebase with JavaScript and React Native and run on multiple platforms.
3. Managed Workflow: Expo handles many technical details behind the scenes, which is especially helpful for developers with limited mobile development experience.
4. Active Community and Resources: Expo benefits from a strong community and a wide range of tools, libraries, and documentation. [5]

## .NET

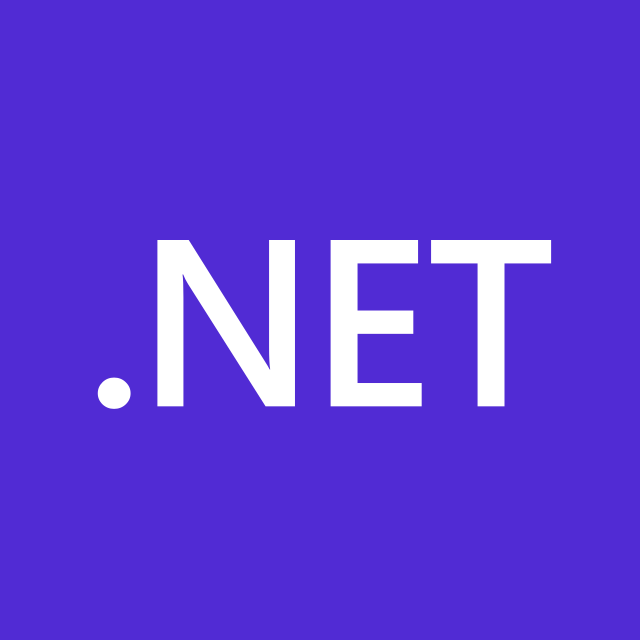


Figure : .NET

“.NET is a free, open-source, and cross-platform framework used to build modern applications and cloud-based services. It supports multiple programming languages, editors, and libraries, allowing developers to build for web, mobile, desktop, games, IoT, and more.” [6]

In this application, .NET was used mainly for the backend, where it helped manage the business logic, user data, and communication with the database.

By using .NET, the backend is able to expose RESTful APIs that allow the React Native mobile app to interact with the server, for example, to register users, process orders, or fetch personalized recommendations.

The main reasons for choosing .NET are:

* Strong support for building secure and efficient web services
* Good integration with databases
* Clean and structured development using C#
* Excellent support and documentation from Microsoft and the developer community

.NET benefits from a vast and mature ecosystem of libraries and tools accessible via NuGet, the official package manager. This enables rapid development by leveraging existing, well-tested solutions for various functionalities, from logging and authentication to data processing and external API integrations, reducing the need to 'reinvent the wheel'.

## Entity Framework



Figure : Entity Framework

Entity Framework (EF) is a modern object-relational mapper (ORM) used in .NET applications to interact with databases in a clean and efficient way. It allows developers to work with data using .NET objects, without writing raw SQL queries. [7]

In this project, Entity Framework was used to create the data access layer, making it easier to manage and query the database. It supports features like LINQ queries, change tracking, data updates, and schema migrations. EF works with different types of databases, such as SQL Server, SQLite, MySQL, and PostgreSQL.

Using EF helped speed up development and ensured that the database logic remained organized, readable, and easy to maintain.

Key Features:

* Cross-Platform: Works on Windows, macOS, and Linux.
* LINQ Support: Allows querying the database using Language Integrated Query (LINQ), making code easier to read and maintain.
* Change Tracking: Keeps track of changes made to objects so that updates to the database can be done automatically.
* Migrations: Makes it easy to apply and manage database schema changes over time. [8]

How EF Core Works:

EF Core uses a special class called DbContext, which acts as a connection between the application and the database. The flow includes three main parts:

1. Model: The C# classes (called entities) represent database tables, and their properties represent the table columns.
2. Context: The DbContext class handles tasks like querying data, adding new entries, updating records, or deleting them.
3. Database Provider: EF Core can work with different types of databases (e.g., SQL Server, MySQL, PostgreSQL, SQLite) using specific database providers. [8]

A key aspect of EF Core is its ability to translate LINQ queries into efficient SQL statements. When a LINQ query is executed against a DbSet, EF Core's query provider analyzes the expression tree and generates the appropriate SQL query, which is then sent to the database. This abstraction allows developers to focus on data manipulation using object-oriented constructs rather than writing database-specific SQL.

