Leqso Sikmashvili

📍 Georgia, Tbilisi | 📞 +995 595 05 11 15 | ✉ leqsosiymashvili.wth@gmail.com

LinkedIn: https://www.linkedin.com/

Instargarm: Instagram

GitHub: Leqso006 (Leqso)

Professional Summary

Highly motivated Software Engineer and Economics student with 3+ years of experience in C#, .NET, and database engineering. Adept at problem-solving, market analysis, and creating efficient solutions tailored to clients' needs. Proficient in Forex trading and economic statistics, with a keen interest in leveraging software skills for innovative problem-solving.

Skills

- Programing Languages : C#, .NET,
- DataBase : Sql

- FrameWorks & Tools : ASP.NET, Web API, Entity Framework, Ado.Net, Dapper, Microsoft Sql Server

- Design Patterns : SOLID Pattern , OOP, UOW(UnitOfWork) , Repository Pattern, Dependency Injcetion
- Languages: Georgian (Native), English (C1), Russian (C1)

Education

Caucasus University (2022–2026)
- Bachelor's in Economics
- Relevant Coursework: Macro/Microeconomics, Financial Accounting, Economic Psychology

Experience

Tera Bank (Present (6 Months on Position))

IT Specialist

Software Engineering
- Developed custom software solutions in C# and .NET to meet client’s goals

Projects

Fitness App

• A fitness application that offers users premium features for free — features that are usually paid in other apps. – *In Development*

Bank System

• A basic banking system that allows user registration and email verification via a sent code. It creates a bank account, allows multiple cards to be linked to it, and includes various other functionalities. All data is stored in a specially designed SQL database, and every action is logged using **Serilog**, both in the database and locally (in the console and as log files). – *In Development*

TourOperator API

• A tour management system that handles tours, user creation, tour assignments, and many other features. Built with **Swagger** for documentation, and all data is stored in a specially designed SQL database.

Georgian Tax Partners (Tax Advisory Platform)

Project Type: Full Stack Web Application / Custom CMS
Architecture: Headless Architecture (Decoupled Frontend & Backend)

1. Technical Stack

Backend: ASP.NET Core Web API (C#)

Database: Microsoft SQL Server (MSSQL)

Frontend: HTML5, CSS3, Vanilla JavaScript (ES6+)

Documentation: Swagger / OpenAPI

Authentication: JWT (JSON Web Tokens)

Hosting/Tools: IIS / Azure (Assumed), Visual Studio, Postman

2. Backend & API Architecture

The core of the system is a robust RESTful API built with **.NET Core**.

RESTful Endpoints: Designed controllers for every entity (News, Team, Pricing, Services).

Swagger Integration: Integrated Swagger UI to visualize and test API endpoints, ensuring the frontend team (or future mobile apps) can easily understand the data structure.

Repository Pattern (Optional - good to mention if true): Used dependency injection and repositories to keep the database logic separate from the controllers.

3. Security & Admin Panel (JWT Implementation)

The system features a secure Admin Panel for content management, protected by **JWT (JSON Web Token)** authentication.

Stateless Authentication: The API is stateless. When the Admin logs in, the server validates credentials and issues a signed **JWT**.

Token Lifecycle:

Login: Admin sends username/password -> Server validates -> Returns `access_token`.

Storage: Frontend stores the token securely (e.g., `localStorage`).

Requests: Every request to protected routes (e.g., `POST /api/news`, `DELETE /api/team`) includes the token in the `Authorization: Bearer <token>` header.

Validation: The server validates the token signature on every request to ensure the user is authorized.

4. Database Design (SQL)

A custom relational database schema was designed to handle dynamic content.

Complex Relations:

Pricing Plans: Implemented a **One-to-Many** relationship where one `PricingPlan` has multiple `PricingSections` (Sub-plans).

Data Integrity: Used Primary Keys, Foreign Keys, and constraints to ensure data consistency.

Stored Data: Handles multi-language text fields (e.g., `Title_EN` and `Title_DE`) directly in the database to support the frontend language switcher.

5. Frontend Logic & Performance

The frontend is built with pure JavaScript (no heavy frameworks) for maximum performance, using a "Smart Caching" strategy.

Smart Caching System: Implemented a custom caching layer using `sessionStorage`.

On load, the app checks the API for a **System Version**.

If the version matches the local storage, it loads data **instantly** from memory (0 latency).

If the version differs, it fetches fresh data from the API and updates the cache.

Multi-Language (i18n): Dynamic language switching (English/German). The JS detects the selected language and renders the correct database field (e.g., `item.description_DE` vs `item.description`).

Marketing Integration: Integrated Google Global Site Tag (gtag.js) for ad tracking and user analytics.

Recommendators

David Dvali

• LinkedIn: https://www.linkedin.com/in/dvali/

Letter From Recommendation :

https://drive.google.com/file/d/1z4yXmi7BXI0fGfE52GJEW4EGtHXiuUJP/view?usp=drive_link