EF Core supports different development approaches, including Code-First, where the database schema is generated from the C# model; Database-First, where the model is scaffolded from an existing database; and conceptually, Model-First (less common in modern EF Core, often a variant of Code-First with a designer). This flexibility allows developers to choose the approach that best fits their project's needs.

## SQL Server

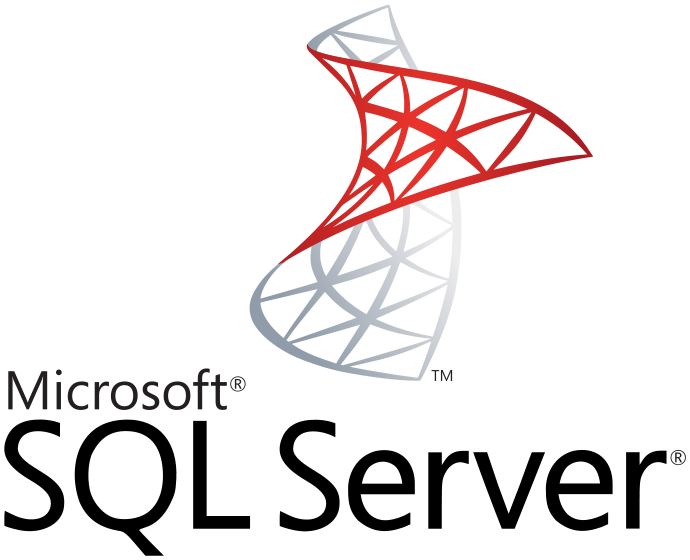


Figure : SQL Server

“Microsoft SQL Server is a popular relational database system. It's designed to handle a wide range of tasks, from managing daily transactions to powering business intelligence and data analytics for companies.

At its core, SQL Server uses Structured Query Language (SQL), the standard language for managing and querying databases. Microsoft's own version, Transact-SQL (T-SQL), allows applications and tools to communicate directly with SQL Server databases.” [9]

Security is a core focus of SQL Server. It includes advanced features like Transparent Data Encryption (TDE) to encrypt data at rest, Row-Level Security to control access to specific rows of data, and robust authentication and authorization mechanisms to protect sensitive information.

SQL Server comes with comprehensive management and development tools, most notably SQL Server Management Studio (SSMS) and Azure Data Studio, which provide graphical interfaces and powerful capabilities for administering, developing, and optimizing databases.

SQL Server offers deep integration with other Microsoft technologies, including Azure cloud services and .NET applications.

## Visual Studio

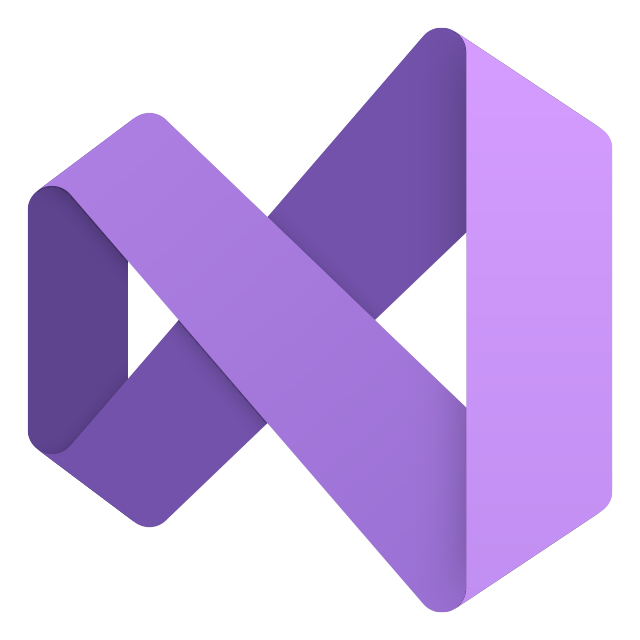


Figure : Visual Studio

“Visual Studio IDE is a comprehensive tool for software development. It lets you write, debug, and build code, and then easily publish your applications. Beyond basic editing and debugging, Visual Studio offers compilers, smart code completion, visual designers, and many other features to streamline the entire development process.” [10]

Visual Studio is highly versatile, supporting a broad range of programming languages beyond C# and enabling development for various platforms including Windows desktop, web, mobile (with Xamarin/MAUI), cloud (Azure), and games.

A significant strength of Visual Studio is its extensive marketplace of extensions. Developers can customize their IDE and add new functionalities, tools, and integrations for almost any need. It also integrates with Git, developers being able to manage their code, track changes, collaborate with teams, and perform operations like committing, branching, and merging directly within the IDE.

Features like GitHub Copilot provide AI-powered code suggestions and completions, directly within the IDE. This means the developer remains in control, guiding the creative process, while Copilot acts as an intelligent assistant, speeding up coding, suggesting boilerplate, and helping explore solutions, thereby embodying the principle: 'You control. AI assists.' [10]

## Visual Studio Code

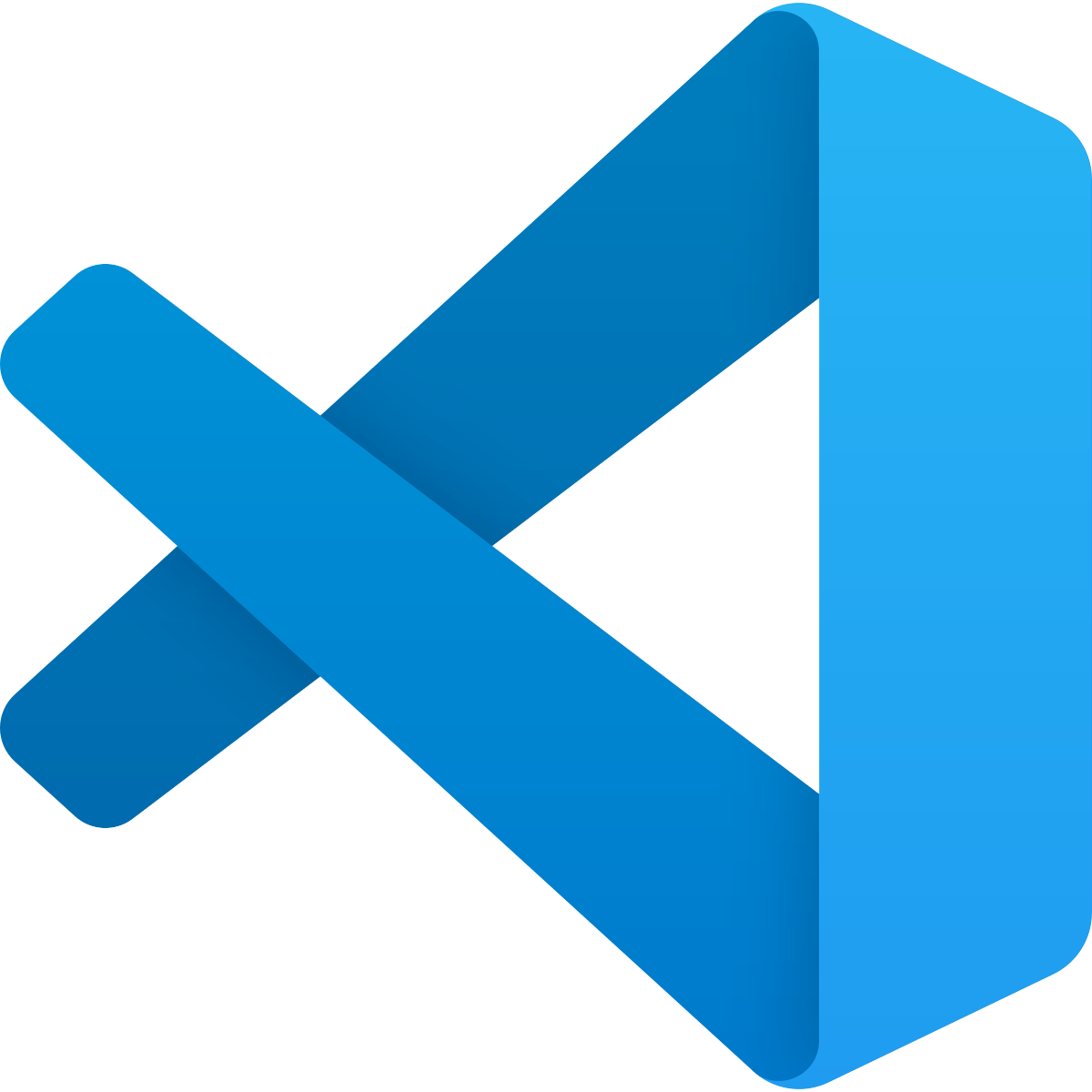


Figure : Visual Studio Code

“Visual Studio Code is a free, stand-alone code editor that runs on Windows, macOS, and Linux. It's a top choice for web and JavaScript developers, offering extensive extensions to support virtually any programming language.” [10]

VS Code has deep, built-in integration with Git, making source code management incredibly intuitive. Developers can perform common Git operations (like committing, branching, and merging) directly from the editor, fostering efficient collaboration and change tracking.

Further enhancing developer productivity, VS Code seamlessly integrates GitHub Copilot. This AI assistant provides real-time code suggestions, completes lines and functions, and can even generate entire code blocks based on natural language comments. It transforms the coding experience by acting as an intelligent pair programmer, embodying the principle that 'You control. AI assists.' by accelerating development while keeping the developer in charge of the final output.

Unlike larger Integrated Development Environments (IDEs), VS Code is renowned for its lightweight nature and fast startup times.

## GitHub

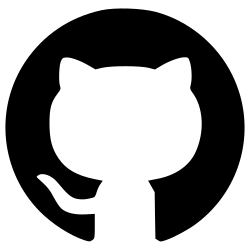


Figure : GitHub

“GitHub is a popular platform where developers can create, store, manage, and share their code. It uses Git for version control, and also offers tools for access control, bug tracking, feature requests, task management, continuous integration, and wikis for every project.” [11]

GitHub is widely recognized as the world's largest hub for open-source projects. It fosters a massive global community of developers, enabling them to discover, contribute to, and learn from millions of public repositories. This collaborative environment is fundamental to the advancement of open-source software.

Further enhancing developer productivity, GitHub integrates GitHub Copilot, an AI-powered coding assistant. This tool provides real-time code suggestions and completions, leveraging the vast amount of code available on GitHub to help developers write code faster and more efficiently.

# Objectives and benchmarks

## Problem Description

Today's customers expect easy home delivery, a trend that puts smaller local food stores, restaurants, and fast-food establishments at a disadvantage. While large chains have invested in advanced delivery systems, these smaller businesses often rely on their delivery personnel but lack the integrated tools to manage the process effectively.

This disconnect leads to significant issues. Manually tracking orders and assigning deliveries are both inefficient and prone to errors, especially during busy periods. Furthermore**, they often struggle to plan efficient delivery routes**, which results in drivers wasting time and fuel, leading to slower deliveries and potentially compromising food quality. Customers also lack real-time updates on their orders, causing frustration and frequent calls to the store.

Ultimately, fragmented communication and outdated systems hinder a business's ability to scale. These problems translate to higher operational costs, lower customer satisfaction, and a missed opportunity for smaller food businesses to compete in the delivery market. There is a clear need for a specialized solution to help these local establishments streamline their delivery operations and enhance customer service using their existing teams.

## Objectives

This application **enables** small local food stores, restaurants, and fast-food **businesses** by centralizing and streamlining their delivery operations. It will provide an intuitive platform for **efficient order processing and assignment** to in-house delivery personnel.

Crucially, the app seeks to enhance customer satisfaction by offering **real-time order tracking** and facilitating **seamless communication** between the store, drivers, and customers. By automating these key aspects, the application intends to significantly **minimize administrative tasks**, improve overall efficiency, and allow these local businesses to **scale their delivery services** effectively in the competitive food market.

Furthermore, by empowering businesses to utilize their delivery teams, this application helps them avoid the high commission fees imposed by large-scale third-party delivery platforms. This allows them to retain a larger share of their revenue from each order, directly contributing to their profitability and sustainable growth.

## Constraints

Each participating store will need to handle its technical setup and put its unique store version of the app on platforms like the Apple App Store and Google Play Store. While our main system helps with this, each store is in charge of managing its app presence.

Another important rule is about how people use the app: everyone must create and keep an active account. This helps us offer a personalized experience and ensures your information is kept safe and secure within the system.

## Identifying User Problems

## Functional Requirements

## Non-Functional Requirements

# Software Architecture

## General Architecture

My application's architecture is highly distributed and modular, designed to allow each food business to operate its own entirely independent digital delivery system. Instead of a single shared backend, each participating store will deploy and manage its own separate instance of the application components.

At the core of each store's independent system are distinct client and server components. On the client side, every store will have a single branded React Native application. This unified app is designed to serve all user roles – customers, store management, and delivery personnel – by presenting different views and functionalities based on the user's login and permissions. This individual store app will be published to app stores under each store's identity.

This React Native frontend for a specific store will communicate directly with that same store's dedicated Backend Service, which is built using ASP.NET API. This means each store maintains its own exclusive backend to manage its unique customer data, order processing, product catalogs, and transaction history. This approach ensures complete data isolation and operational independence for each business.

This architectural design prioritizes the autonomy of each store, allowing them full control over their data and operations. It also inherently provides scalability at an individual store level, ensures security through data compartmentalization, and offers the flexibility for each business to manage its specific delivery ecosystem without sharing backend resources with other stores.

## Database Structure

The data for each individual store's application is managed using Microsoft SQL Server 2019. This robust relational database management system was chosen for its reliability, security features, and seamless integration with the ASP.NET API backend.



Figure : Database ERD (entity-relationship diagram)

### Stored Procedures

The stored procedure was specifically created so it could be added as a distinct step within the SQL Server Agent, enabling automated execution of the contained logic.

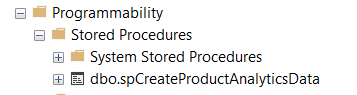


Figure : Stored Procedures Structure

### SQL Server Agent Jobs

Building on the use of stored procedures, I also leveraged **SQL Server Agent Jobs** to manage time-consuming tasks within the database. These jobs are configured to execute specific operations periodically, offloading heavy processing from the live user experience and thereby ensuring that user performance is not negatively impacted during peak hours.

For example, computing all the recommendation cost edges from my algorithm was computed using a stored procedure scheduled every hour.

To ensure system stability, SQL Server Agent can be configured to send email notifications directly to administrators if any scheduled job fails, allowing for immediate intervention.

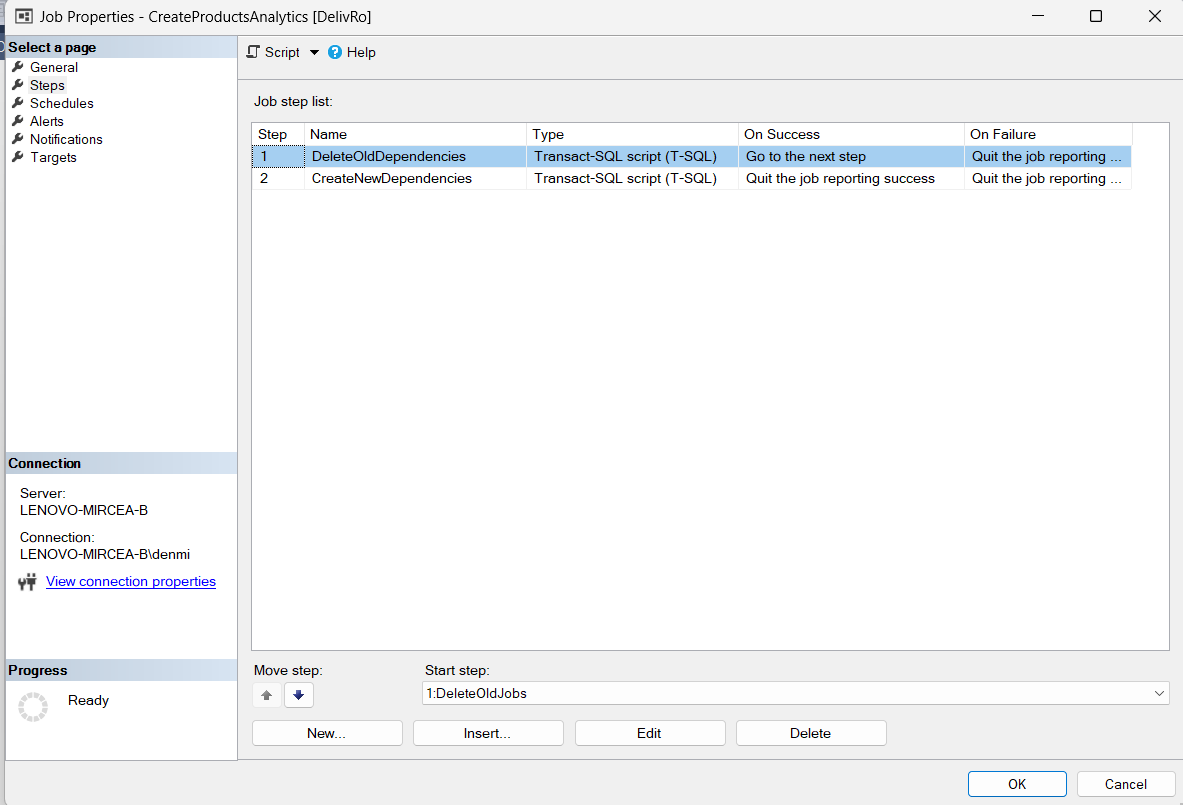


Figure : SQL Server Agent Job Configuration

## Backend Structure

## Frontend Structure

# Software Implementation

Languages and Libraries

## React Native Maps

JWT

Use case diagrams

Database Implementation

# Conclusions and future improvements

# Bibliography

|  |  |
| --- | --- |
| [1] | M. Budziński, "What Is React Native? Complex Guide for 2024," 2024. [Online]. Available: <https://www.netguru.com/glossary/react-native>. |
| [2] | "Learn once, write anywhere.," 2025. [Online]. Available: <https://reactnative.dev/>. |
| [3] | "Get Started with React Native," 14 04 2025. [Online]. Available: <https://reactnative.dev/docs/environment-setup>. |
| [4] | "Create amazing apps that run everywhere," 2025. [Online]. Available: <https://docs.expo.dev/>. |
| [5] | R. B, "Expo: A Gateway to Simplified Cross-Platform App Development," 24 02 2024. [Online]. Available: <https://medium.com/@talktorahul.b/expo-a-gateway-to-simplified-cross-platform-app-development-820184506c2f>. |
| [6] | "Learn .NET," 2025. [Online]. Available: <https://dotnet.microsoft.com/en-us/learn>. |
| [7] | "Entity Framework documentation hub," [Online]. Available: <https://learn.microsoft.com/en-us/ef/>. |
| [8] | R. Patel, "A Beginner’s Guide to Entity Framework Core (EF Core)," [Online]. Available: <https://medium.com/@ravipatel.it/a-beginners-guide-to-entity-framework-core-ef-core-5cde48fc7f7a>. |
| [9] | A. H. C. S. Rajul A, "Microsoft SQL Server," 15 03 2024. [Online]. Available: <https://www.techtarget.com/searchdatamanagement/definition/SQL-Server>. |
| [10] | "Visual Studio 2022," [Online]. Available: [https://visualstudio.microsoft.com/#vs-section](https://visualstudio.microsoft.com/%23vs-section). |
| [11] | "GitHub," [Online]. Available: <https://en.wikipedia.org/wiki/GitHub>. |

# Source Code

# Site-ul web al proiectului

# CD / DVD

Autorul atașează în această anexă obligatorie, versiunea electronică a aplicației, a acestei lucrări, precum și prezentarea finală a tezei.



# Index

B

Bibliografie 9

C

CUPRINSUL xi

D

Dimensiuni 3

F

Figuri 4

Formulele matematice 4

I

Ilustrațiile 4

L

Legenda 6

LISTA FIGURILOR xii

LISTA TABELELOR xiii

R

Referințe web 10

S

Structura documentului 2

T

Tabele 5